



Decision trees



Задачи:

1. Классификация
2. Регрессия

Решающее дерево - алгоритм машинного обучения, цель которого состоит в разбиении пространства признаков на рекурсивные области, таким образом, чтобы в каждой области преобладало одно значение целевой переменной.

Самый простой пример Дерева решений



По сути в данном случае мы решаем задачу *бинарной классификации*

Задача классификации. Энтропия как мера информативности



$$H(X) = - \sum_{i=1}^n p(x_i) \log_2 p(x_i)$$

H - энтропия Шеннона

n - число классов


Это будет аналог *функции потерь* и мы будем пытаться минимизировать это число.

Чем выше значение данной метрики, тем менее упорядочены наши данные и наоборот.

Если в наших данных классы сбалансированы, то изначальную энтропию можно рассчитать по формуле снизу и это число будет максимальным для данной выборки.

$$H(v) = - \sum_{i=1}^k \frac{1}{k} \log_2 \frac{1}{k} = k \cdot \frac{1}{k} \log_2 k = \log_2 k$$

Прирост информации (Information Gain)


$$IG = H(v) - \sum_{i=1}^k \frac{n_i}{n} H(v_i)$$

k - число узлов которые получились
после разбиения

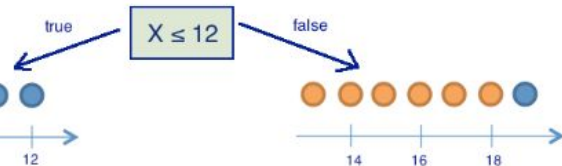
n - число элементов в материнском
узле

Данная формула помогает нам выбрать критерий для разбивки наших данных. То есть перебирая признаки объекта, мы выберем именно тот признак и его пороговое значение, при котором *прирост информации будет наибольший*.

Игрушечный пример

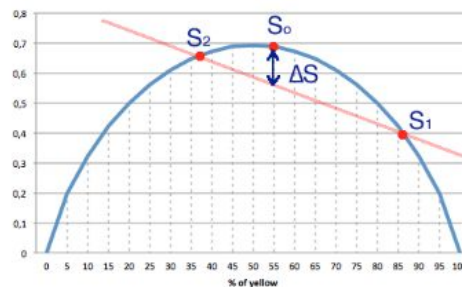


$$S_0 = -\left(\frac{9}{20}\right) * \text{Ln}\left(\frac{9}{20}\right) - \left(\frac{11}{20}\right) * \text{Ln}\left(\frac{11}{20}\right) \approx 0,69$$



$$S_2 = -\left(\frac{8}{13}\right) * \text{Ln}\left(\frac{8}{13}\right) - \left(\frac{5}{13}\right) * \text{Ln}\left(\frac{5}{13}\right) \approx 0,66$$

$$S_1 = -\left(\frac{6}{7}\right) * \text{Ln}\left(\frac{6}{7}\right) - \left(\frac{1}{7}\right) * \text{Ln}\left(\frac{1}{7}\right) \approx 0,4$$



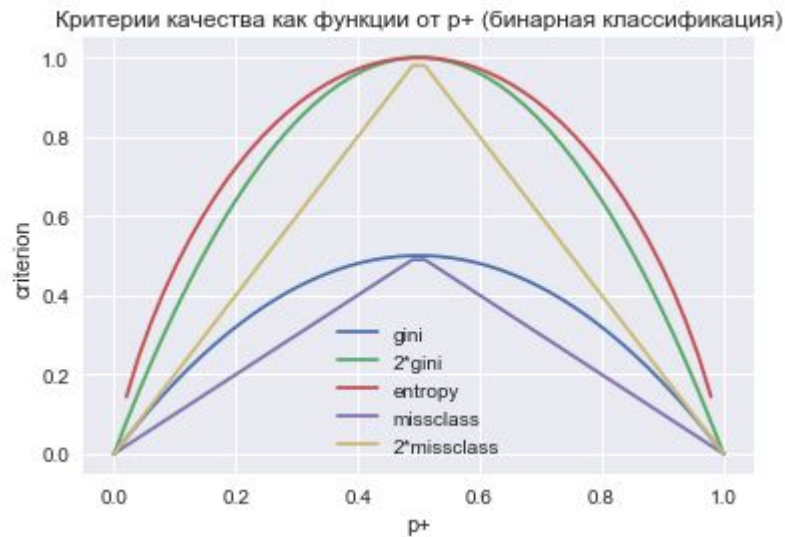
Алгоритм



1. s_0 = вычисляем энтропию исходного множества
2. Если $s_0 == 0$ значит:
 - a. Все объекты исходного набора, принадлежат к одному классу
 - b. Сохраняем этот класс в качестве листа дерева
3. Если $s_0 \neq 0$ значит:
 - a. Перебираем все элементы исходного множества:
 - b. Для каждого элемента перебираем все его атрибуты:
 - c. На основе каждого атрибута генерируем предикат, который разбивает исходное множество на два подмножества
 - d. Рассчитываем среднее значение энтропии. Вычисляем ΔS
 - e. Нас интересует предикат, с наибольшим значением ΔS
 - f. Найденный предикат является частью дерева принятия решений, сохраняем его
4. Разбиваем исходное множество на подмножества, согласно предикату
5. Повторяем данную процедуру рекурсивно для каждого подмножества

Другие методы измерения информативности

$$\text{Gini}(v) = \sum_{i=1}^k \left(p_i \sum_{j \neq i} p_j \right) = \sum_{i=1}^k p_i (1 - p_i) = \sum_{i=1}^k (p_i - p_i^2) = \sum_{i=1}^k p_i - \sum_{i=1}^k p_i^2 = 1 - \sum_{i=1}^k p_i^2$$



Методы измерения информативности для задачи регрессии



Для задачи регрессии мы будем минимизировать *среднеквадратичную ошибку*.

$$\text{MSE}(v) = \frac{1}{n} \sum_{(x_i, y_i) \in v} (y_i - \hat{y})^2$$

Или более простыми словами мы будем минимизировать *дисперсию* наших данных.

Предсказанным значением для объекта попавшего в определенную ноду, будет *среднее значение* всех элементов в этой ноде.

$$\hat{y} = \bar{y} = \frac{1}{n} \sum_{(x_i, y_i) \in v} y_i$$

Критерии остановки



1. `max_depth` - ограничение максимальной глубины деревьев
2. `min_samples_leaf` - ограничение минимального числа объектов в каждом листе(если при делении один из листьев имеет меньшее количество объектов деления не происходит)
3. `min_samples_split` - ограничение на количество объектов в ноде для дальнейшего разбиения
4. `max_leaf_nodes` - ограничение на максимальное количество листьев
5. `min_impurity_decrease` - нода будет разделяться только тогда, когда значения уменьшения информативности больше или равно данному значению

Pruning



1. Pre-pruning - для регуляризации используем перечисленные на предыдущем слайде гиперпараметры. Идея заключается в том чтобы не дать дереву разрастись и переобучиться.
2. Post-pruning - даём дереву возможность дойти до самого конца, а дальше убираем его отдельные ноды.



Как подготовить данные?

1. Imputation
2. OneHotEncoder, OrdinalEncoder, custom и т.д.



Преимущества:

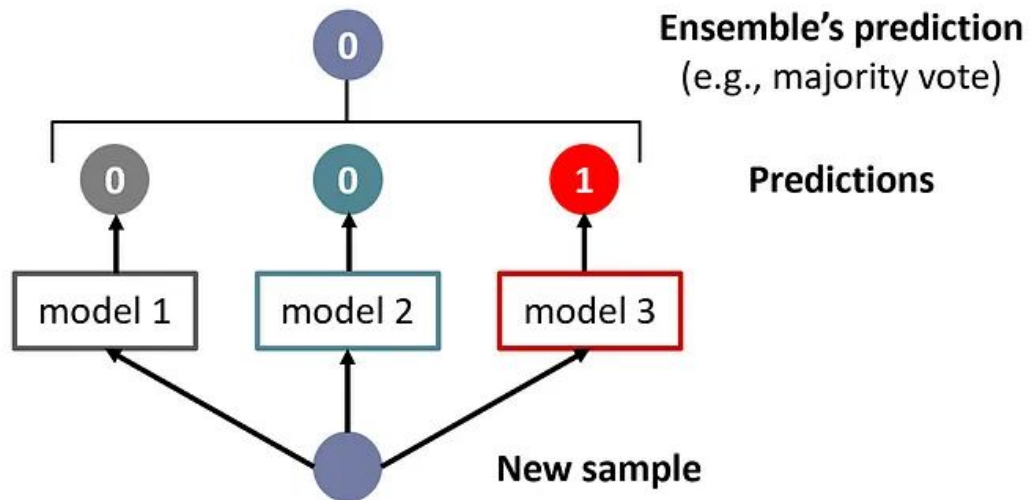
1. Можно моделировать не только линейные зависимости
2. Минимальная подготовка данных
3. Модель можно хорошо визуализировать и объяснить
4. Могут быть полезными на этапе `feature_selection`, признаки которые будут вверху дерева наиболее информативные
5. Служат основой для очень хороших алгоритмов



Недостатки:

1. Переобучение
2. Очень сложно найти оптимальное дерево
3. Нестабильные, небольшое изменение в тренировочной выборке могут привести к большим последствиям

Ансамбли



Как мы будем создавать ансамбли?

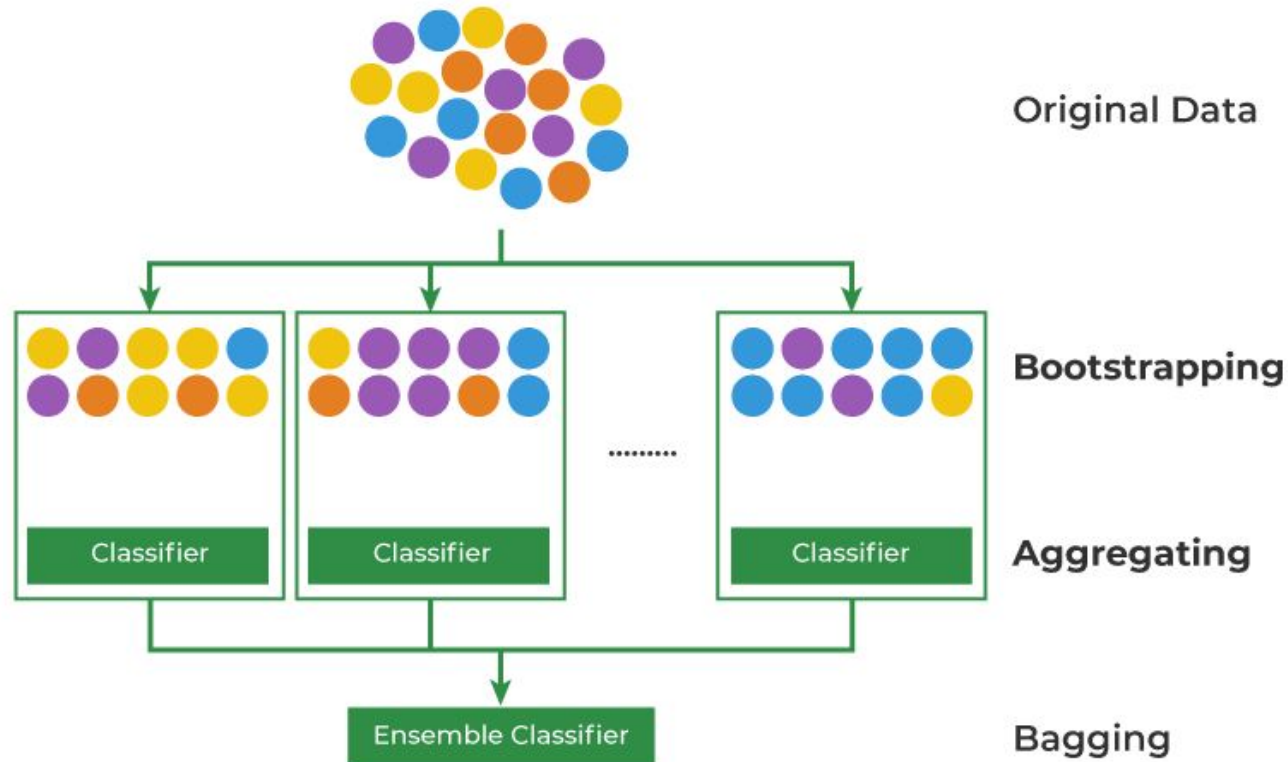


1. Мы можем использовать различные алгоритмы для набора наших базовых моделей
2. Обучать базовые модели на разных частях тренировочной выборки
3. Использовать разный набор фичей для разных базовых моделей

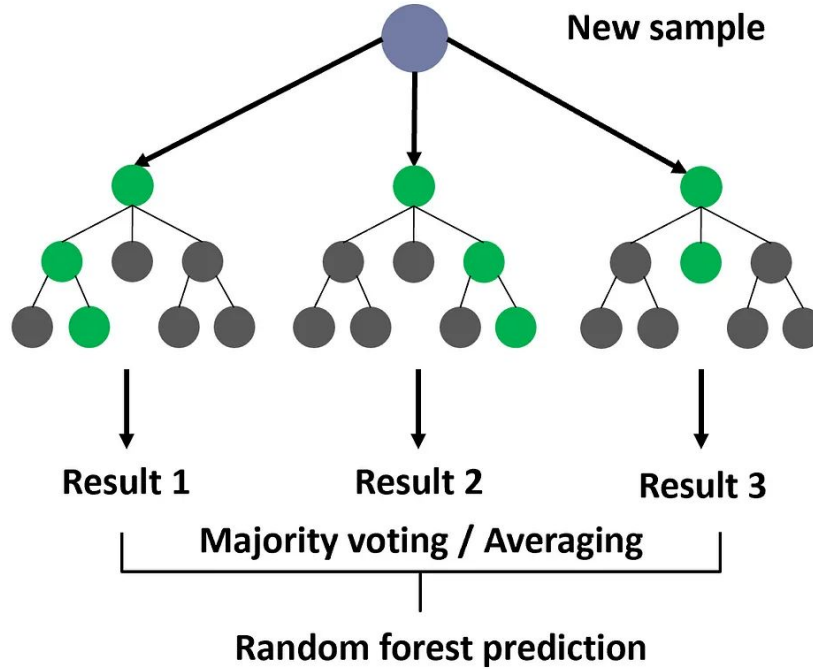
Алгоритмы ансамблирования:

1. Bagging
2. Boosting
3. Stacking

Bagging



Random forest

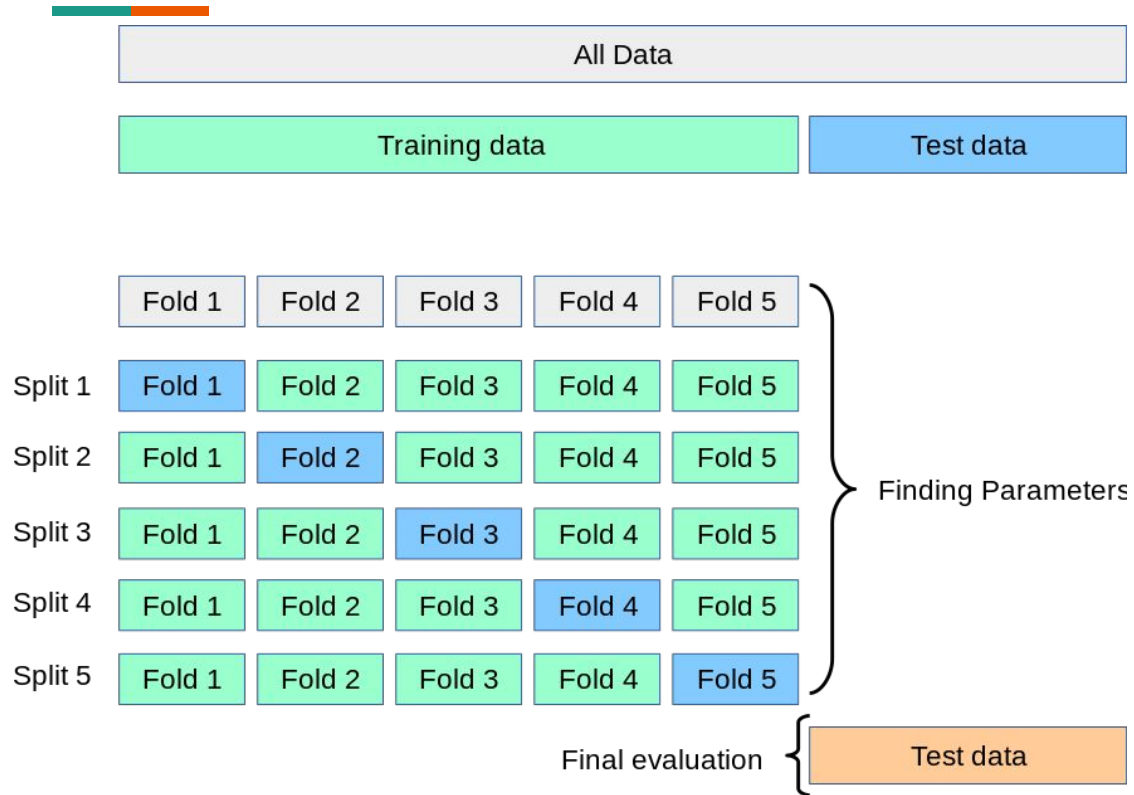




Model Validation

1. Train - Test
2. Train - Valid - Test
3. Cross Validation

K - Fold



Отдыхайте

