Занятие № 9

Ансамблирование моделей

Содержание

«Мудрость толпы»

Фрэнсис Гальтон «Мудрость толпы»

Фрэнсис Гальтон в 1906 году посетил рынок, где проводилась некая лотерея для крестьян.
Их собралось около 800 человек, и они пытались угадать вес быка, который стоял перед ними. Бык весил 1198 фунтов. Ни один крестьянин не угадал точный вес быка, но если посчитать среднее от их предсказаний, то получим 1197 фунтов.
Эту идею уменьшения ошибки применили и в машинном обучении.

Теорема Кондорсе «о жюри присяжных» (1784).

Если каждый член жюри присяжных имеет независимое мнение, и если вероятность правильного решения члена жюри больше 0.5, то тогда вероятность правильного решения присяжных в целом возрастает с увеличением количества членов жюри и стремится к единице. Если же вероятность быть правым у каждого из членов жюри меньше 0.5, то вероятность принятия правильного решения присяжными в целом монотонно уменьшается и стремится к нулю с увеличением количества присяжных.

N — количество присяжных

p — вероятность правильного решения присяжного

μ— вероятность правильного решения всего жюри

m — минимальное большинство членов жюри, $m = \text{floor}(N/2) + 1$

C_N^i - число сочетаний из N по i

$$\mu = \sum_{i=m}^{N} C_N^i p^i (1-p)^{N-i}$$

Если $p > 0.5$, то $\mu > p$

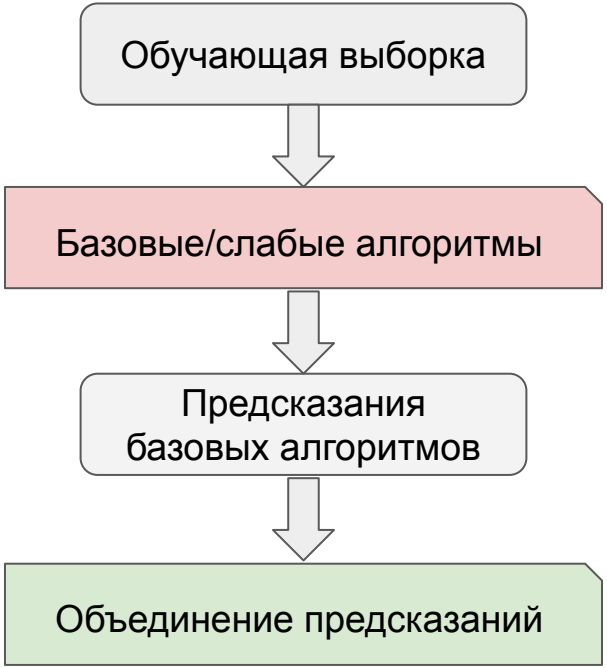
Если $N \to \infty$, то $\mu \to 1$

Ансамбли алогритмов

Основная идея:

1) используем множество алгоритмов
Называет базовыми/слабыми)

2) объединим их ответы
Различные стратегии объединения:
 а) голосование или усреднение (с весами / без весов)
 б) мета-алгоритм



Проблемы реальных задач

Алгоритмы не независимы, потому что:

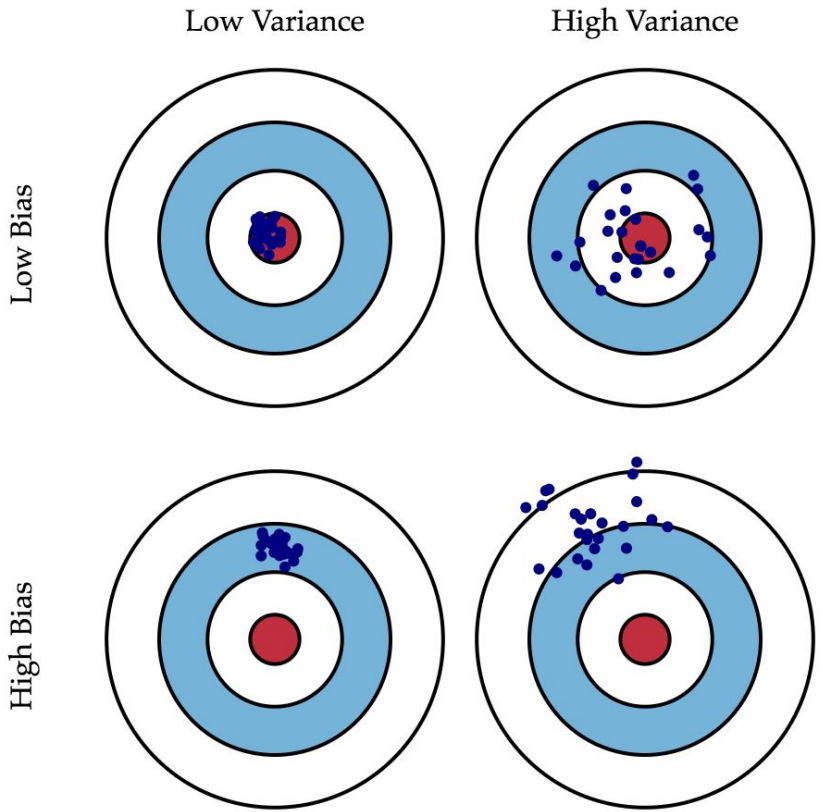
1. Решают одну задачу
2. Обучаются для предсказания одной и той же целевой переменной
3. Могут строиться на основе моделей одного типа

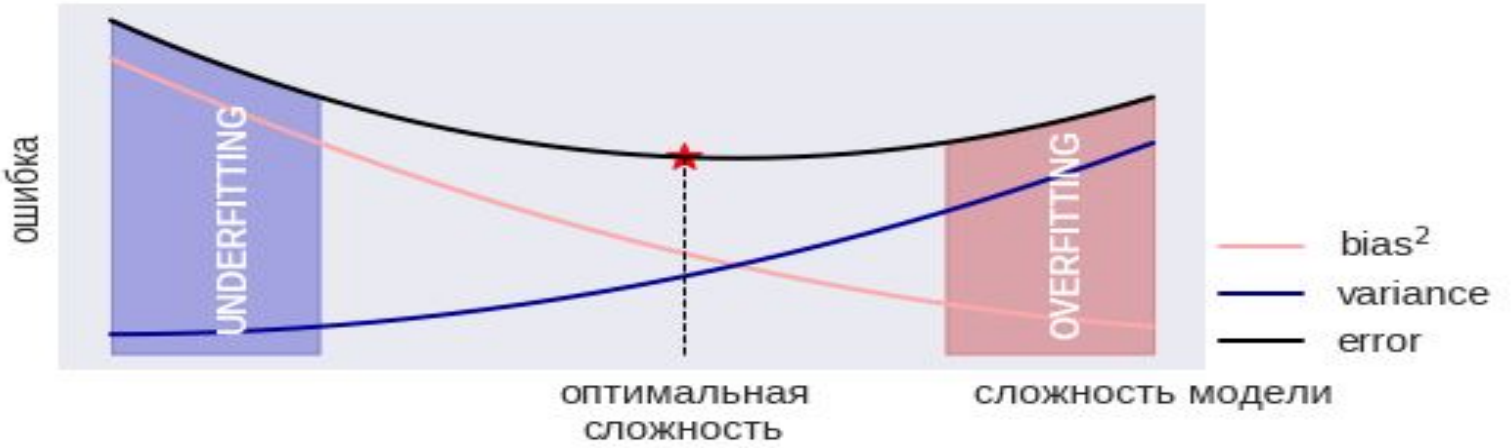
При построении ансамбля важно добиться независимости составляющих базовых алгоритмов.

Основные подходы:
1) Изменять обучающую выборку (случайные подпространства признаков и семплов)
2) Брать алгоритмы разных типов

разброс (variance) - дисперсию ответов алгоритмов характеризует разнообразие алгоритмов (из-за случайности обучающей выборки, в том числе шума, и стохастической природы настройки)

смещением (bias) – матожидание разности между истинным ответом и выданным алгоритмом, характеризует способность модели алгоритмов настраиваться на целевую зависимость.





Для простых моделей характерно недообучение (они слишком простые, не могут описать целевую зависимость и имеют большое смещение), для сложных – переобучение (алгоритмов в модели слишком много, при настройке мы выбираем ту, которая хорошо описывает обучающую выборку, но из-за сильного разброса она может допускать большую ошибку на тесте).

Усреднение

1. Простое усреднение (для регрессии), голосование (для классификации)

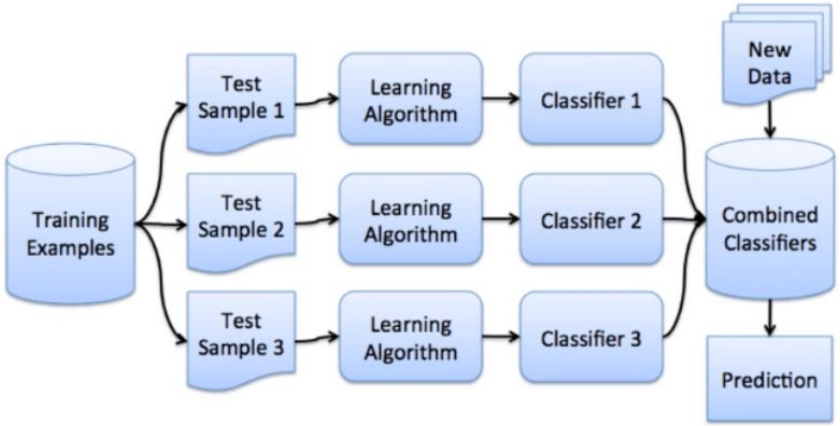
2. Усреднение с весами.
   а) веса подбираются исходя из доверия алгоритму (например с помощью простого алгоритма типа линейной регрессии) (Важна регуляризция)
   б) веса подбираются с для каждого семпла с помощью алгоритма

Бегинг (Bagging)

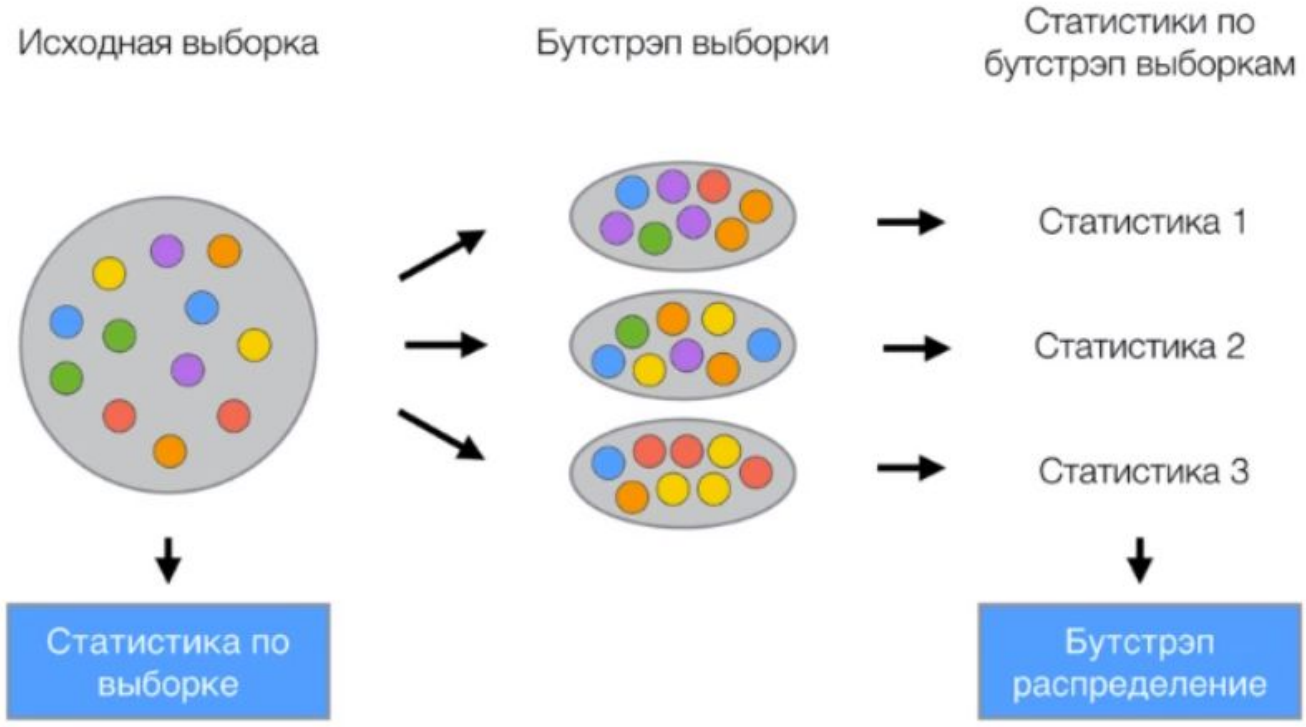
Bagging - bootstrap aggregating

Используют однородные базовые алгоритмы и обучают параллельно и на случайном подмножестве обучающей выборки.

При объединении результатов мы складываем случайные величины.
По этому для получения несмещенного результата нам нужны модели с малым смещением.
Большой разброс при объединении уменьшиться в соответствии с ЦПТ.



Бутстреп

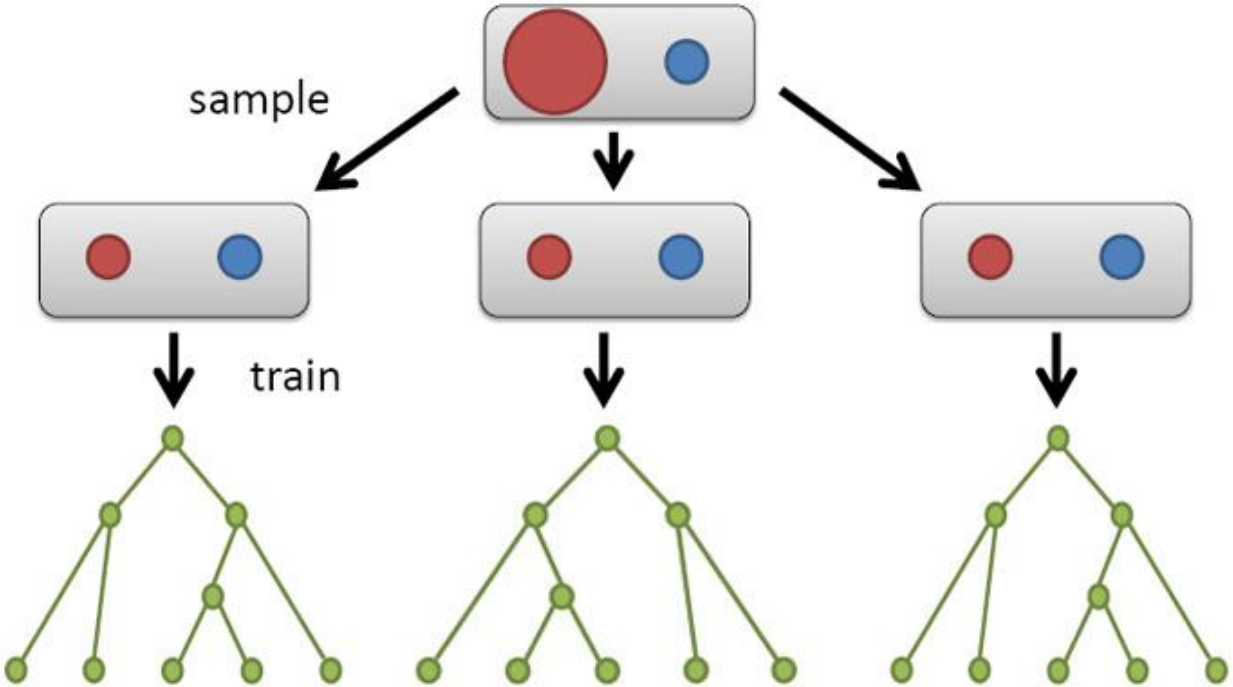

Исходная выборка
Бутстрэп выборки
Статистики по бутстрэп выборкам
Статистика 1
Статистика 2
Статистика 3
Статистика по выборке
Бутстрэп распределение

Бэгинг

from sklearn.ensemble import BaggingRegressor

base_estimator object or None – модель регрессии из sklearn (по умолчанию деревья решений)
n_estimators – количество моделей
max_samples int or float, optional (default=1.0) Количество сэмплов для обучения
max_features int or float, optional (default=1.0) Количество признаков для обучения

Случайный лес



Случайный лес

Бэггинг + случайные подпространства = случайный лес

Случайный лес - вариация беггинга над деревьями, дающая даже лучшие результаты: Как и в беггинге, мы создаем ансамбль деревьев решений, используя выборки из обучающего набора. Однако при построении каждого дерева каждый раз, когда производится расщепление, признак выбирается из случайной выборки размера m из всех признаков. Новая случайная выборка признаков формируется для каждого отдельного дерева в каждом отдельном расщеплении. Для классификации m обычно выбирается как квадратный корень из p. Для регрессии m обычно выбирается где-то между $p/3$ и p.

Случайный лес

Случайные подпространства

Предположим, что в наборе данных есть один очень сильный признак. При использовании беггинга большая часть деревьев будет использовать этот признак в качестве первого, по которому производится деление, в результате чего образуется ансамбль похожих деревьев, которые сильно коррелированы. Усреднение высококоррелированных величин не приводит к значительному уменьшению дисперсии (что является целью беггинга). Случайно исключая признаки из каждого расщепления, Random Forest «декоррелирует» деревья, так что процесс усреднения может уменьшить дисперсию результирующей модели.

Случайный лес

from sklearn.ensemble import DecisionTreeRegressor

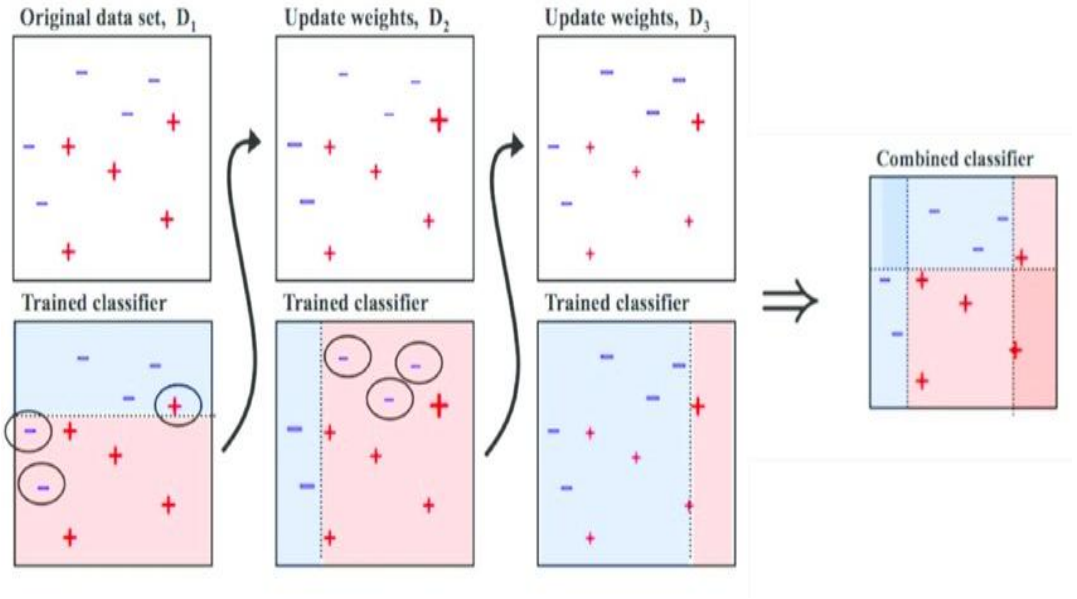
criterion – метод оценки ошибки (MSE по умолчанию)
splitter str – метод разделения ”better” или “random”
n_estimators – количество моделей
max_deph – максимальная глубина деревьев
max_samples int or float, optional (default=1.0) Количество сэмплов для обучения
max_features int or float, optional (default=1.0) Количество признаков для обучения

Бустинг

Бустинг (англ. boosting — улучшение) — это процедура последовательного построения композиции однородных базовых алгоритмов, когда каждый следующий алгоритм стремится компенсировать недостатки композиции всех предыдущих алгоритмов.

Обучение происходит последовательно

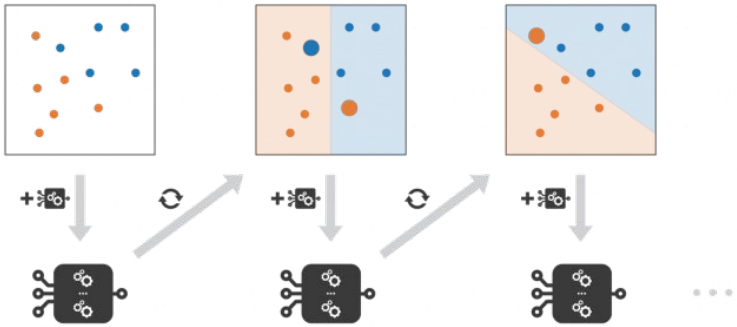
Так как очередной алгоритм исправляет предыдущий, то нам не подходят алгоритмы с большим разбросом.



Бустинг

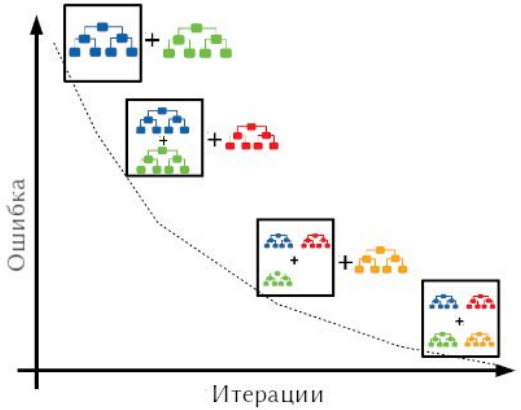
AdaBoosting

Для каждого следующего алгоритма увеличивается вес ошибки для неверно полученых результатов



Градиентный бустинг

Сводит задачу к градиентному спуску и на каждой итерации подгоняет очередной базовый алгоритм в соответствии с антиградиентом ошибки текущей модели ансамбля



Стекинг

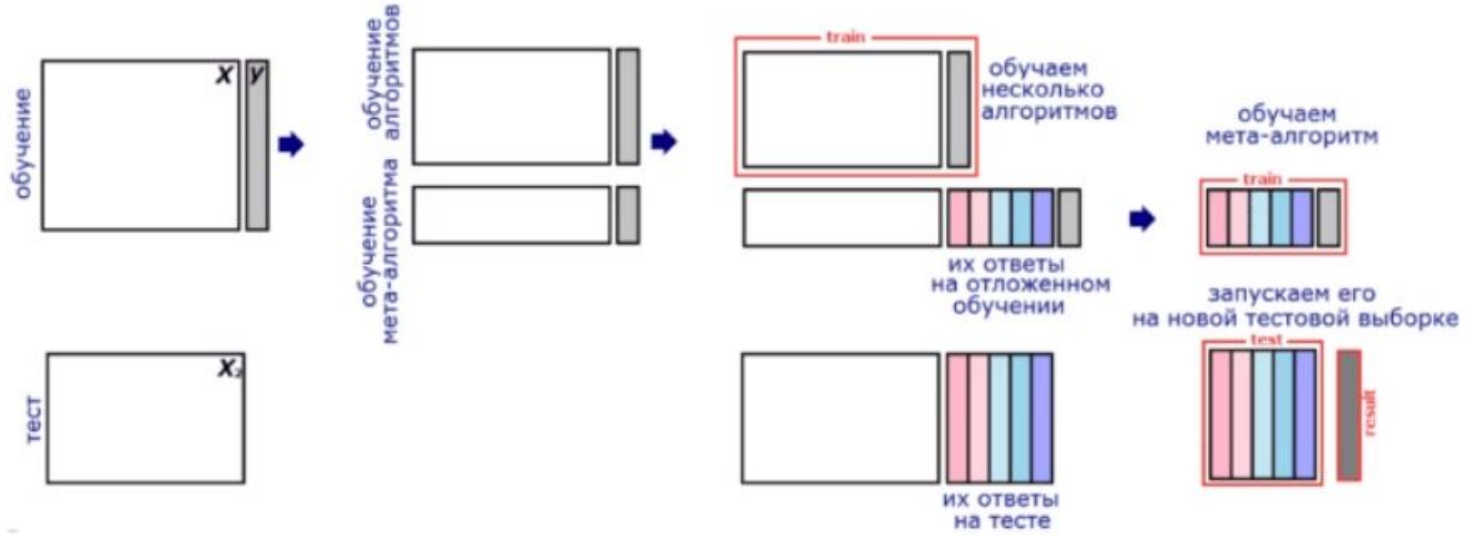
Используют разнородные базовые алгоритмы, обучают их параллельно. Для объединения результатов используется так называемая метамодель (или модель второго уровня) для предсказания конечного результата.

Можно строить не только два уровня, но более уровней стекинга.

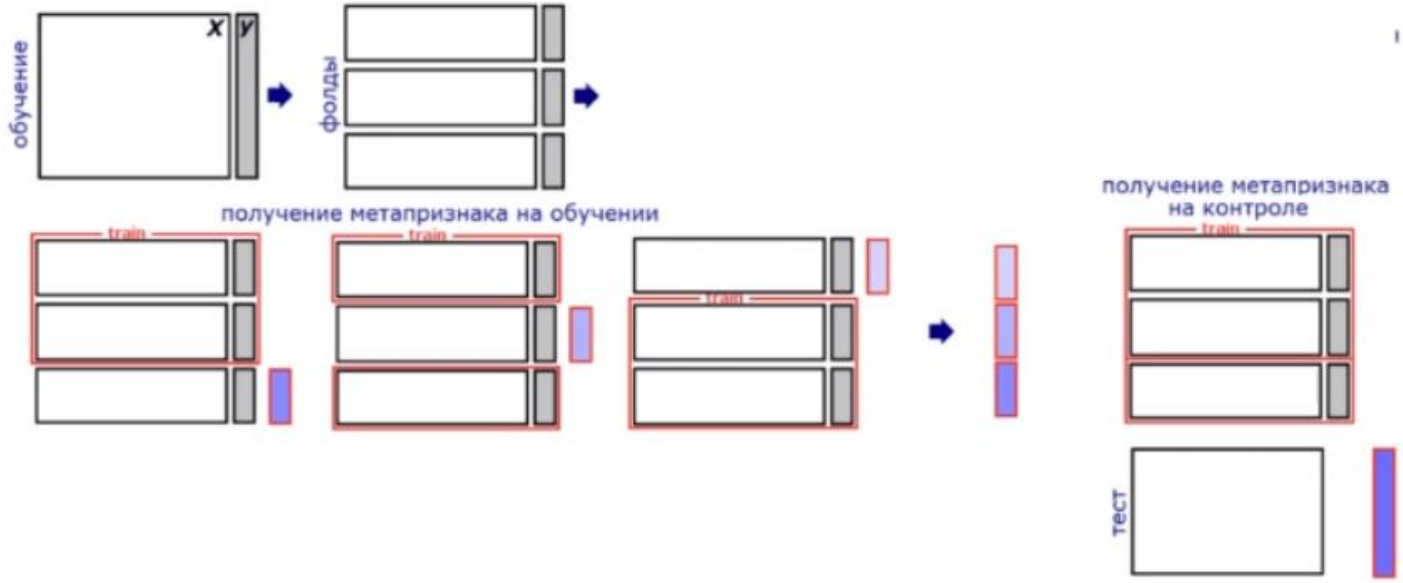
Два основных подхода к обучению стекинга

1) Блендинг
2) Стекинг

Блендинг


обучение
X
y
тест
X₂
обучение алгоритмов
обучение мета-алгоритма
train
обучаем несколько алгоритмов
их ответы на отложенном обучении
их ответы на тесте
обучаем мета-алгоритм
train
запускаем его на новой тестовой выборке
test
result

Стекинг


обучение
X
y
фолды
получение метапризнака на обучении
train
train
train
получение метапризнака
на контроле
train
тест

Спасибо за внимание!

Рекомендации для ознакомления:
https://dyakonov.org/2019/04/19/%D0%B0%D0%BD%D1%81%D0%B0%D0%BC%D0%B1%D0%BB%D0%B8-%D0%B2-%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%BC-%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B8/

https://neurohive.io/ru/osnovy-data-science/ansamblevye-metody-begging-busting-i-steking/