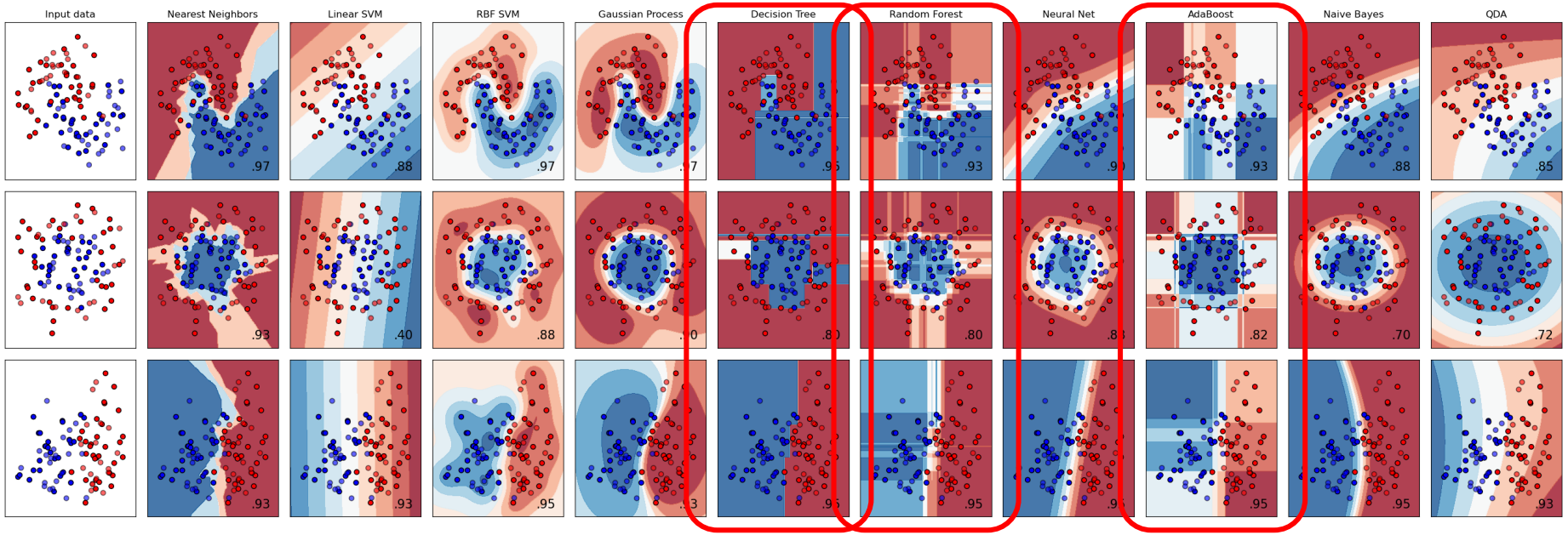
Classification problem revisited: decision trees

Materials for this section:

- https://scikit-learn.org/stable/modules/tree.html
- https://scikit-learn.org/stable/modules/ensemble.html
- StatQuest: https://statquest.org/video-index/

Decision Trees

- Decision trees a very simple but robust and efficient approach to classification problem



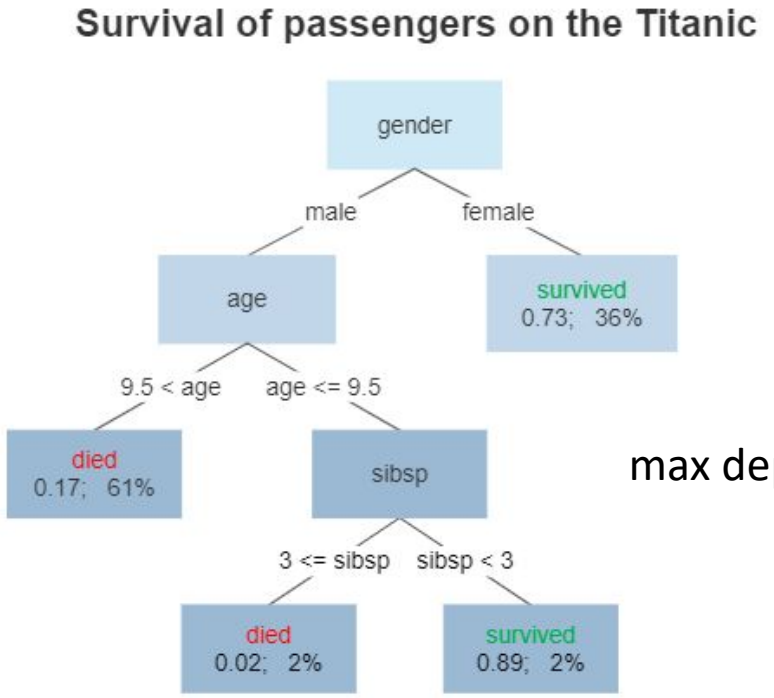
https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

```
classifiers = [
    KNeighborsClassifier(3),
    SVC(kernel="linear", C=0.025),
    SVC(gamma=2, C=1),
    GaussianProcessClassifier(1.0 * RBF(1.0)),
    DecisionTreeClassifier(max_depth=5),
    RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
    MLPClassifier(alpha=1, max_iter=1000),
    AdaBoostClassifier(),
    GaussianNB(),
    QuadraticDiscriminantAnalysis()]
```

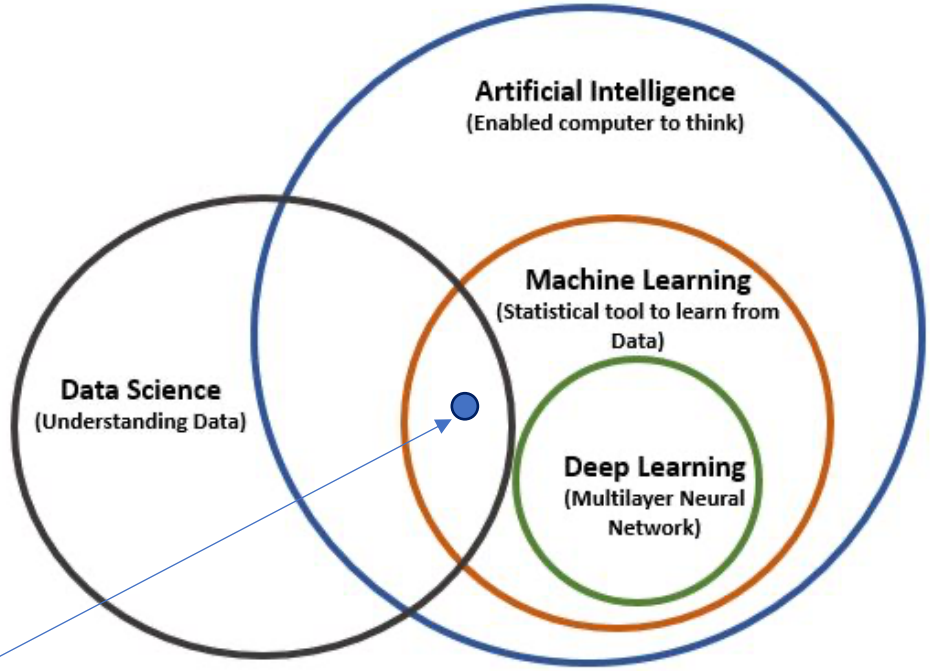
- k-Nearest Neighbors
- SVM
- Gaussian Process
- Decision Trees
- Random Forest
- Neural Nets
- AdaBoost
- Naive Bayes
- QDA

Decision Trees

- Decision Trees are a non-parametric supervised learning method used for **classification** and **regression**.
- We go through **decisions** about sample (represented in the *branches*) to conclusion about the sample target value (represented in the *leaves*).
- Decision trees breaks the input (**feature vectors**) space into regions and has separate class or value for each region.



max depth = 3



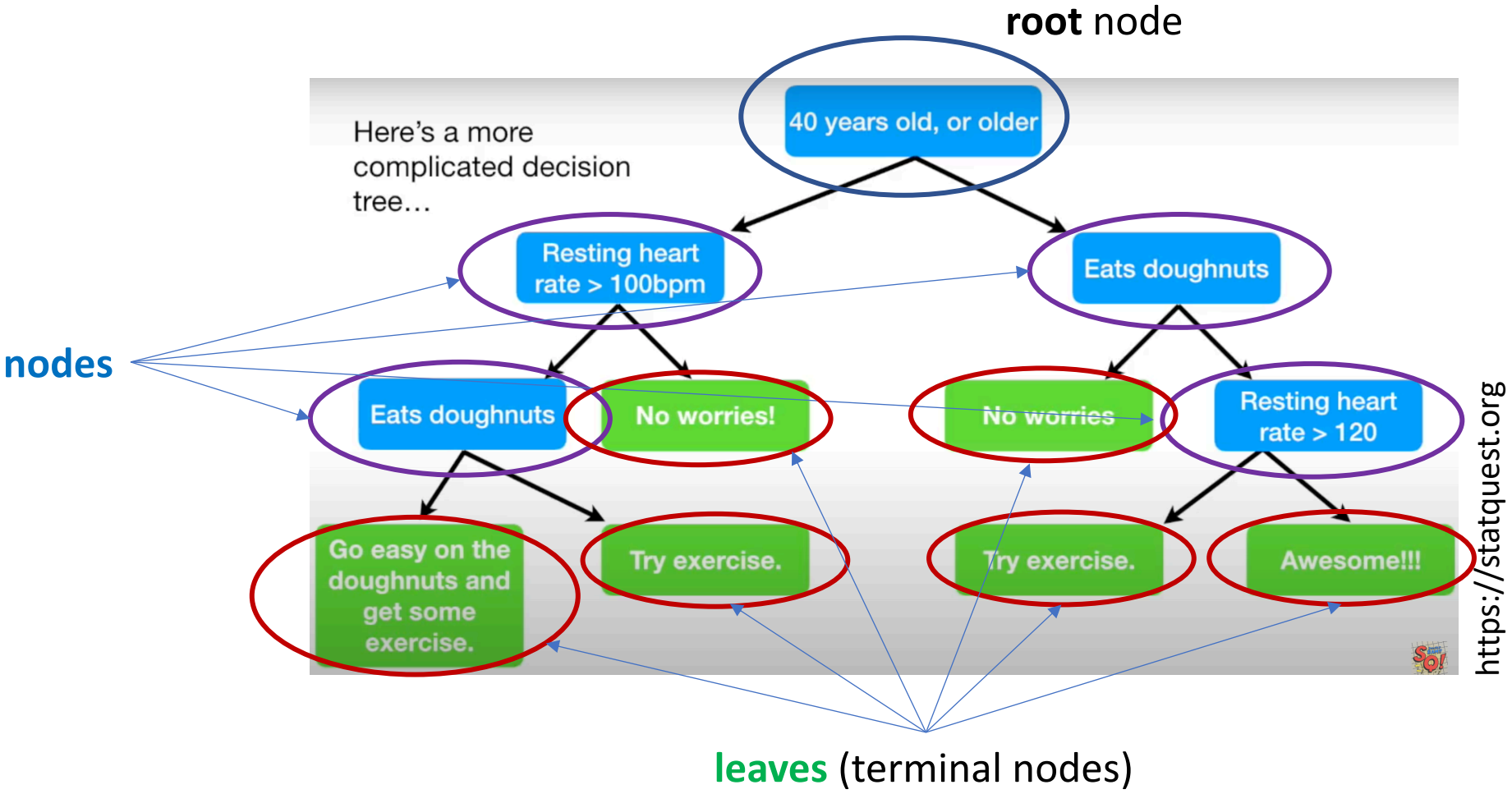
DTs belong to intersection between data science and ML

https://bcdworld.com

en.wikipedia.org/wiki/Decision_tree_learning

Decision Trees

Tree structure: nomenclature



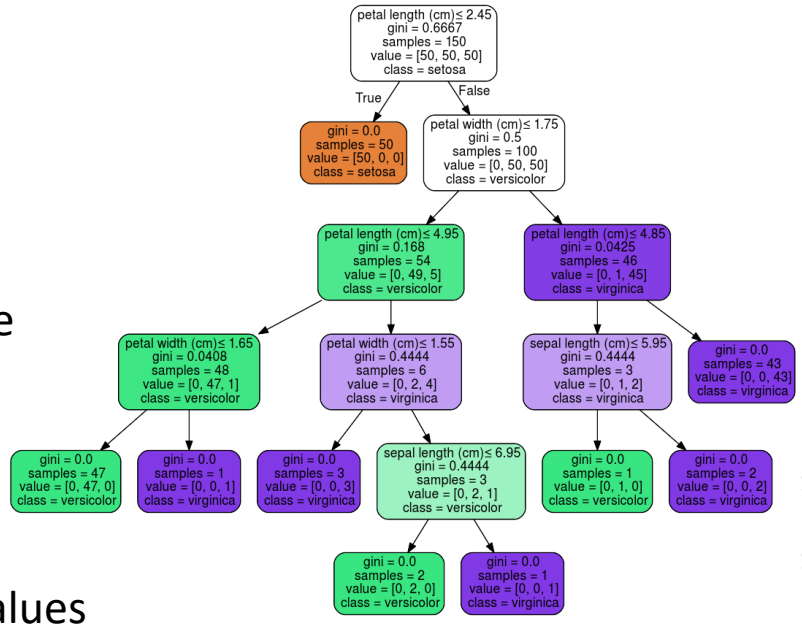
Decision Trees

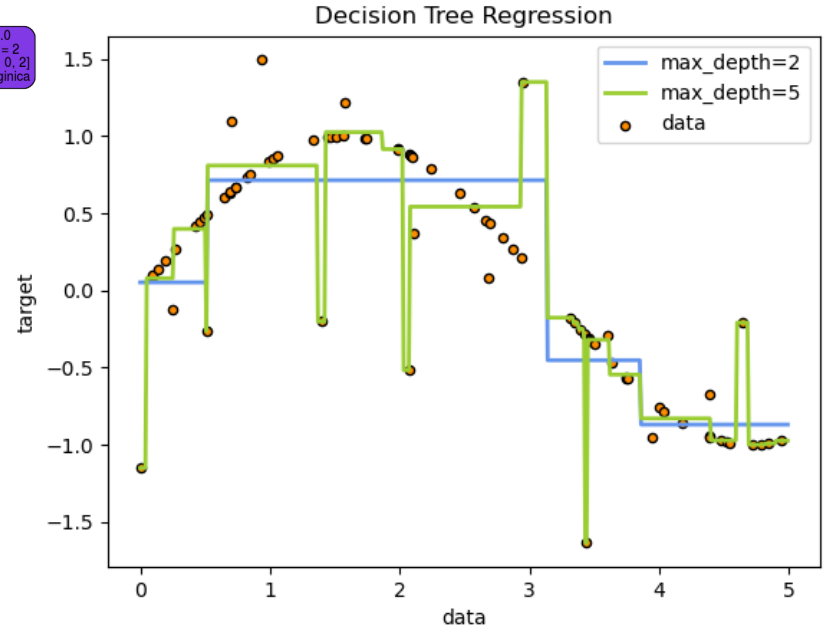
classification tree

- discrete class to which the data sample belongs

regression tree

- the target variable takes continuous values
- regression trees are pretty useful when input data have multiple (e.g. dozens of) features



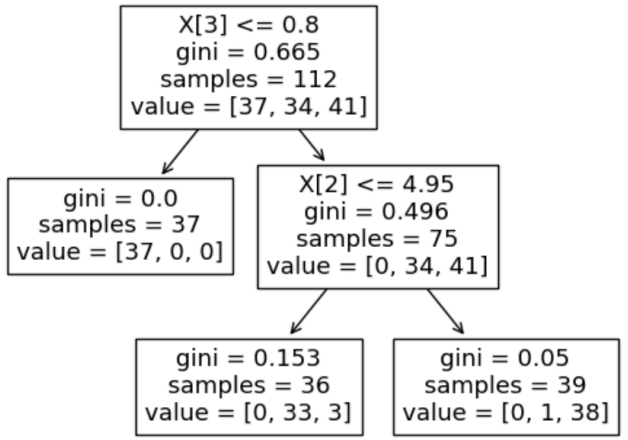


https://scikit-learn.org/stable/modules/tree.html

Decision Trees

Some advantages of decision trees are:

- Simple to understand and to interpret
- Can be easily implemented after rewriting to a series of *if-else* statements
- Uses a ”white box” model. If a given situation is observable in a model, the explanation can be easily expressed in Boolean logic.
- The cost of using a tree (evaluating the model) is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data



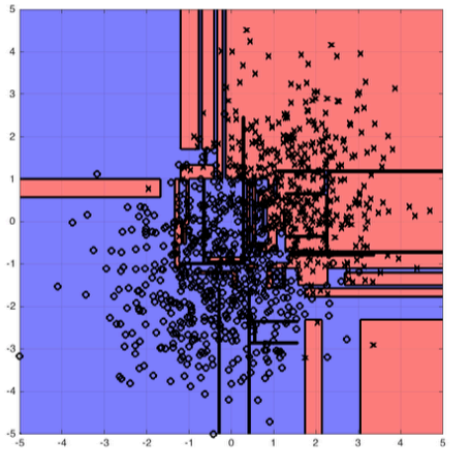

```
node=0 is a split node: go to node 1 if X[:, 3] <= 0.800000011920929 else to node 2.
        node=1 is a leaf node.
        node=2 is a split node: go to node 3 if X[:, 2] <= 4.950000047683716 else to node 4.
                node=3 is a leaf node.
                node=4 is a leaf node.
```

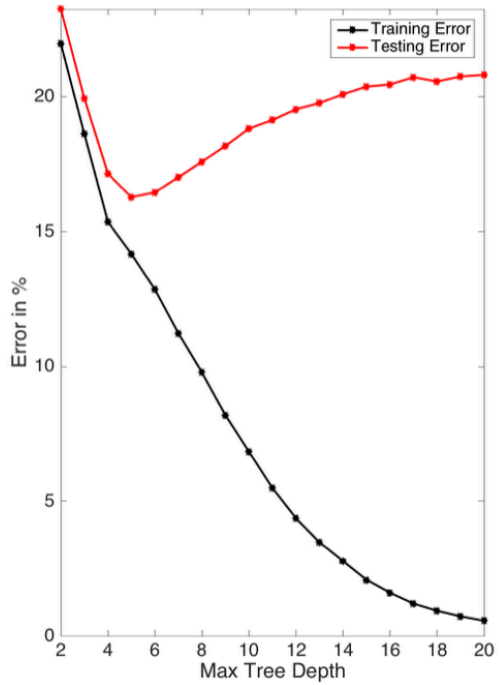
https://scikit-learn.org/stable/auto_examples/tree/plot_unveil_tree_structure.html

Decision Trees

Some disadvantages:

- Decision-tree learners can create over-complex trees that leads to overfitting.
- Decision trees can be unstable.
- Predictions of decision trees are neither smooth nor continuous.
- Decision tree learners create biased trees if some classes dominate.





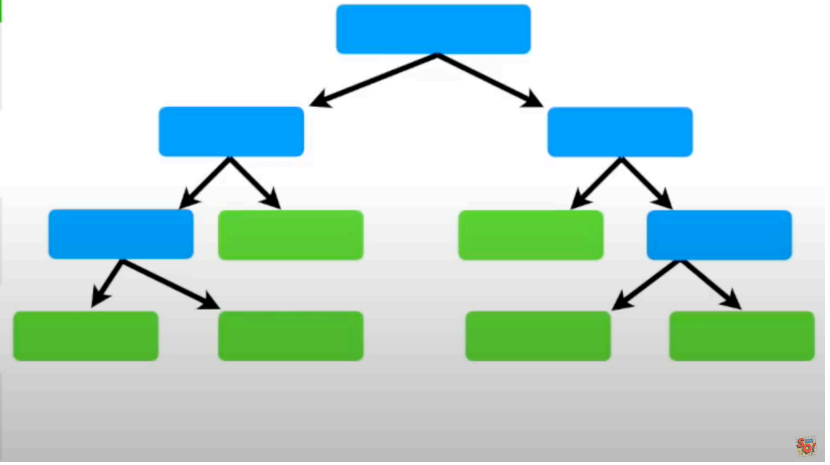
https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote17.html

Decision Trees: growing

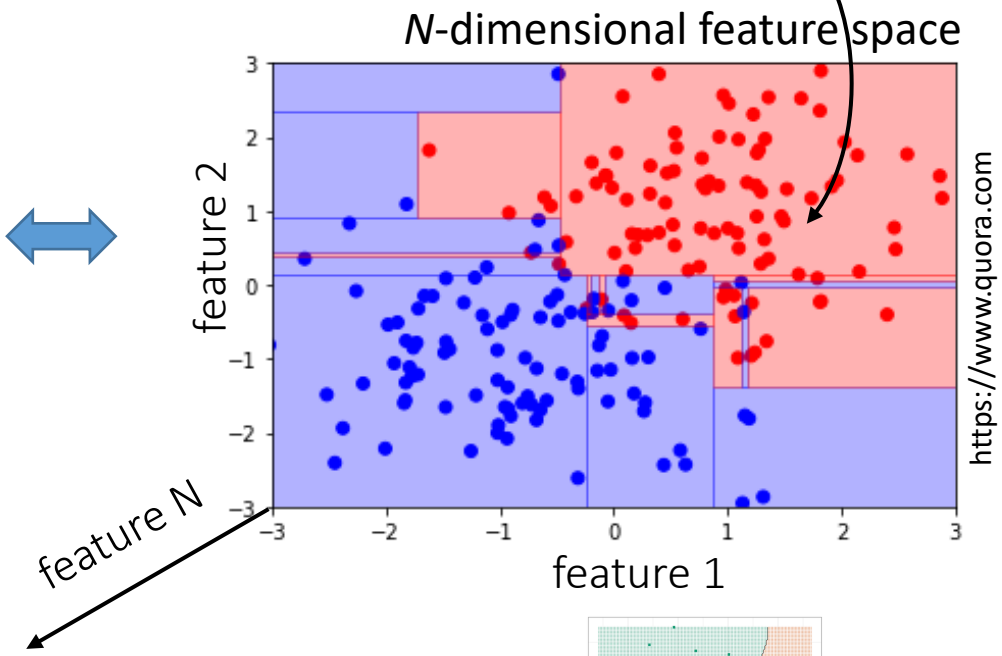
How to train DT on a given data?

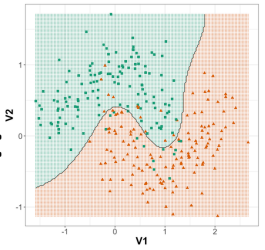
Given L training feature vectors $x_i \in R^N,\ i = 1,..,L$, each storing N features, a decision tree recursively partitions the feature space such that the samples with the same classes or similar target values are grouped together.

Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc...	etc...	etc...	etc...



https://statquest.org/video-index [Decision Trees Part 1]

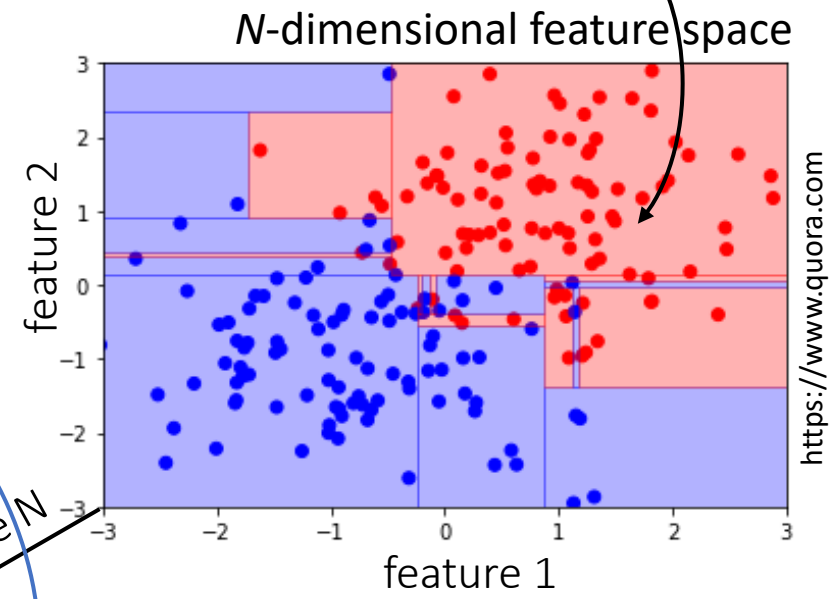


(e.g. for SVM: )

Decision Trees: growing

How to train DT on a given data?

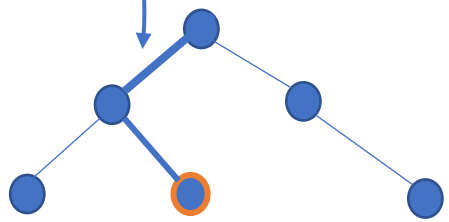
Given L training feature vectors $x_i \in R^N,\ i = 1, .., L$, each storing N features, a decision tree recursively partitions the feature space such that the samples with the same classes or similar target values are grouped together.



1. Decision made at each node splits given subset into further subsets and so on.
2. How to chose feature f_m (from unused at given path) that spits given subset Q_m at given node m best?
   – various DT algorithms:

$$IG(Q_m, f_m) = H(Q_m) - H(Q_m|f_m) = \max,$$
$$f_m = \underset{\text{remaining } f}{\operatorname{argmax}}\ IG(Q_m, f)$$

- ID3: best feature minimizes **entropy** (*information gain* – reduction in entropy - is the largest)
- C4.5: handles also numerical values (not only discrete yes/no as ID3), implements “error-based” *pruning* technique
- CART: best feature minimizes **Gini impurity** coefficient, uses “minimal cost-complexity” pruning



Decision Trees: growing

CART (Classification and Regression Tree) **algorithm in details**

Let the data at node m be represented by Q_m with N_m samples. For each candidate split $\theta = (j, t_m)$ consisting of a feature j and threshold t_m, partition the data into $Q_m^{left}(\theta)$ and $Q_m^{right}(\theta)$ subsets

$$Q_m^{left}(\theta) = \{(x, y) | x_j <= t_m\}$$
$$Q_m^{right}(\theta) = Q_m \setminus Q_m^{left}(\theta)$$

The quality of a candidate split of node m is then computed using an impurity function $H(.)$,

$$G(Q_m, \theta) = \frac{N_m^{left}}{N_m} H(Q_m^{left}(\theta)) + \frac{N_m^{right}}{N_m} H(Q_m^{right}(\theta))$$

$$H(Q) = 1 - \sum_{k=1}^{K} q_k^2(Q)$$

$q_k(Q)$ – a fraction of samples from Q, labelled with a class k

Select the parameters that minimises the impurity

$$\theta^* = \text{argmin}_\theta \, G(Q_m, \theta)$$

Recurse for subsets $Q_m^{left}(\theta^*)$ and $Q_m^{right}(\theta^*)$ until the maximum allowable depth is reached, $N_m < \min_{samples}$ or $N_m = 1$.

https://scikit-learn.org/stable/modules/tree.html#mathematical-formulation

clearly explained: https://youtu.be/7VeUPuFGJHk?t=210

Decision Trees

Useful tips when implementing tree models:

- Decision trees tend to overfit on data with a large number of features
  (tree with few samples in high dimensional space is very likely to overfit)

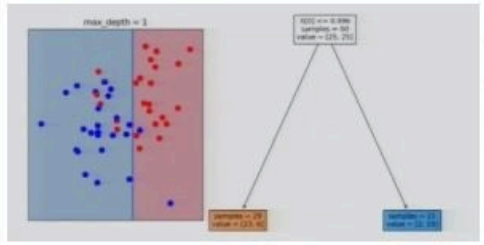
- It may be useful to priory perform *dimensionality reduction* (e.g. PCA or LDA) or *feature selection*

- It is also useful to analyze decision tree structure (after training)

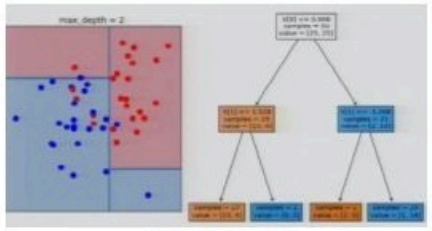
- Tuning structure hiperparameters:
  - use **max_depth** to control the size of the tree to prevent overfitting:
    - use max_depth = 3 at start and then gradually increase the depth
    - each additional level doubles number of samples required to populate the tree
  - use **min_samples_split** or **min_samples_leaf** to ensure that multiple samples inform every decision in the tree

- *Balance* dataset before training (e.g. apply samples weights)

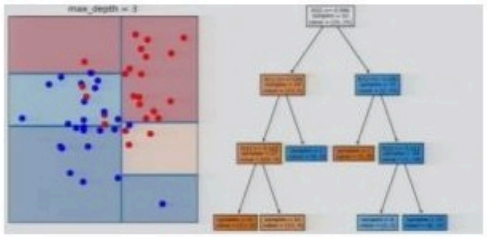
Decision Trees: overfitting



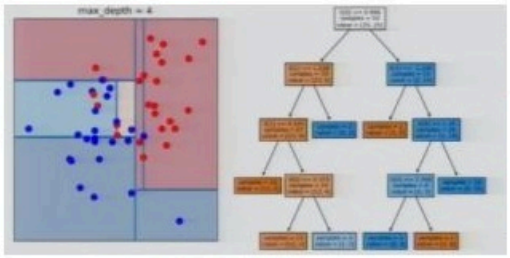
Max Depth = 1



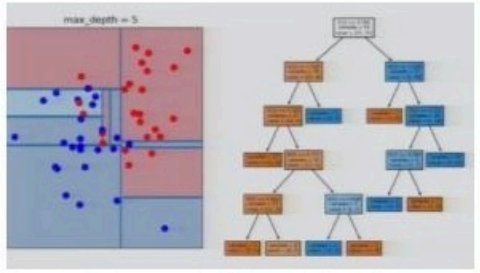
Max Depth = 2



Max Depth = 3



Max Depth = 4



Max Depth = 5

Max Depth = 8

Controling max depth

https://www.datavedas.com/decision-trees/

Pruning provides another option to control the size of a tree

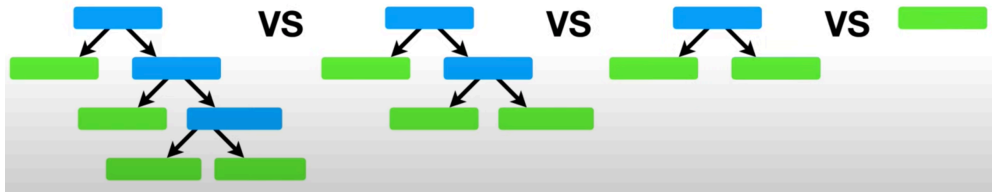
https://ogrodniktomek.pl/

Decision Trees: pruning

Minimal cost complexity (post) pruning.

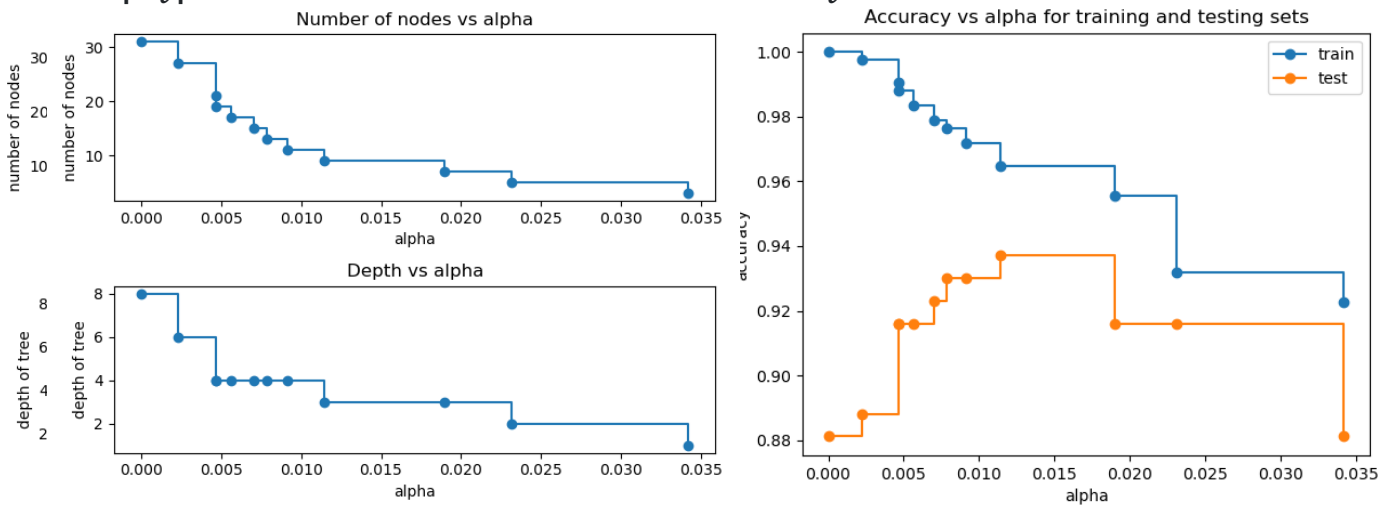
Assume that we already have trained tree T

- generate a series of subtrees: $T_0, \dots, T_m$, where T_0 is the initial tree and T_m is the root alone



https://statquest.org

- calculate sum of squared errors – residuals SSR (using train subset) for each of subtree
- and cost-complexity score $R = SSR + \alpha |T_i|$, where $|T_i|$ – number of leaves in subtree T_i
- hiperparameter α is optimized using validation subset:
  - we chose α that tree pruned with such assumed alpha has the lowest SSR on validation subset
- finally, we chose subtree which minimizes R



https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html

Ensemble methods

Ensemble methods
(where the true story begins):

- combine **ensemble** of multiple *weak learners* to create single *strong learner*
- weak learner is simplified base-model (e.g. decision tree) with poor performance

There are two popular families of ensemble methods:

- **bagging** (**b**ootstrap **agg**regat**ing**), e.g. *random forest*
- **boosting**, e.g. *AdaBoost, Gradient Boost*

Idea: predictions are made by **averaging** the predictions from all the individual base-models:

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x_{\text{test}})$$

prediction made by a given weak learner

Averaging reduces variance of the model:

$$\text{Var}(\hat{f}) = \frac{1}{B^2}\sum_{b=1}^{B}\text{Var}(f_b) \approx \frac{B\sigma_b^2}{B^2} = \frac{1}{B}\sigma_b^2$$

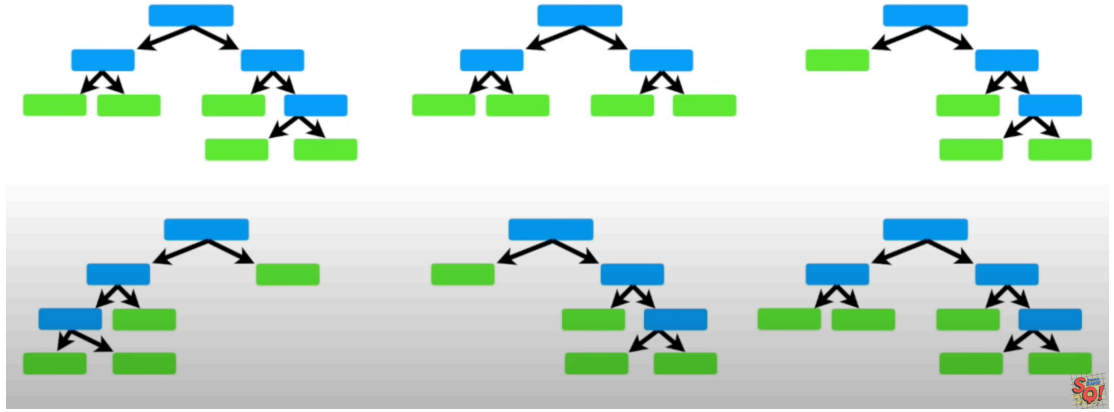
typical variance for a weak learner

without increasing the bias.

Random Forests

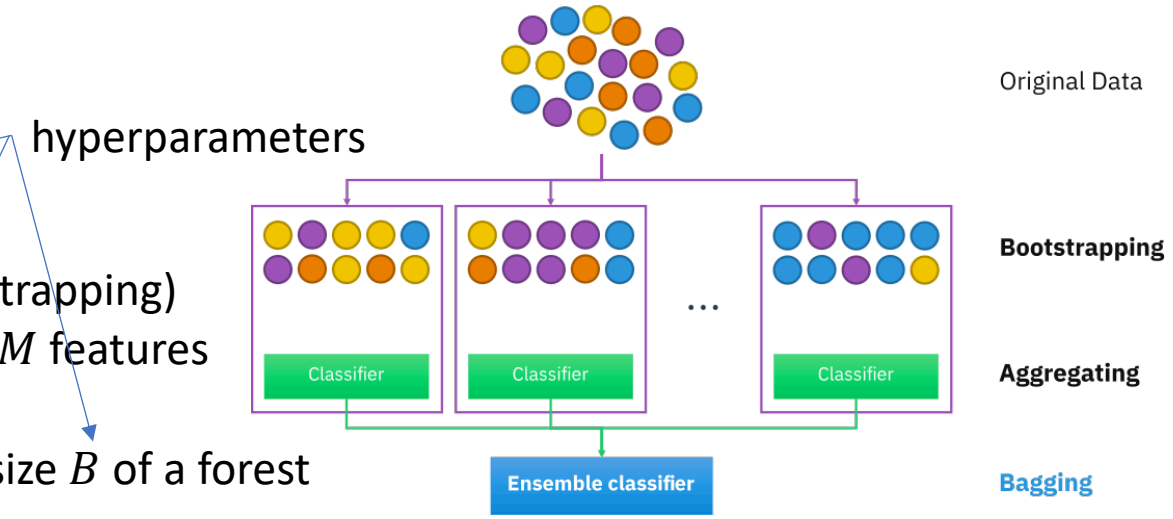
Building and evaluating Random Forests

- we create a **forest** of *random* trees
- random because each tree is trained on a random subsets of samples and features (training many trees on a same set would give the same tree many times)
- each tree contributes to a final prediction



https://statquest.org

- Assume that we have training set with *N* feature-vectors – each of length *M* (no of features)
- Sample, **with replacement**, *n* training samples from *N* (bootstrapping)
- When training tree on this subset, **at each node** select $m \ll M$ features (usually $m \sim \sqrt{M}$)
- Repeat the procedure for next trees until you reach desired size B of a forest
- The final result is prediction of a random forest is an average predictions (for regression) or a majority *vote* (classification) -- this stage is called aggregating



bagging = **b**ootstrap **agg**rega**ting**

en.wikipedia.org/wiki/Bootstrap_aggregating

Random Forests

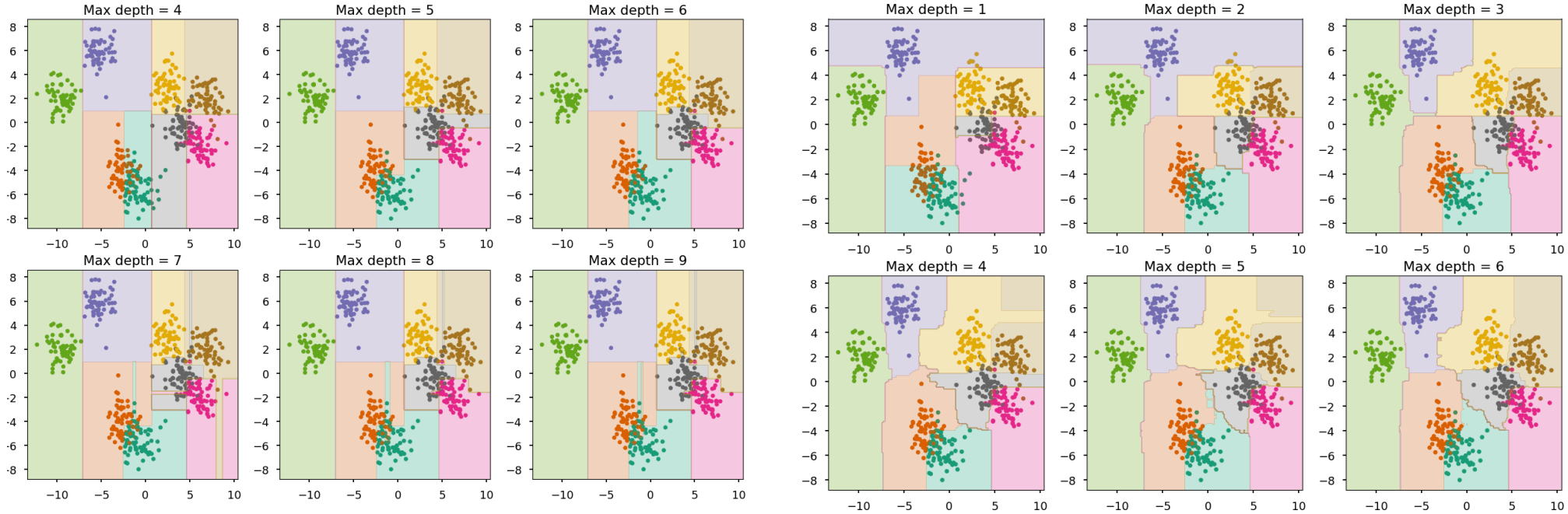
Out-of-bag error:

- used for machine learning models using bootstrap aggregation (like random forests) **instead** of cross-validation
- bagging involves random sampling with replacement: some samples are not used in the training process (out-of-bag samples) and therefore can be used to calculate test error
- OOB error is the mean prediction error on each training sample x_i using only the trees that did not have x_i in their bootstrap subset.

---

source: https://tomaszgolan.github.io/introduction_to_machine_learning

Decision Tree vs Random Forrest



Boosted Trees

The idea is similar to bagging, key differences are:

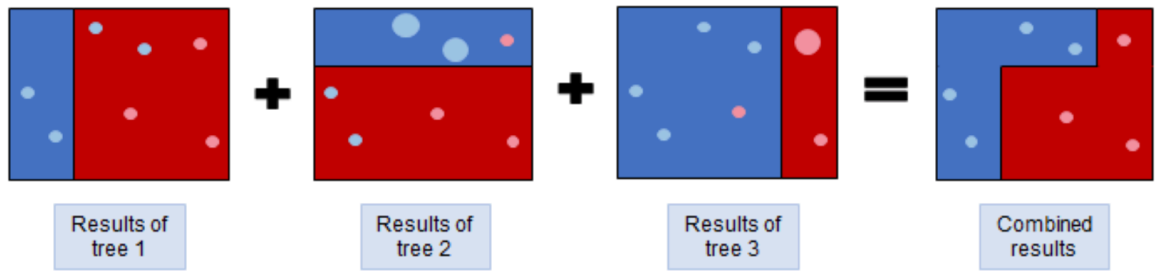
- Data is *reweighted* every time a weak learner is added (so future learners focus on misclassified samples)
- The final prediction is weighted average (better classifiers have higher weights)
- **AdaBoost** is considered to be one of the best classifiers today

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

reweighting

https://en.wikipedia.org/wiki/Boosting



https://towardsdatascience.com/the-ultimate-guide-to-adaboost-random-forests-and-xgboost-7f9327061c4f