Decisions Trees, Random Forests and Ensemble Methods

Benoit Gaüzère

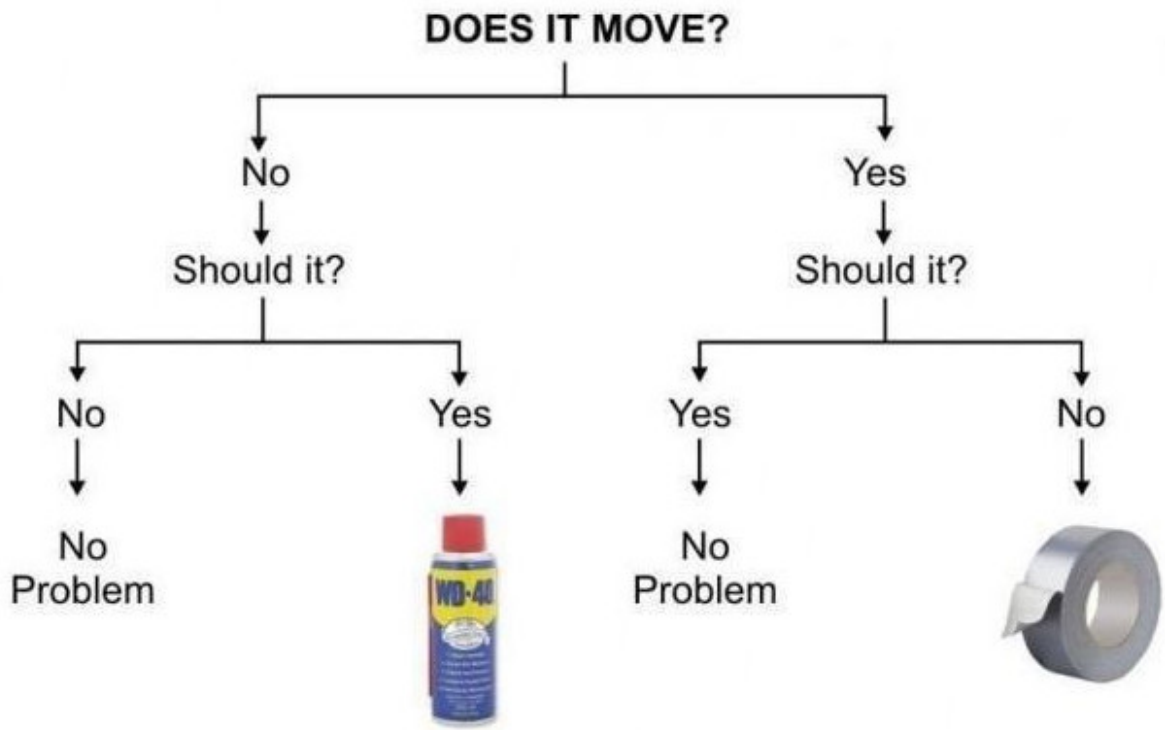
INSA Rouen Normandie - Laboratoire LITIS

October 28, 2023

Outline

Introduction to Decision Trees

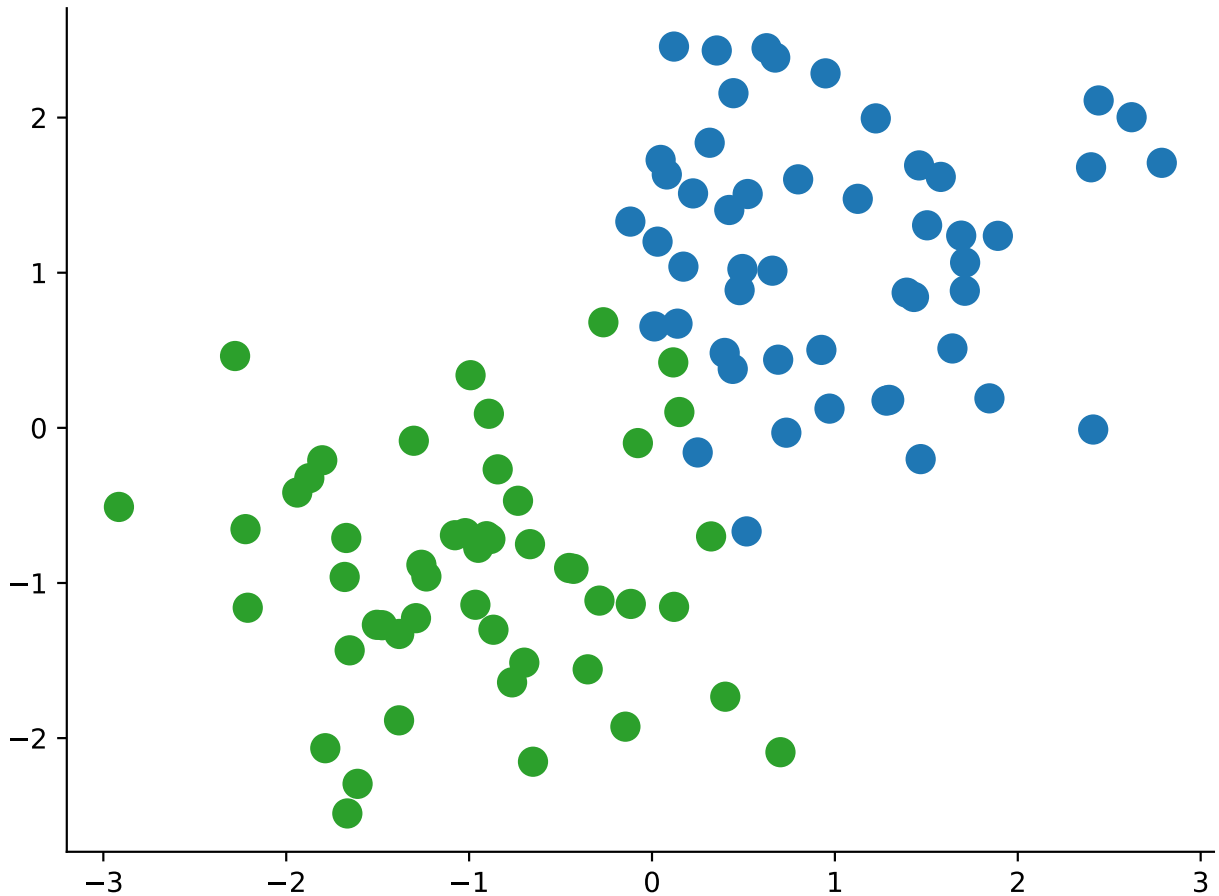
- Supervised learning for classification and regression.
- Simple to understand and interpret.
- Recursive algorithm to construct the decision trees



Decision Tree

Principle

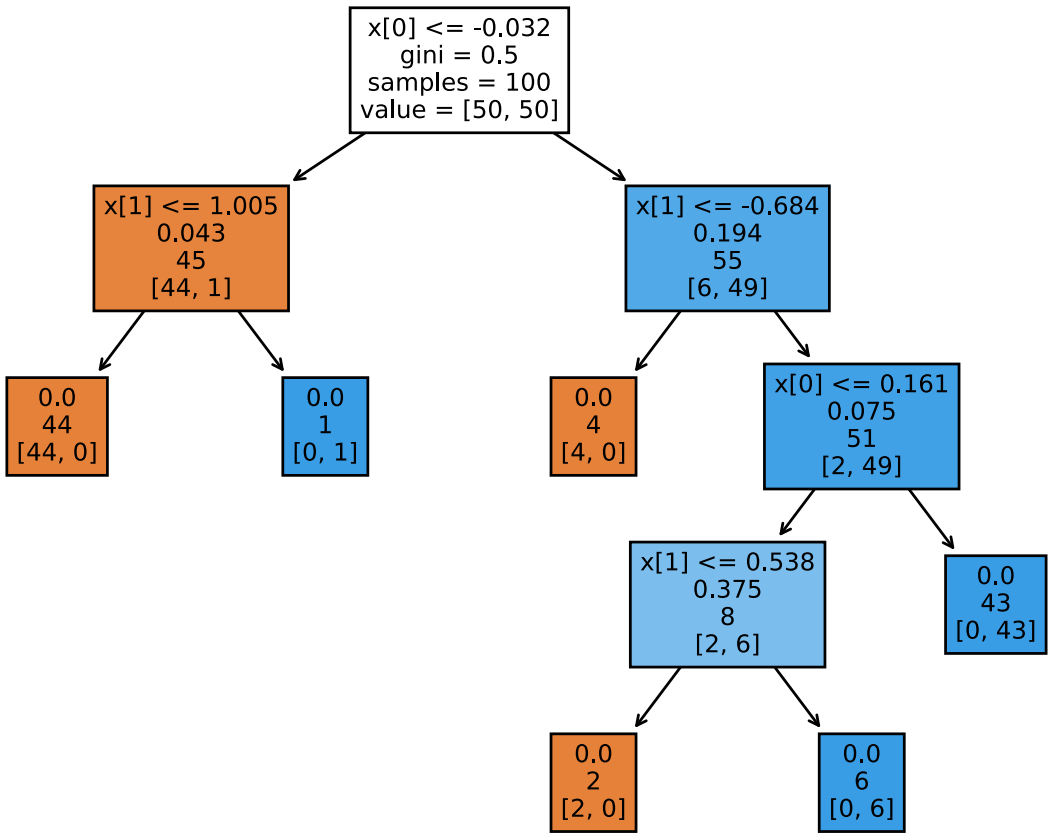
Learn decision rules to separate the data.



Decision Tree

Principle

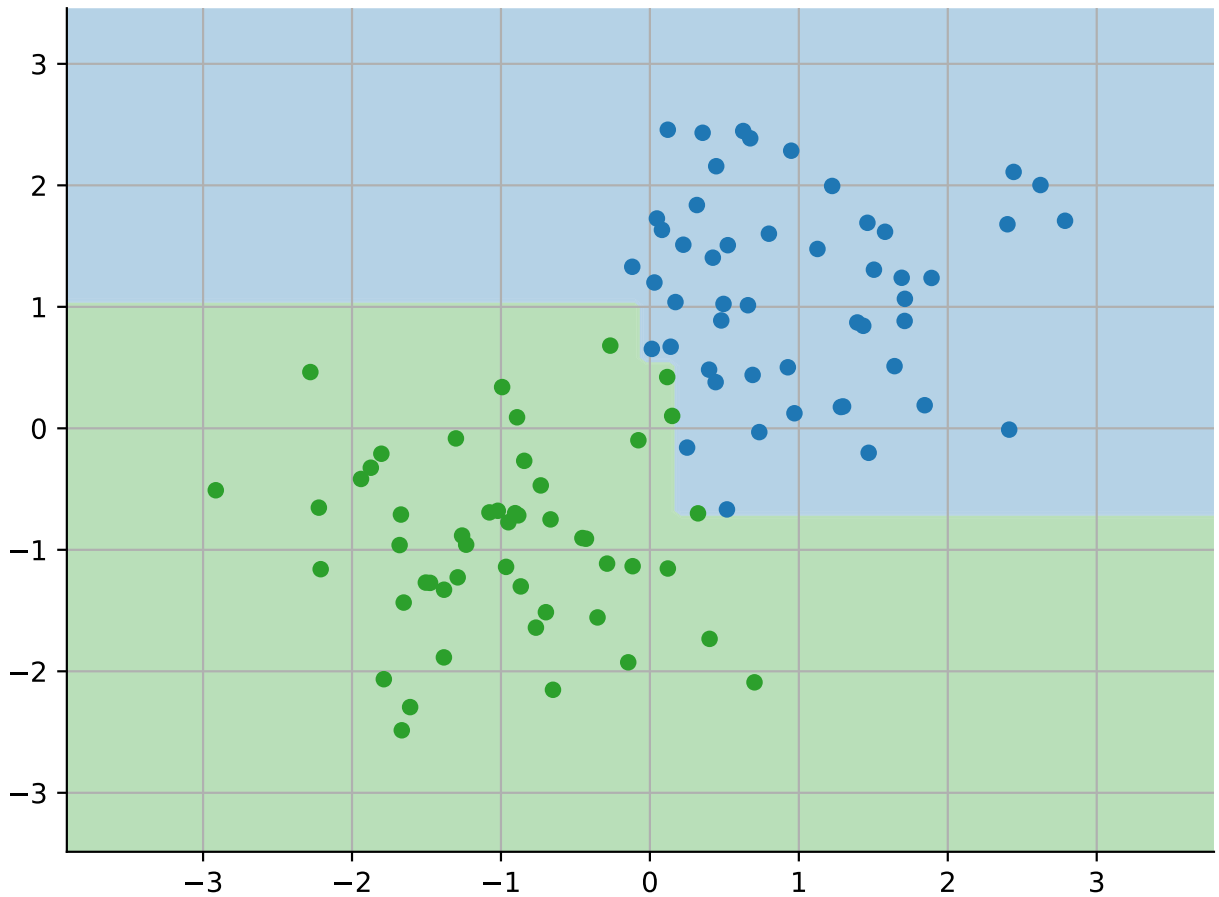
Learn decision rules to separate the data.



Decision Tree

Principle

Learn decision rules to separate the data.



How to learn Decision Trees ?

Split Data

1. Choose a feature f
2. Compute a threshold t_f

How to learn Decision Trees ?

Split Data

1. Choose a feature f
2. Compute a threshold t_f

How to determine t_f ? (and f)

How to learn Decision Trees ?

Split Data

1. Choose a feature f
2. Compute a threshold t_f

How to determine t_f ? (and f)

Maximize Information Gain

$$IG(D_p, f) = I(D_p) - \frac{N_l}{N_p} I(D_l) - \frac{N_r}{N_p} I(D_r)$$

with:

- I : measure of impurity
- D_p, D_l and D_r the datasets corresponding to parent, left node and right node.

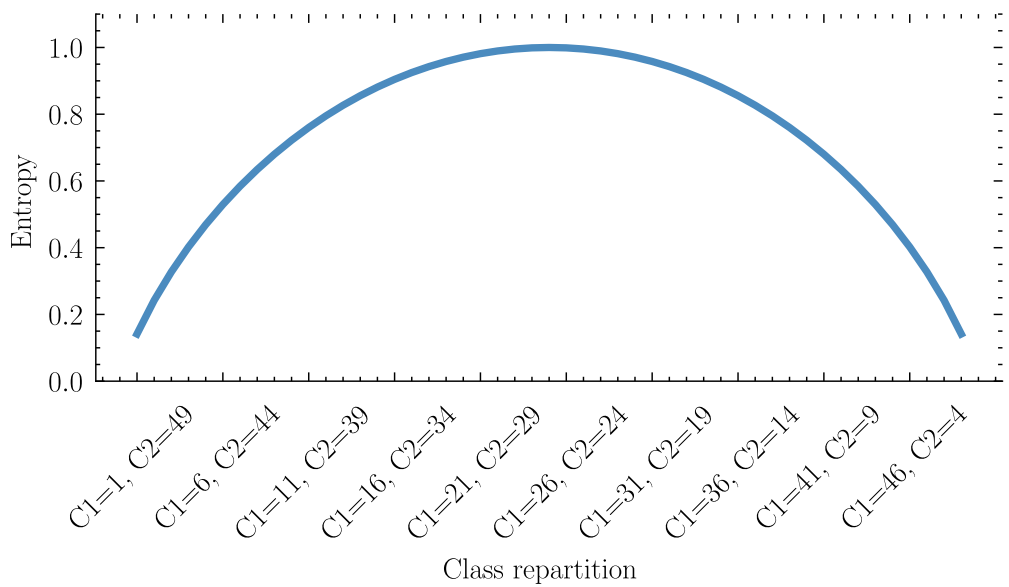
Impurity measures I

To minimize !

Entropy

$$I_e(p) = -p * log_2(p) - (1-p) * log_2(1-p)$$

with p characterizing the probability for a sample to belong to one class in a given node ($P(C_k|D)$).



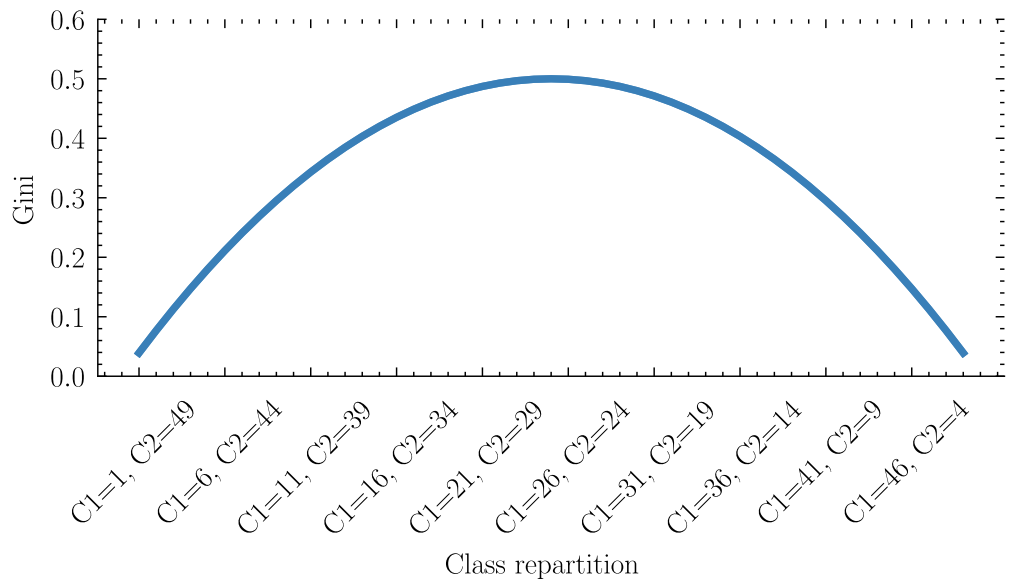
Impurity measures II

To minimize !

Gini Impurity

$$I_G(p) = \sum_{i=1}^{c} p_i * (1 - p_i) = 1 - \sum_{i=1}^{c} p_i^2$$

for $c = 2, I_G(p) = 1 - p^2 - (1 - p)^2$.



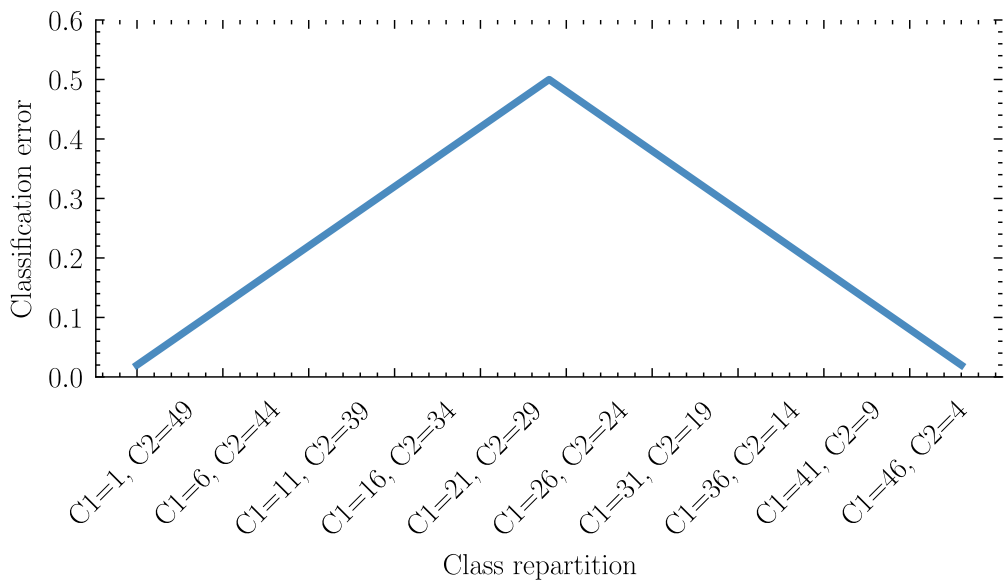
Impurity measures III

To minimize !

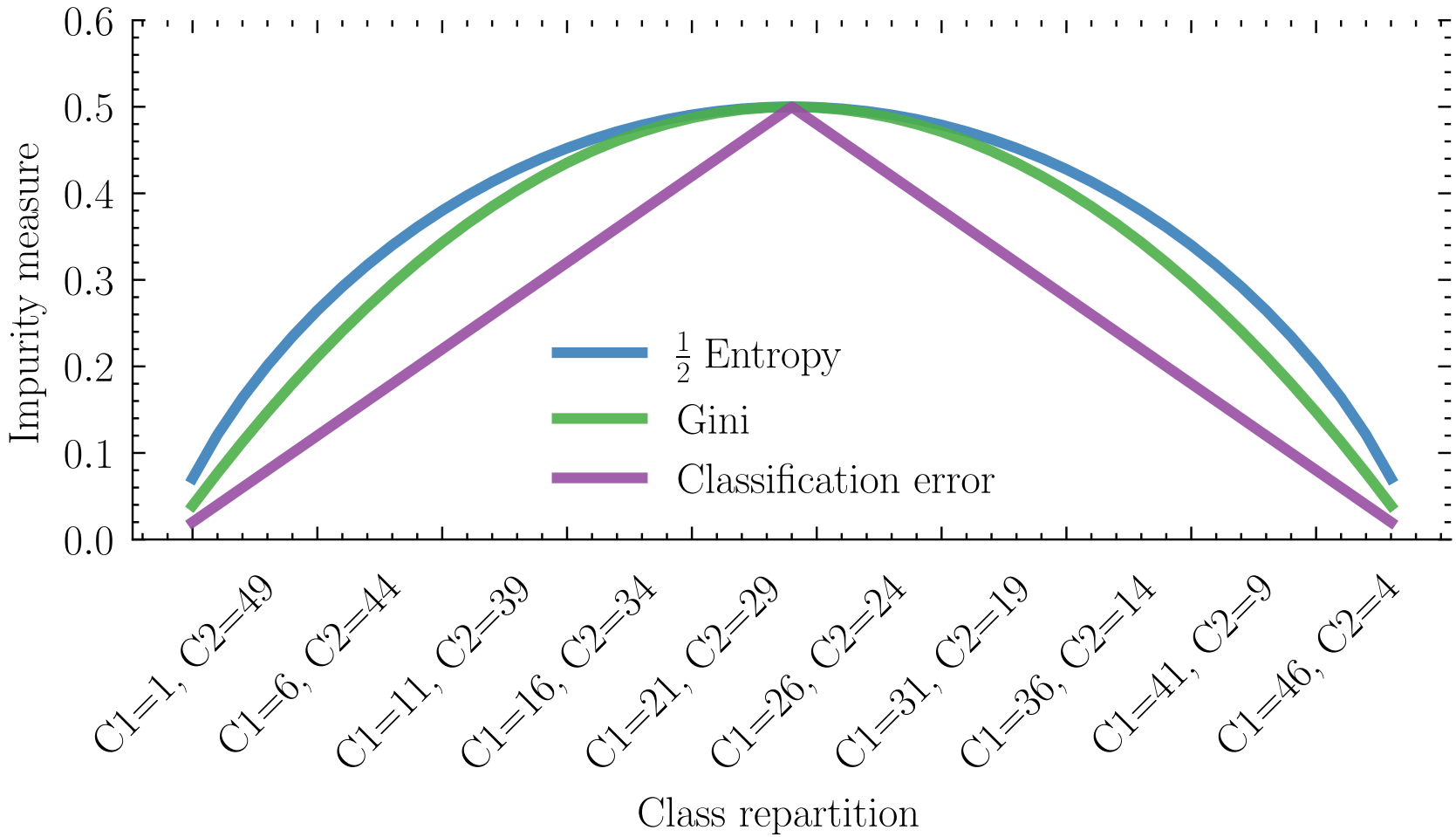
Classification error

$$I_E(p) = 1 - \max_{i \in 1...c} p_i$$

Less sensitive to good repartition.



Differences between impurity



The CART algorithm

function SPLIT_RECUR(D)
  if not a leaf **then**
    $\theta^\star = \text{argmax}_\theta \, IG(D, \theta)$
    $D_l, D_r = \text{partition}(D, \theta^\star)$
    SPLIT_RECUR(D_l)
    SPLIT_RECUR(D_r)
  end if
end function

- with $\theta = (f, tf)$
- Recursively select the best split which maximize the IG
- When does it stop ?

The CART algorithm

function SPLIT_RECUR(D)
    if not a leaf **then**
        $\theta^\star = \text{argmax}_\theta \, IG(D, \theta)$
        $D_l, D_r = \text{partition}(D, \theta^\star)$
        SPLIT_RECUR(D_l)
        SPLIT_RECUR(D_r)
    end if
end function

- with $\theta = (f, tf)$
- Recursively select the best split which maximize the IG
- When does it stop ?
  - Only one class in D
  - Max depth reach
  - Min number of samples D reached

Hyperparameters of Decision Tree

Maximum depth

Specify the maximal depth of the tree. An higher depth will make dedicate categories, but prone to overfit.

Min number of splits

Same action as previous one.

$\rightarrow$ Both are used to terminate the recursive operation

Building a decision tree - the code

```
1    from sklearn.tree import DecisionTreeClassifier
2    max_depth = 10
3    criterion = 'gini'
4    clf = DecisionTreeClassifier(max_depth=max_depth,
5    criterion=criterion)
6    clf = clf.fit(X, y)
7    ypred = clf.predict(X)
```

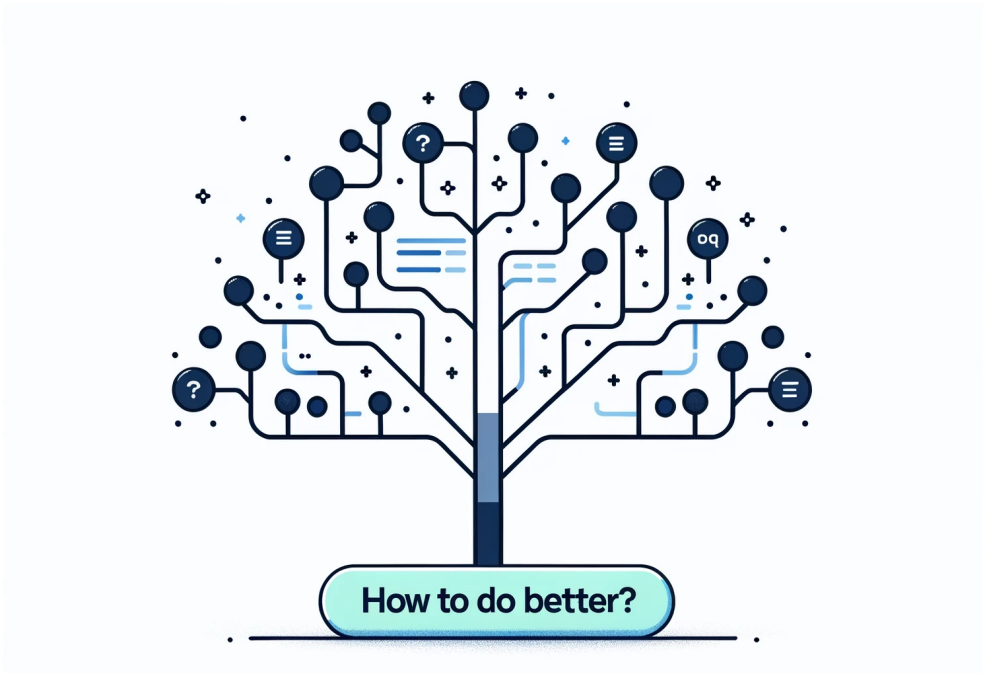
- User guide for hyperparameters : link
- $\Rightarrow$ Notebook
- the documentation

Limitations

- Simple yet effective algorithm
- Prone to overfitting
  $\rightarrow$ one leaf $\Leftrightarrow$ one sample

Limitations

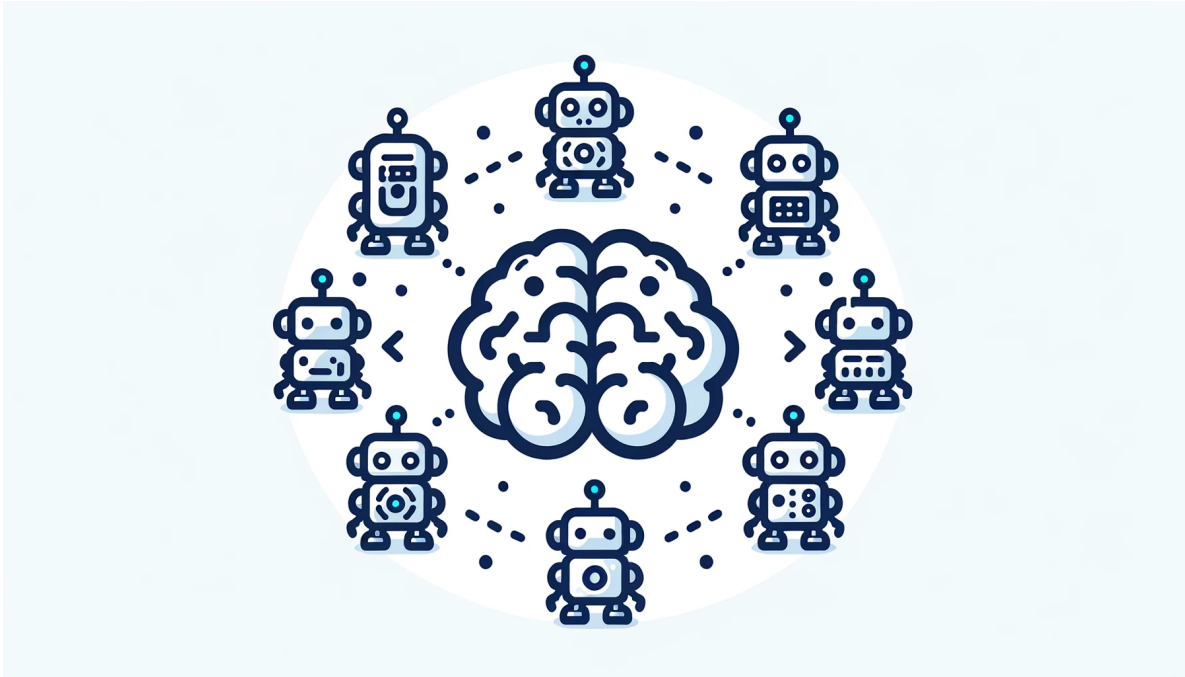
- Simple yet effective algorithm
- Prone to overfitting
  $\rightarrow$ one leaf $\Leftrightarrow$ one sample



Ensemble Methods

Idea

United we stand



How to combine them ?

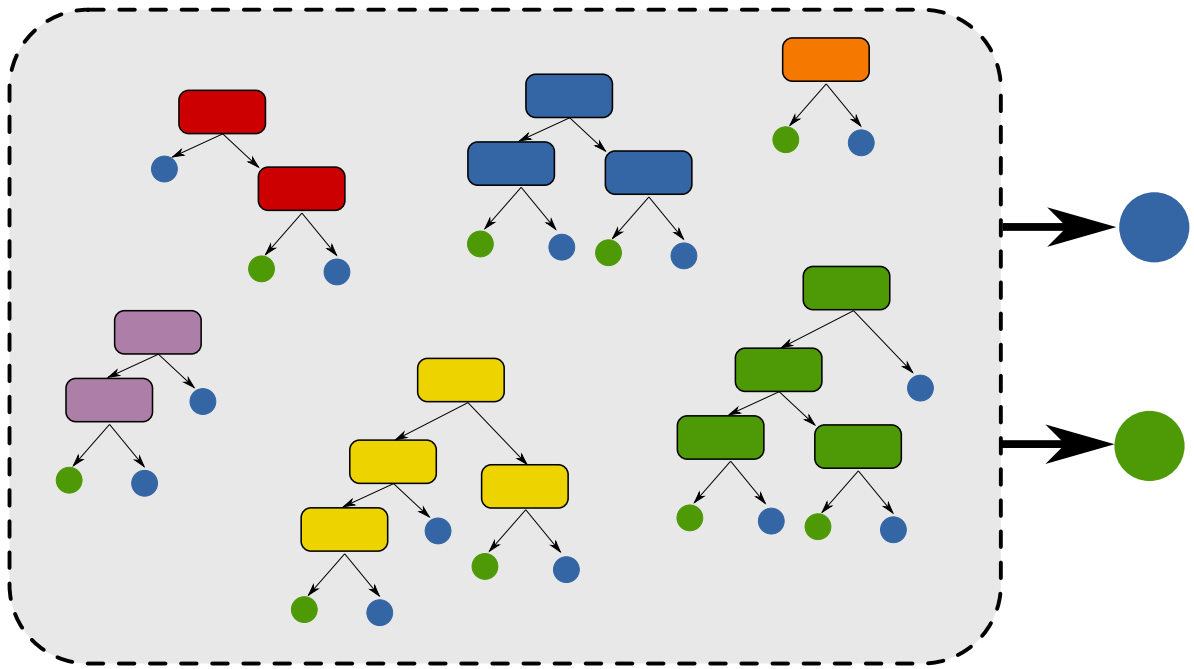
- Majority voting, Bagging and Boosting

Random Forests

Random Forests

Principle

- Combine many decision trees to learn complex functions
- Ensemble methods, majority voting
- Bagging Breiman [1996]



Algorithm summarization

1. Randomly choose n examples (bootstrap)
2. Build a decision tree from the bootstrap
   - 2.1 Randomly select d features
   - 2.2 Split according to best pair feature/threshold
3. Repeat k times
4. Aggregate decision by majority vote or average probability

Random Forests Hyperparameters

Number of trees

Adjust the number of trees composing the forests

- low number : fast to compute, but less accurate
- high number : slower to compute, but more accurate up to some number

Number of features

Determine the number of features to be used when splitting the data

- See the guidelines of `scikit-learn`

Tree depth

Specify the maximal depth of tree. An higher depth will make dedicate categories, but less generalizable.

Random Forests : the code !

```
1  from sklearn.ensemble import RandomForestClassifier
2  n_estimators = 20 # the number of trees in the forest
3  max_depth = None # expand as you can
4  max_features = "sqrt" # RTFM
5  clf = RandomForestClassifier(n_estimators=n_estimators,
6                               max_depth=max_depth,
7                               max_features=max_features)
8  clf.fit(X,y)
9  ypred = clf.predict(X)
```

- User guide for hyperparameters : link
- $\Rightarrow$ Notebook
- the documentation

Boosting

Boosting

Schapire [1990]

Principle

- Weak learners, just better than random guess
- Focus to exemples hard to classify

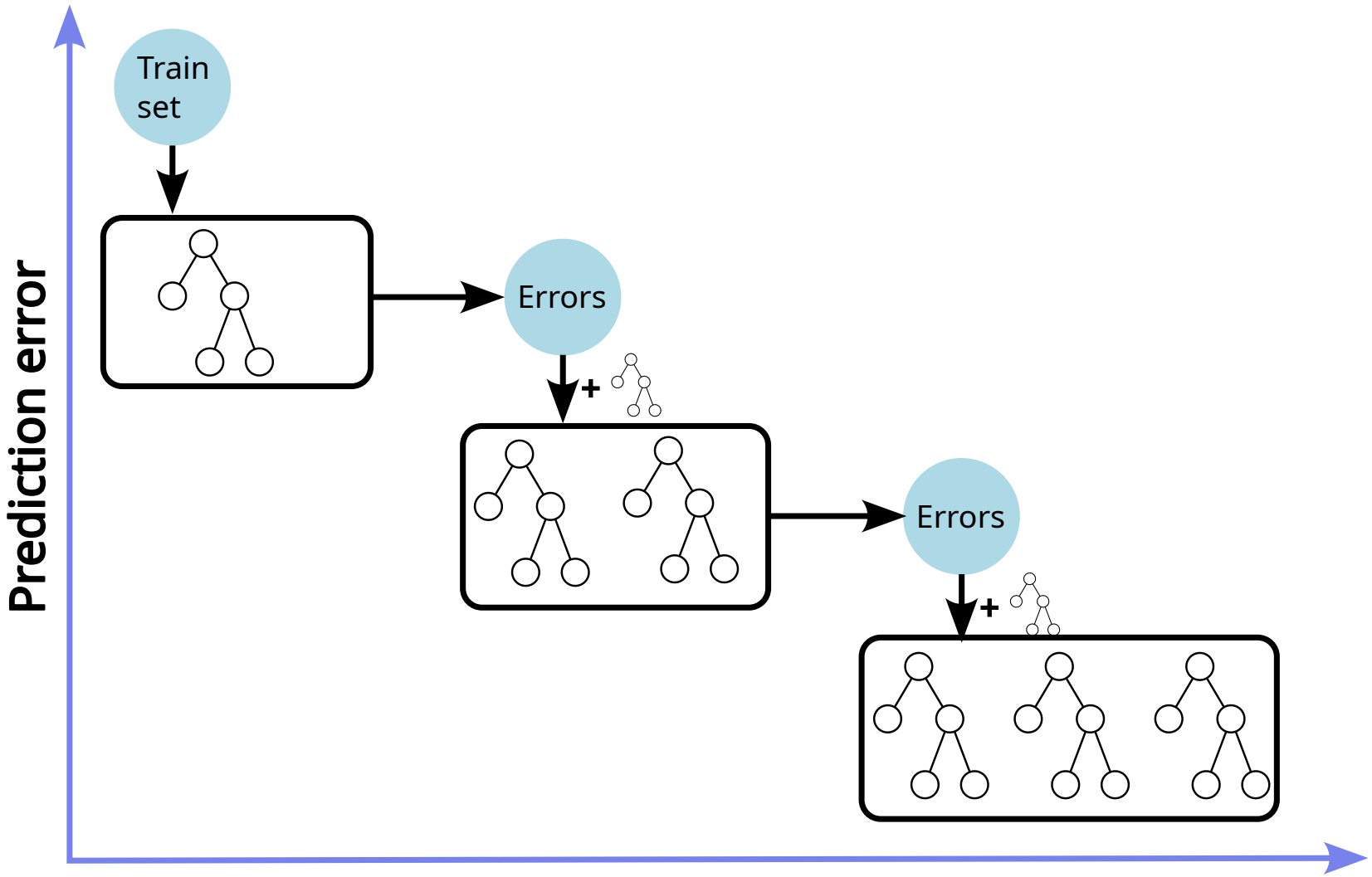
Boosting

1. Train a weak learner C_1 on a subset of training examples D_1
2. Train a second weak learner C_2 on a subset of training examples D_2 with 50% of misclassified data by C_1
3. Train a third weak learner C_3 on the data on which C_1 and C_2 disagree
4. Combine the weak learners C_1, C_2, and C_3 via majority voting.

- AdaBoost :weight misclassified examples betweeen rounds

Gradient Boosting

- Build a series of trees
- Each tree learns on the error of the previous ones
- The ensemble is improving by small steps
- Steps are computed according to a loss gradient



Gradient Boosting Hyperparameters

Number of trees

Same as before :

- low number : fast to compute, but less accurate
- high number : slower to compute, but more accurate up to some number

Tree depth

Specify the maximal depth of tree. An higher depth will make dedicate categories, but less generalizable.

Learning rate

Determine how much each weak learner contributes to the decision

- Regularization : small values produces a better test error
- help : https://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting-shrinkage

Implementations of Boosting I

GradientBoosting with sklearn

```
1        from sklearn.ensemble import GradientBoostingClassifier
2        n_estimators = 20 # the number of weak learners
3        learning_rate = .1
4        clf = GradientBoostingClassifier(n_estimators=n_estimators,
5        learning_rate=learning_rate)
6        clf.fit(X,y)
7        ypred = clf.predict(X)
```

Implementations of Boosting II

XgBoost

```
1        import xgboost as xgb
2        n_estimators = 20 # the number of weak learners
3        learning_rate = .1
4        clf = xgb.XGBClassifier(n_estimators=n_estimators,
5        learning_rate=learning_rate)
6        clf.fit(X,y)
7        ypred = clf.predict(X)
```

Conclusion

Decision trees and Co.

- Works (very) well on tabular data
- Interpretable
- State of the art on many challenges (Kaggles)
- Overfitting
- Need of a tabular representation of the data

References

References

Christopher M Bishop and Nasser M Nasrabadi. Pattern recognition and machine learning, volume 4. Springer, 2006.

Leo Breiman. Bagging predictors. Machine learning, 24:123–140, 1996.

Aurélien Géron. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. " O'Reilly Media, Inc.", 2022.

Sebastian Raschka, Yuxi Hayden Liu, Vahid Mirjalili, and Dmytro Dzhulgakov. Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python. Packt Publishing Ltd, 2022.

Robert E Schapire. The strength of weak learnability. Machine learning, 5:197–227, 1990.