Trees are fun

Paweł Gelar

Warsaw 11.2022

Contents

1. Introduction to decision trees
2. Building a tree (CART)
3. Decision trees properties
4. Ensemble learning
5. Boosting
6. Pseudo-case study

Introduction to decision trees

Tree as seen in nature

Figure: A tree

Tree as seen in data science

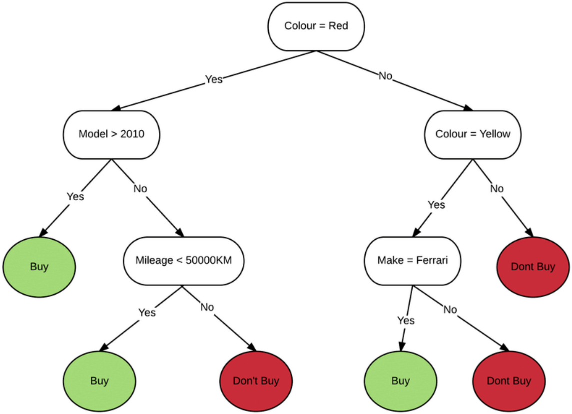


Figure: A normal tree

A little bit different perspective of the trees

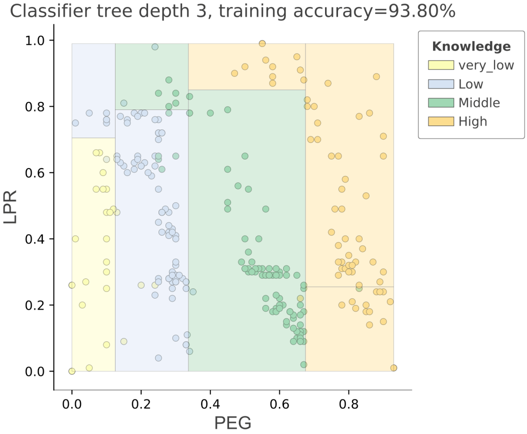


Figure: 2D-plane division of a tree

Building a tree (CART)

Classification and Decision Tree (CART)

What is CART?

CART is the algorithm used to build trees in most packages (Scikit-Learn in Python, rpart in R)

CART algorithm

1. Find the best split i.e. pair (k, t_k) of feature k and value t_k minimising $J(k, t_k)$ (1)
2. Repeat on every created subset until end conditions apply
3. Assign values to leaves

$$J(k, t_k) = \frac{m_{\mathit{left}}}{m} G_{\mathit{left}} + \frac{m_{\mathit{right}}}{m} G_{\mathit{right}} \tag{1}$$

where:
m - number (sum of weights) of all samples
$m_{\mathit{left/right}}$ - number (sum of weights) of samples in left/right subset
$G_{\mathit{left/right}}$ - value of the criterion function on left/right subset

Decision functions - classification

- Gini impurity

$$G = 1 - \sum_{i=1}^{k} p_i^2 \tag{2}$$

- Entropy

$$H = -\sum_{i=1}^{k} p_i log_2(p_i) \tag{3}$$

where:
k - number of classes
p_i - fraction of class i members in given node

Decision functions - regression

- Mean Squared Error (MSE)

$$MSE = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 \tag{4}$$

where:
m - number of points in given node
y_i - target value of i-th datapoint
$\hat{y}_i$ - predicted value of i-th datapoint

- Friedman MSE
Upgraded version of the MSE recommended in the Sklearn documentation

Decision trees properties

A quick puzzle

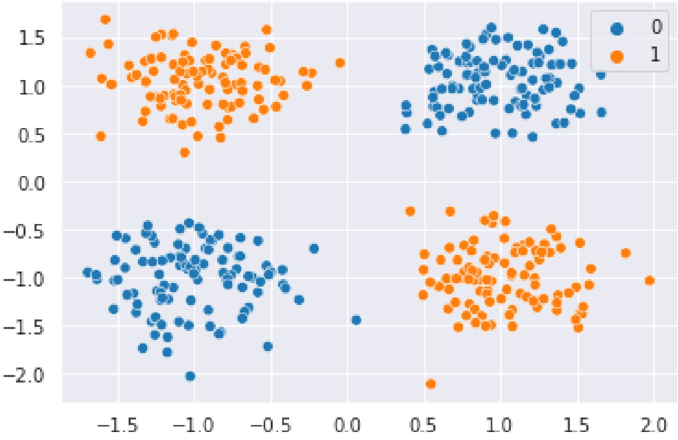


Figure: How's gonna look a decision tree build on this dataset?

Greediness

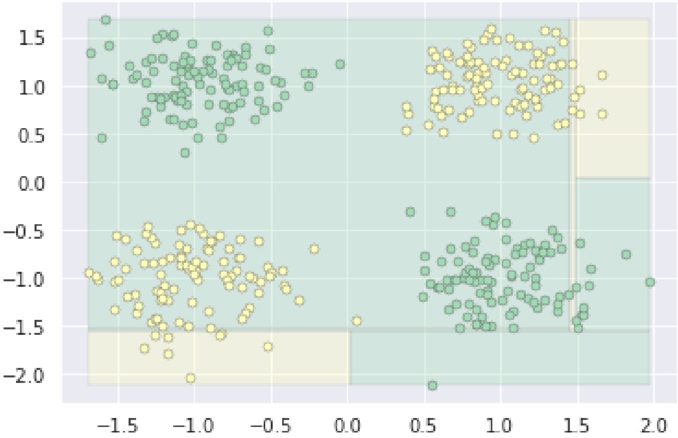


Figure: Total mess

Non-robustness

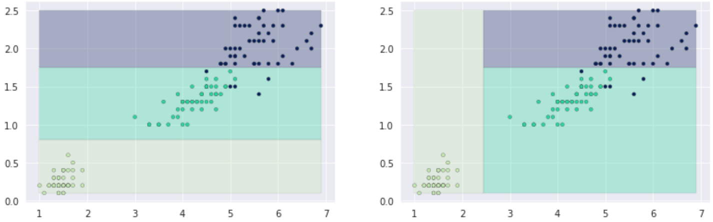


Figure: Two decision tree models trained on very similar data

Overfitting

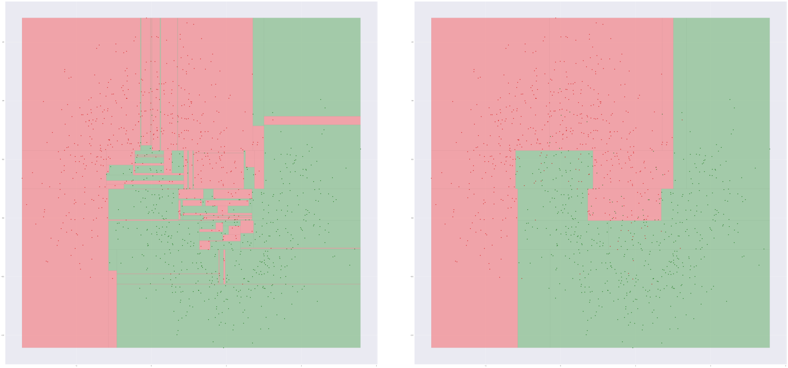
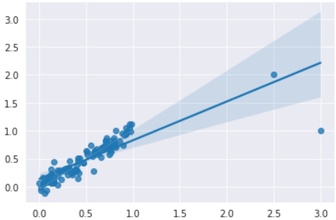
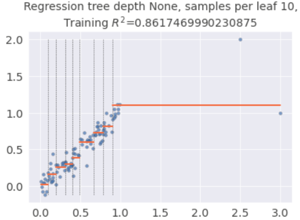


Figure: Overgrown and restricted tree

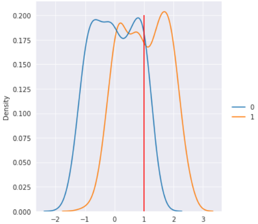
Outliers

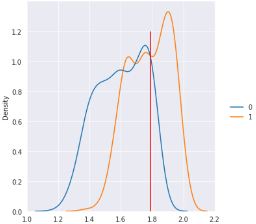




Trees are insensitive to outliers (unlike many models e.g. linear regression on the left)

Variables transformation





Single variable transformations have no effect on decision tree (if they preserve order of the datapoints

Rotating dataset

Figure: Decision tree before and after rotating the dataset

Regularising hyperparameters

- splitter
- max_depth
- min_samples_split
- min_samples_leaf
- max_features
- max_leaf_nodes
- class_weight - in classifier (not regularising)

Summary (decision tree)

Pros:

- Usually no data preprocessing is required
- Computationally lightweight (both learning and inference)
- "White box" models (explainable)
- Can estimate probability of belonging to class

Cons:

- Prone to overfitting
- Extremely non-robust
- Decision function is constant almost everywhere
- Greedy (doesn't find optimal solution)

Ensemble learning

Voting classifiers

1. Hard voting
   - Every voting classifier makes prediction. Class with the most single predictions is predicted
   - Can be used with any classifiers
2. Soft voting
   - Class with the highest average probability is predicted
   - Can be used only with classifiers returning probabilities

Making models a little independent

- Learning on different dataset:
  - Pasting
  - Bagging (bootstrap aggregating)
- Making models random
  - Random splitting (`splitter='random'`)
  - Random choosing subset of features
  - Even more randomness - extremely randomized trees (Extra-Trees)
- Smaller models

Boosting

AdaBoost (Adaptive Boosting)

Trees are build on weighted dataset.
Bigger weights are asigned to datapoints, on which the model performed poorly.

Figure: Component trees of boostting model

AdaBoost (Adaptive Boosting)

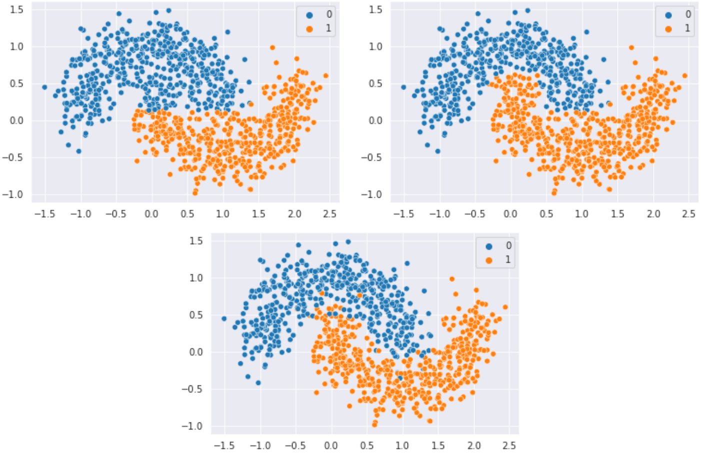


Figure: Model predictions (3/11/100) estimators

Gradient boosting

In histogram boosting estimators learn on residuals of previous models

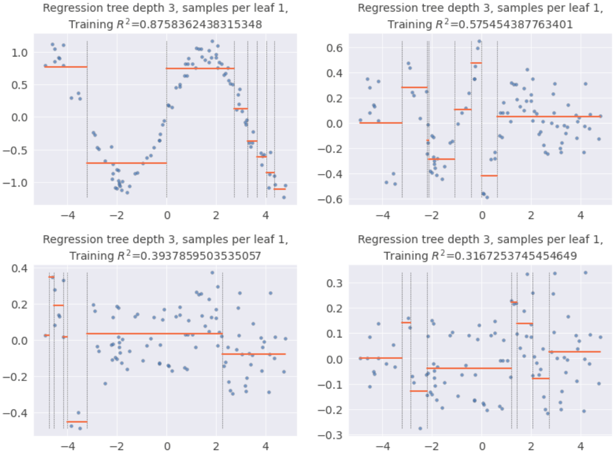


Figure: Estimators in gradient boosting

xgboost (eXtreme Gradient Boosting)

xgboost is heavily optimised library for gradient boosting available on many platforms (Python, R, Julia, Scala, C++ more)

Pseudo-case study

Data

1. tweet_length
2. hashtag_count
3. numbers_reference_count
4. time_reference_count
5. geopolitical_reference_count
6. has_link
7. has_emoji
8. 96D embedding of tweet text
9. 96D embedding of tweet keyword

Thanks for your attention