KARPOV.COURSES >>>
КОНСПЕКТ

# Конспект > 19 урок >Композиции алгоритмов. Случайный лес



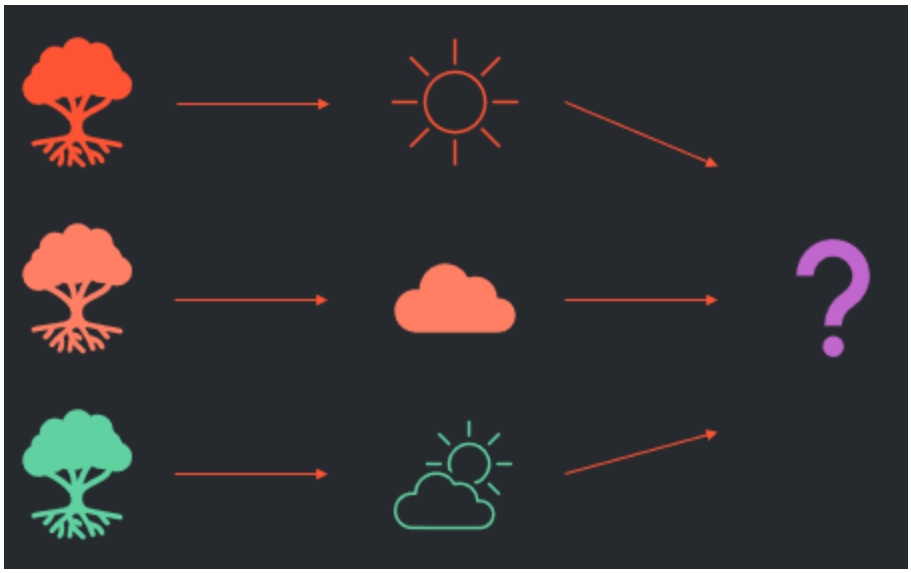
## >Композиции алгоритмов

Мы уже рассмотрели несколько моделей машинного обучения: метод ближайших соседей, линейные алгоритмы и решающее дерево. Все эти алгоритмы имеют свои преимущества и недостатки: у **метода ближайших соседей** недостаточная обобщающая способность, но он прост в понимании. **Линейный алгоритм** не имеет достаточное количество степеней свободы для сложных зависимостей, но стабилен к выбросам. **Решающее дерево** обладает значительной вариативностью, но склонно к переобучению и требовательно к размеру обучающей выборки. А что если, использовать несколько алгоритмов машинного обучения одновременно? Чтобы вместе они образовывали более сильную модель, как-бы, компенсируя недостатки друг друга.

Построение **композиции алгоритмов (**ансамбль моделей, Ensemble of models**) -** является важной концепцией в машинном обучении. Идея композиции проста: при

решении задачи, каждый, даже очень качественный, алгоритм может ошибаться. Но если, разные виды моделей могут "изучить" разные детали одной и той же задачи, то скомбинировав результаты прогнозов различных моделей, можно получить более точный результат.

Применение нескольких алгоритмов для повышения точности результата можно пронаблюдать на примере теоремы Кондорсе о жюри присяжных. Если каждый алгоритм угадывает ответ на конкретном объекте более чем в 50% случаев и количество алгоритмов стремится к бесконечности, то с вероятностью 1, большинство из них окажутся правы. Иллюстрацией принципа Кондорсе является эксперимент Фрэнсиса Кальтона и такой метод принятия решений называется методом простого голосования.

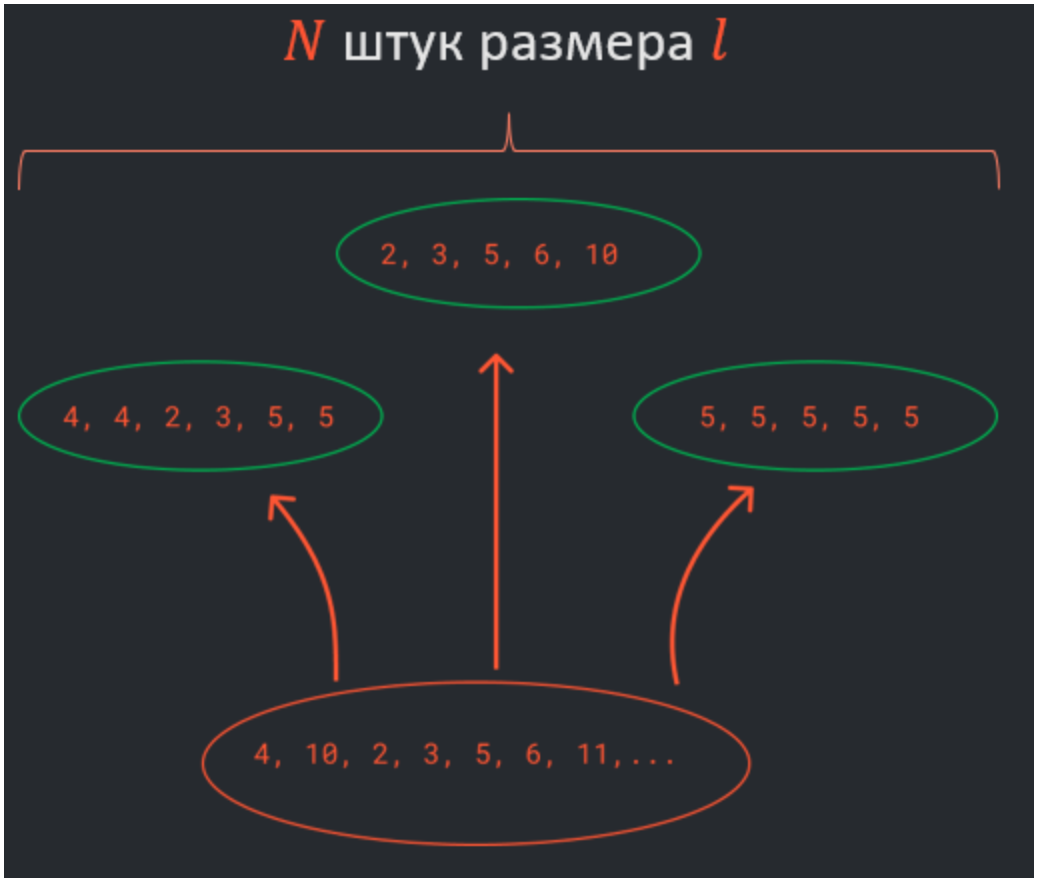


# >Бэггинг

**Бэггинг** (bootstrap aggregation) - принцип построения композиции, построенный на простом голосовании. Обычно, в случае бэггинга базовые алгоритмы используются из одного семейства. Пусть мы имеем **N** базовых алгоритмов: $b_1(x), b_2(x), ..., b_n(x)$. Тогда, в случае регрессии финальным ответом будет среднее по всем алгоритмам, а в случае классификации решение через простое голосование: $a(x) = \frac{a}{N} \sum_N^i b_i(x)$. Описанный алгоритм, если в качестве базовых алгоритмов выступают решающие деревья, ещё называют случайным лесом.

Базовые алгоритмы могут быть похожи, ведь если на одной и той же выборке каждый раз строить дерево с одинаковыми гиперпараметрами, тогда в результате получится одинаковая модель. И строить композицию из одинаковых моделей, а потом усреднять по ним прогноз бессмысленно. Значит надо как-то изменить модели, например, добавить в процесс построения выборки случайность.

Называется этот подход **bootstrap.** Пусть у нас имеется выборка $X = \{(x_i, y_i)\}_i^n$. Возьмем случайным образом элемент этой выборки l раз с возвратом (возвращаем выбранные элементы в выборку), получим псевдовыборку. Повторим процедуру N раз и получим N подвыборок $X_1, \ldots, X_N$ размера l.




Если можно рандомным образом сгенерировать N подвыборок, значит можно обучить N различных моделей.

Пусть у нас есть большой датасет, с помощью процедуры бутстрапа можно разбить этот датасет на 3 подвыборки, т.е сгенерировать 3 экземпляра, которые будут приблежать нашу изначальную выборку. Затем, обучим решающее дерево

на полученных подвыборках и получим 3 разных модели. Финальная модель, бэггинг ансамбль — среднее значение от построенных моделей.


Представим, мы обучили N базовых алгоритмов: $b_1(x), b_2(x), ..., b_n(x)$. Ошибку каждого алгоритма в случае MSE на объекте x можно записать как $e_i^2 = (b_i(x) - y(x))^2$.

А среднее её значение, то есть ожидаемое: $E_x(b_i(x) - y(x))^2 = E_x e_x^2$.

Если для каждого i-того базого алгоритма можно усредниться по его ошибкам, то также можно усредниться от среднего, т.е посчитать какие в среднем средние ошибки имеют базовые алгоритмы. Обозначим за величину $E_1$ среднюю ожидаемую ошибку по всем обученным алгоритмам: $E_1 = \frac{1}{N} * \sum_i^N E_x e_i^2$.

Построим бэггинг-алгоритм $a(x) = \frac{1}{N} * \sum_i^N b_i(x)$ и замерим его ожидаемую ошибку $E_N$:

$E = \frac{1}{N^2} * E_x(\sum_i^N e_j^2(x) + \sum_i e_i(x)e_j(x))$.

Допустим, ошибки базовых алгоритмов нескоррелированы, тогда $\sum_i e_i(x)e_j(x) = 0$. Если это слагаемое равно нулю, то его можно опустить:

$E_N = \frac{1}{N^2} * E_x \sum_i^N e_j^2 = \frac{1}{N} * \frac{1}{N} * \sum_i^N E_x e_j^2 = \frac{1}{N} * E_1$

Получается, ошибка бэггинг-алгоритма уменьшится в N раз!

Можно сделать вывод: если построить бэггинг на нескоррелированных базовых алгоритмах, то средняя ошибка итогового алгоритма уменьшиться в N раз (в сравнении со средней ошибкой базовых алгоритмов).

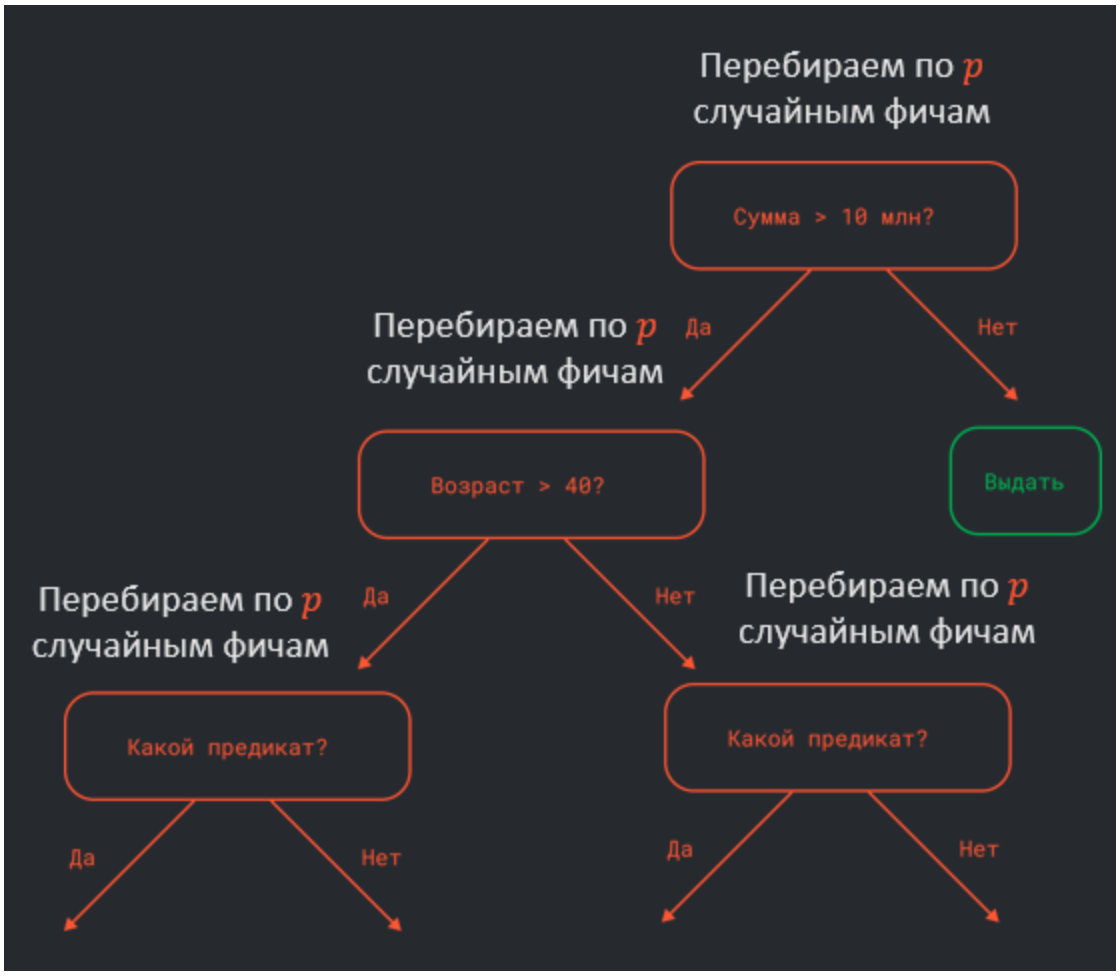
# >Random Forest

Случайный лес - один из популярнейших способов применить бэггинг.

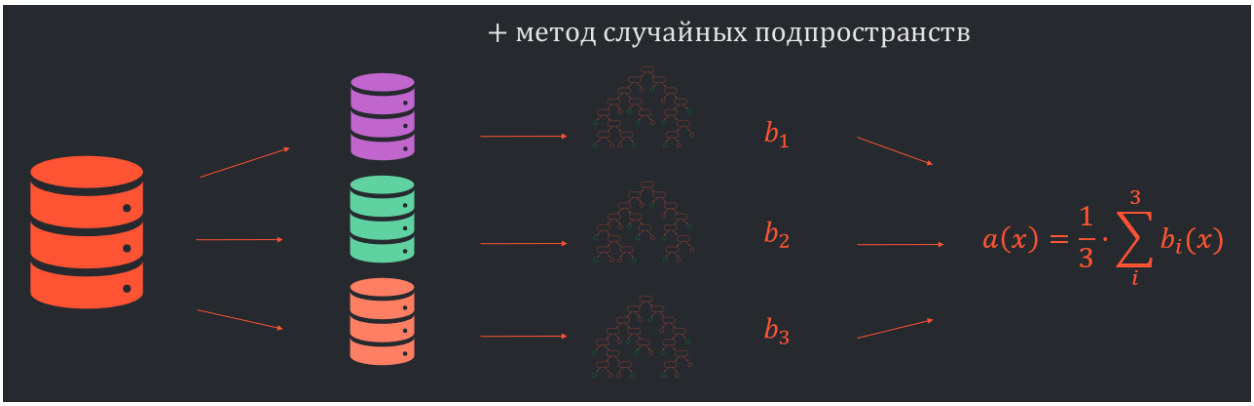
Суть заключается в том, чтобы в качестве базовых моделей $b_1(x), b_2(x), \ldots, b_N(x)$ взять решающие деревья. И желательно, чтобы базовые модели были максимально некоррелированные, чтобы прогнозы и ошибки были разные. Но как этого достичь?

- Во-первых, рандомизация достигается за счет бутстрапа
- Во-вторых, включается метод случайных подпространств: при построении решающих деревьев выберется лучший предикат не среди всех $m$ признаков, а из случайно выбранных $p < m$. Данная технология позволяет получать различные друг от друга решающие деревья и это добавляет рандомизации в построение базовых алгоритмов.

При решении задачи регрессии/классификации число выбираемых признаков должно быть равно примерно $\frac{m}{3}$ или $\sqrt{m}$ (округляя вниз).



Пусть у нас есть большой датасет, с помощью процедуры бутстрапа можно разбить этот датасет на 3 подвыборки, т.е сгенерировать 3 экземпляра, которые будут приближать нашу изначальную выборку. Затем, обучим **глубокое решающее дерево** на полученных подвыборках и получим 3 разных модели $b_1(x), b_2(x), \ldots, b_N(x)$. Финальная модель, бэггинг ансамбль — среднее значение от построенных моделей



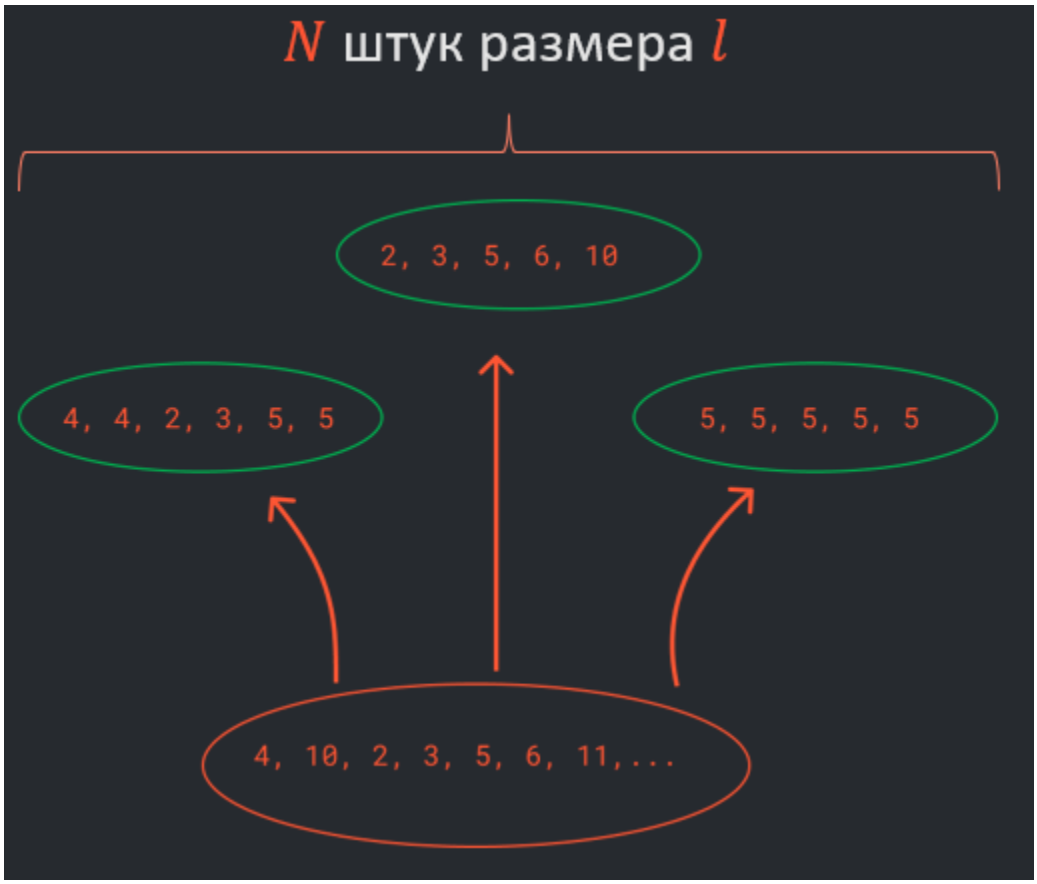

# >Out-of-bag ошибка

Каждый раз, при построении ансамбля, который генерирует подвыборки (с помощью бутстрап) и на них обучается, можно посчитать ошибку особого вида. Строя случайный лес не обязательно делить выборку на трейн и тест чтобы понять обобщающую способность алгоритма.

При использовании случайного леса нет необходимости в кросс-валидации, чтобы получить несмещенную оценку ошибки набора тестов

Бутстрапируя выборку, для отдельных алгоритмов часть объектов выкидываются. Тогда для каждого объекта можно замерить ошибку по тем деревьям, которые на нем не обучались, и получить out-of-bag ошибку:

$$OOB = \sum_i^n L(y_i, \frac{1}{\sum_j^N [x_i \notin X_j]} \sum_j^N [x_i \notin X_j] \cdot b_j(x_i))$$




Т.е элементы, которые не вошли в бутстрапированную подвыборку могут являться тестовыми (для алгоритма обученного на бутстрапированной подвыборке)

## >Стекинг

Недостаток случайного леса это слишком долгое и дорогое вычисление. Чтобы его вычислить нужно спросить все базовые алгоритмы, а затем усреднить предсказания.

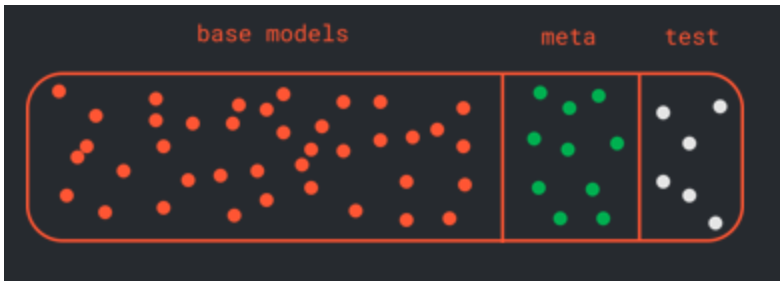
Что если, мы не хотим обучать тысячи обучающих деревьев, а хотим обойтись небольшим количеством алгоритмов. Пусть у нас есть N базовых алгоритмов $b_1(x), b_2(x), \ldots, b_N(x)$. Эти модели уже обучены и могут давать какие-то прогнозы. И для каждого объекта x можно получить $N$ различных прогнозов, отправив его в каждую обученную базовую модель. Теперь можно использовать эти прогнозы как новое признаковое пространство и скормить их мета-алгоритму, который обучается на выходах базовых моделей. Т.е использовать прогнозы N базовых алгоритмов как новые фичи и скормить их в финальную модель.

Пусть у нас есть большой датасет, с помощью процедуры бутстрапа можно разбить этот датасет на 3 подвыборки, т.е сгенерировать 3 экземпляра, которые будут приближать нашу изначальную выборку. Затем, обучим **3 различных моделей** на полученных подвыборках, а пронозы моделей будем использовать как новое признаковое описание объектов. И уже на полученных фичах обучим финальную модель.


Проблема классического стекинга заключается в склонности к переобучению. Например, среди базовых алгоритмов затесалось глубокое переобученное дерево, тогда мета-алгоритм подстроится под такую базовую модель и тоже переобучится! Например $a(x) = 1 \cdot b_1(x) + 0 \cdot b_2(x) + 0 \cdot b_3(x)$, где $b_1(x)$ - переобученный базовый алгоритм!

## >Борьба с переобучением

Бороться с переобучением в стекинге можно с помощью отложенной выборки. Обучать базовые модели будем не на всем трейне $Xtrain$, а на отложенном кусочке $Xtrain^k$ Метамодель будем обучать на оставшейся части, то есть на $Xtrain/Xtrain^k$



Или более сложный вариант-модификация: Разобьем обучающую выборку на K частей, каждый из $N$ базовых алгоритмов $b_1(x), b_2(x), \ldots, b_N(x)$ обучим K раз на разной отложенной выборке. Финальную мета-модель будем строить, минимизируя сумму ошибок по каждому фолду.