

Data Mining In Action

Ансамбли моделей

Александр Самойлов
@asamoylov
s3ng@yandex.ru



О чем поговорим

- Смещение и разброс (bias / variance)
- Композиция алгоритмов
- Бутстрап (bootstrap) и бэггинг (bagging)
- Бустинг (boosting)
- Блендинг (blending)
- Стэкинг (stacking)

Смещение и разброс. Минутка математики

Предположим, что у нас существует зависимость $y \equiv y(x) = f(x) + \varepsilon$

Она не совсем точная - есть шум, распределенный нормально $\varepsilon \sim \text{norm}(0, \sigma^2)$

Мы построили алгоритм $a = a(x)$, хотим оценить мат.ожидание квадрата отклонения его ответа от истины

$$\begin{aligned} E(y - a)^2 &= E(y^2 + a^2 - 2ya) = \\ &= E y^2 - (E y)^2 + (E y)^2 + E a^2 - (E a)^2 + (E a)^2 - 2f E a = \\ &= D y + D a + (E y)^2 + (E a)^2 - 2E ya = \\ &= D y + D a + f^2 + (E a)^2 - 2f E a = \\ &= D y + D a + (E(f - a))^2 \equiv \sigma^2 + \text{variance}(a) + \text{bias}^2(f, a) \end{aligned}$$

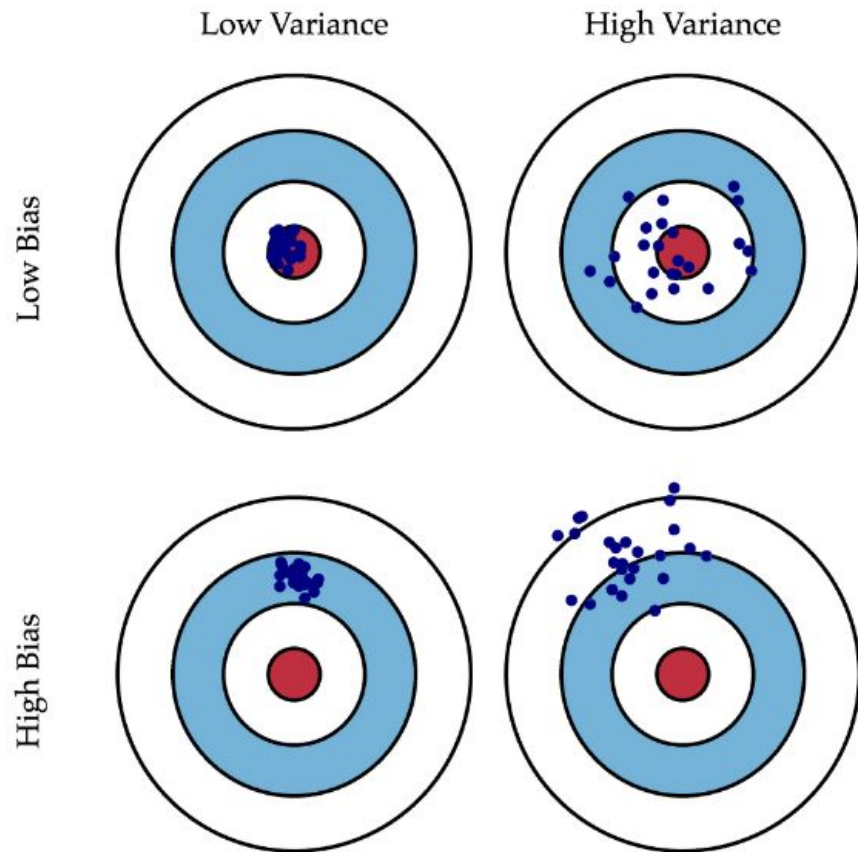
Смещение и разброс

Неустраняемая ошибка - шум в данных

Разброс (*variance*) - дисперсия ответов алгоритма $D\mathbf{a}$

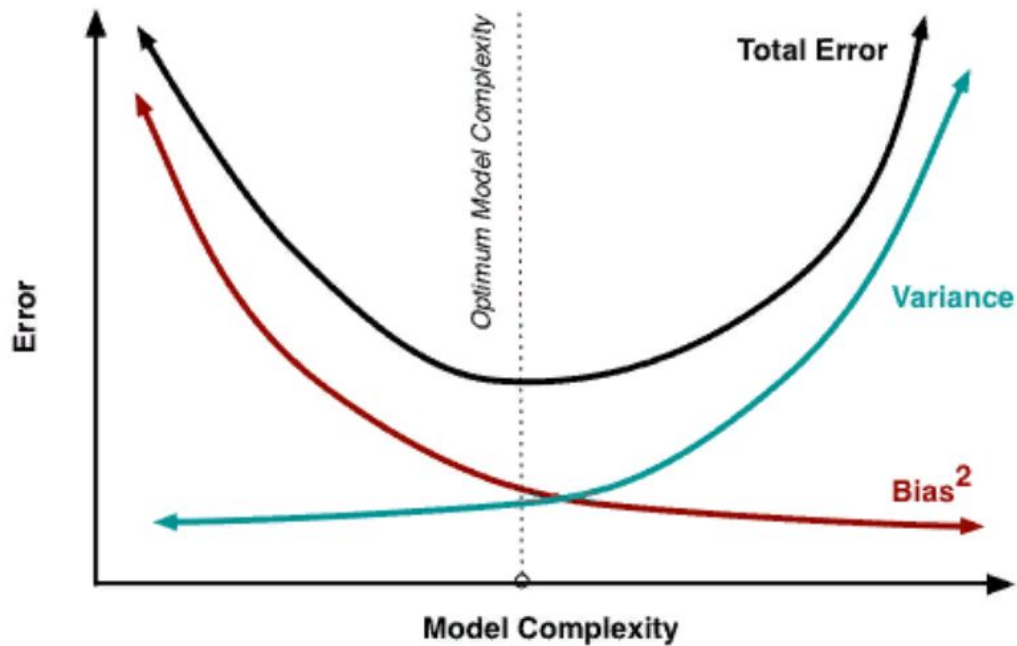
Смещение (*bias*) - мат.ожидание разности ответа алгоритма и истинного значения $E(\mathbf{f} - \mathbf{a})$

Надо что-то с этим делать



Смещение и разброс

В зависимости от сложности модели смещение и разброс варьируются. Нужен оптимум



Композиция алгоритмов. Ансамбль

По Ю.И.Журавлеву

Опр. 1.1. Композицией T алгоритмов $a_t(x) = C(b_t(x))$, $t = 1, \dots, T$ называется суперпозиция алгоритмических операторов $b_t: X \rightarrow R$, корректирующей операции $F: R^T \rightarrow R$ и решающего правила $C: R \rightarrow Y$:

$$a(x) = C(F(b_1(x), \dots, b_T(x))), \quad x \in X. \quad (1.1)$$

a - композиция алгоритмов

$b: X \rightarrow R$ - алгоритмический оператор или базовый алгоритм

R - пространство оценок

$F(b_1, \dots, b_T)$ - отображение из X в R

Говоря простым языком, исходные объекты отображаются в пространство оценок, оценки агрегируются и преобразуются решающим правилом в ответ, но сами оценки это не ответ

Композиция алгоритмов. Примеры

Теорема Кондорсе о присяжных

- равная вероятность правильного выбора для всех
- голосование честное, никто не ведет себя стратегически
- все принимают решение независимо

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

Мудрость толпы: в 1906 году 800 человек пытались угадать вес быка (1198 фунтов) на одном из рынков, никто не угадал, но после усреднения всех вариантов получилось 1197 фунтов

All three are correct

$$0.7 * 0.7 * 0.7 \\ = 0.3429$$

Two are correct

$$0.7 * 0.7 * 0.3 \\ + 0.7 * 0.3 * 0.7 \\ + 0.3 * 0.7 * 0.7 \\ = 0.4409$$

Two are wrong

$$0.3 * 0.3 * 0.7 \\ + 0.3 * 0.7 * 0.3 \\ + 0.7 * 0.3 * 0.3 \\ = 0.189$$

All three are wrong

$$0.3 * 0.3 * 0.3 \\ = 0.027$$

Общие идеи

Давайте применять все это в машинном обучении

- усреднение
- линейные комбинации
- смеси экспертов
- бустинг
- стэкинг
- логические правила
-

По наблюдениям ансамбли работают лучше, если его компоненты (отображения, базовые алгоритмы) различны, еще лучше - если исправляют ошибки друг друга

Источники разнообразия

Предобработка

- декомпозиция данных: по признакам / по объектам
- различные преобразования признаков
log sqrt tf-idf PCA NMF tSNE Encoder-Decoder W2V FastText BERT MeanEncoding OHE
- комбинация групп и преобразований

Сэмплирование

- по объектам (bootstrap, undersampling, oversampling)
- по признакам (RSM)

Источники разнообразия

Алгоритмы одного семейства с разными параметрами

- глубина дерева
- число соседей
- число слоев нейронов / число нейронов / оптимизатор
- параметры регуляризации
- число латентных факторов
- что еще?

Разные семейства алгоритмов

- линейные модели
- наивный байес
- факторизационные машины
- деревья
- unsupervised learning
- нейронные сети (разная топология)
- что еще?

Bootstrap и bagging. Bootstrap

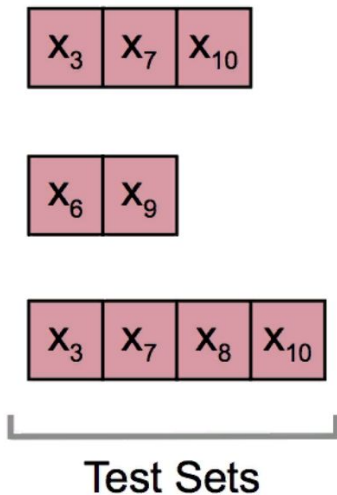
Имеется выборка из N объектов

Выбираем N объектов с возвращением

Объекты выбираются равновероятно

Повторяем M раз

Оцениваем статистики / что-то делаем



Original Dataset



Bootstrap 1



Bootstrap 2



Bootstrap 3



Training Sets



Bootstrap и bagging. Bagging

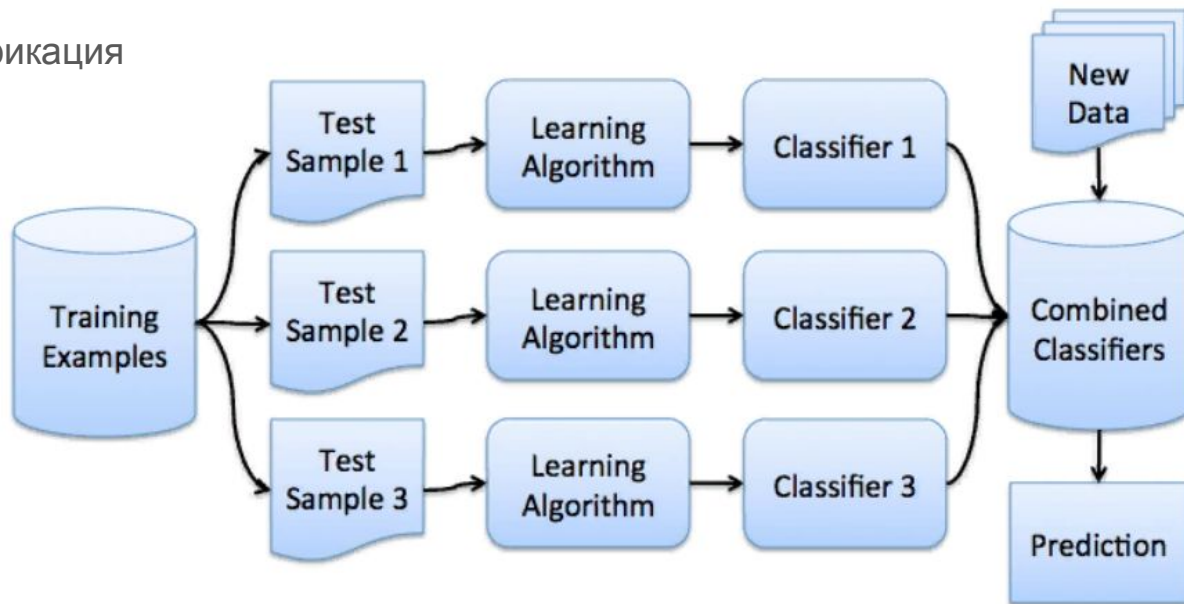
С помощью bootstrap генерируем n выборок

Обучаем независимо столько же моделей

Усредненные оценки алгоритмов = предсказание

Голосование если классификация

$$a(x) = \frac{1}{n} \sum_{i=1}^n b_i(x)$$



Bootstrap и bagging. Bagging

Наиболее простой и достаточно мощный метод

Особенно хорошо работает, если

- в базовых алгоритмах есть случайность
- используется семплирование
- базовые алгоритмы из разных семейств

$$a(x) = \frac{1}{n} \sum_{i=1}^n b_i(x)$$

Вариации

- взвешенное среднее
- ограничения на коэффициенты (неотрицательность, нормировка)
- усреднение порядков (если в разных шкалах или если l2r-задача)

$$\sum_{t=1}^T \alpha_t b_t(x), \quad x \in X, \quad \alpha_t \in \mathbb{R}$$

Bootstrap и bagging. Bagging

Почему работает?

Bootstrap и bagging. Минутка математики

Пусть есть базовые алгоритмы и их композиция $a(x) = \frac{1}{n} \sum_{i=1}^n b_i(x)$

Поищем среднеквадратичную ошибку

$$\begin{aligned} E_n &= E_x \left(\frac{1}{n} \sum_{i=1}^n b_i(x) - y(x) \right)^2 \\ &= E_x \left(\frac{1}{n} \sum_{i=1}^n \varepsilon_i \right)^2 \\ &= \frac{1}{n^2} E_x \left(\sum_{i=1}^n \varepsilon_i^2(x) + \sum_{i \neq j} \varepsilon_i(x) \varepsilon_j(x) \right) \\ &= \frac{1}{n} E_1 \end{aligned}$$

Позволяет снизить разброс (variance)

Bootstrap и bagging. Bagging

Плюсы:

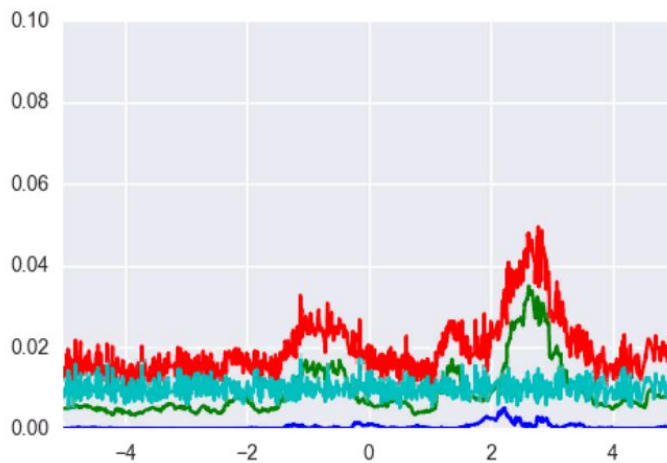
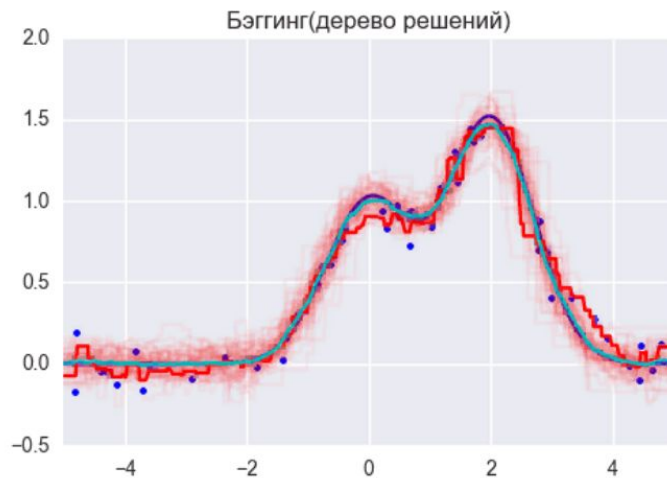
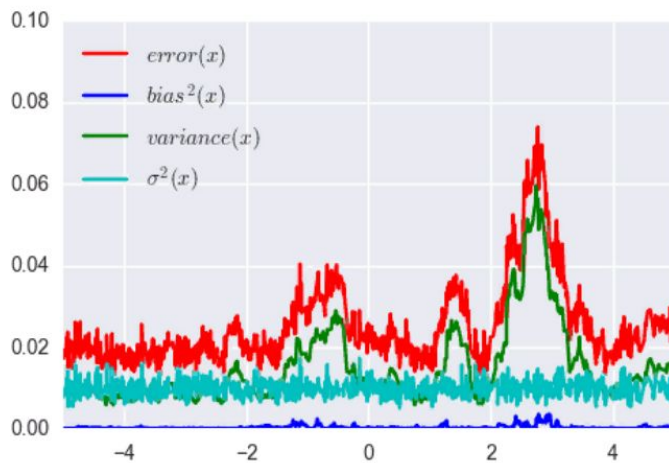
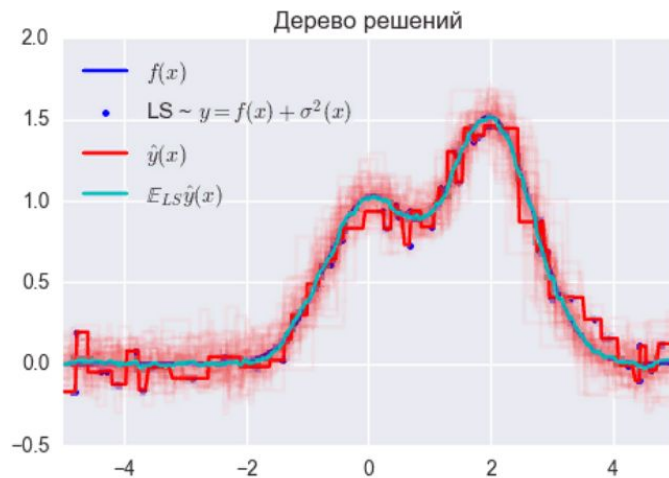
- легко реализовать
- эффективен
- снижает разброс
- предотвращает переобучение

Минусы:

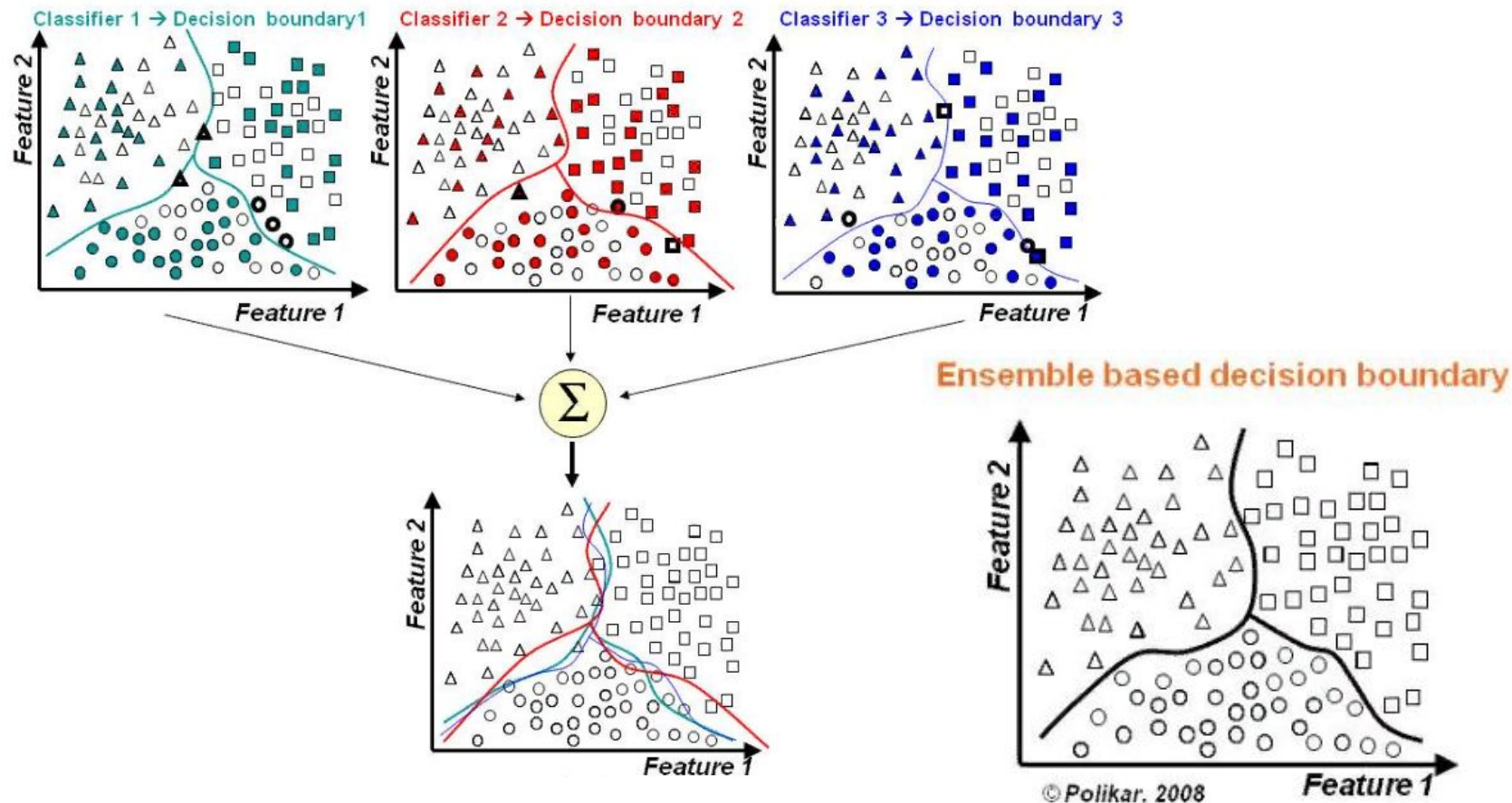
- довольно простой класс отображений

Коэффициенты смеси (веса) можно подбирать по лидерборду

Bootstrap и bagging. Примеры



Bootstrap и bagging. Примеры



Bootstrap и bagging. Примеры

Стандартный метод

DecisionTree + сэмплирование + усреднение \approx RandomForest

kaggle-avito-prohibited-content

kNN + линейная модель = решение в топ5

kaggle-malware

xgboost + oversample(a=7) + усреднение

kaggle-tube-pricing

NN + dropout\dropconnect + усреднение

любой конкурс

xgboost + сэмплирование(subsample, colsample_bytree) + усреднение

Boosting

Это метод взвешенного усреднения моделей

Новые базовые алгоритмы строятся последовательно









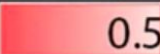
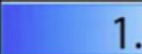
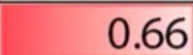
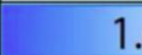


Новые алгоритмы опираются на предыдущие, пытаюсь улучшить качество

Rownum	x0	x1	x2	x3	y
0	0.94	0.27	0.80	0.34	1
1	0.84	0.79	0.89	0.05	1
2	0.83	0.11	0.23	0.42	1
3	0.74	0.26	0.03	0.41	0
4	0.08	0.29	0.76	0.37	0
5	0.71	0.76	0.43	0.95	1
6	0.08	0.72	0.97	0.04	0

Boosting. Weight based

Rownum	x0	x1	x2	x3	y	pred	abs.error
0	0.94	0.27	0.80	0.34	1	0.80	 0.20
1	0.84	0.79	0.89	0.05	1	0.75	 0.25
2	0.83	0.11	0.23	0.42	1	0.65	 0.35
3	0.74	0.26	0.03	0.41	0	0.40	 0.40
4	0.08	0.29	0.76	0.37	0	0.55	 0.55
5	0.71	0.76	0.43	0.95	1	0.34	 0.66
6	0.08	0.72	0.97	0.04	0	0.02	 0.02

Boosting. Weight based

Rownum	x0	x1	x2	x3	y	pred	abs.error	weight
0	0.94	0.27	0.80	0.34	1	0.80	 0.20	 1.20
1	0.84	0.79	0.89	0.05	1	0.75	 0.25	 1.25
2	0.83	0.11	0.23	0.42	1	0.65	 0.35	 1.35
3	0.74	0.26	0.03	0.41	0	0.40	 0.40	 1.40
4	0.08	0.29	0.76	0.37	0	0.55	 0.55	 1.55
5	0.71	0.76	0.43	0.95	1	0.34	 0.66	 1.66
6	0.08	0.72	0.97	0.04	0	0.02	 0.02	 1.02

Boosting. Weight based

Prediction = $\text{pred}_0 \cdot \eta + \text{pred}_1 \cdot \eta + \dots + \text{pred}_N \cdot \eta$

Rownum	x0	x1	x2	x3	y	weight
0	0.94	0.27	0.80	0.34	1	1.20
1	0.84	0.79	0.89	0.05	1	1.25
2	0.83	0.11	0.23	0.42	1	1.35
3	0.74	0.26	0.03	0.41	0	1.40
4	0.08	0.29	0.76	0.37	0	1.55
5	0.71	0.76	0.43	0.95	1	1.66
6	0.08	0.72	0.97	0.04	0	1.02

Boosting. Residual based

Rownum	x0	x1	x2	x3	y	pred	error
0	0.94	0.27	0.80	0.34	1	0.80	0.20
1	0.84	0.79	0.89	0.05	1	0.75	0.25
2	0.83	0.11	0.23	0.42	1	0.65	0.35
3	0.74	0.26	0.03	0.41	0	0.40	-0.40
4	0.08	0.29	0.76	0.37	0	0.55	-0.55
5	0.71	0.76	0.43	0.95	1	0.34	0.66
6	0.08	0.72	0.97	0.04	0	0.02	-0.02

Boosting. Residual based

Rownum	x0	x1	x2	x3	y
0	0.94	0.27	0.80	0.34	0.2
1	0.84	0.79	0.89	0.05	0.25
2	0.83	0.11	0.23	0.42	0.35
3	0.74	0.26	0.03	0.41	-0.4
4	0.08	0.29	0.76	0.37	-0.55
5	0.71	0.76	0.43	0.95	0.66
6	0.08	0.72	0.97	0.04	-0.02

Boosting. Residual based

Prediction = prediction_0 + pred_1*eta + ... + pred_N*eta

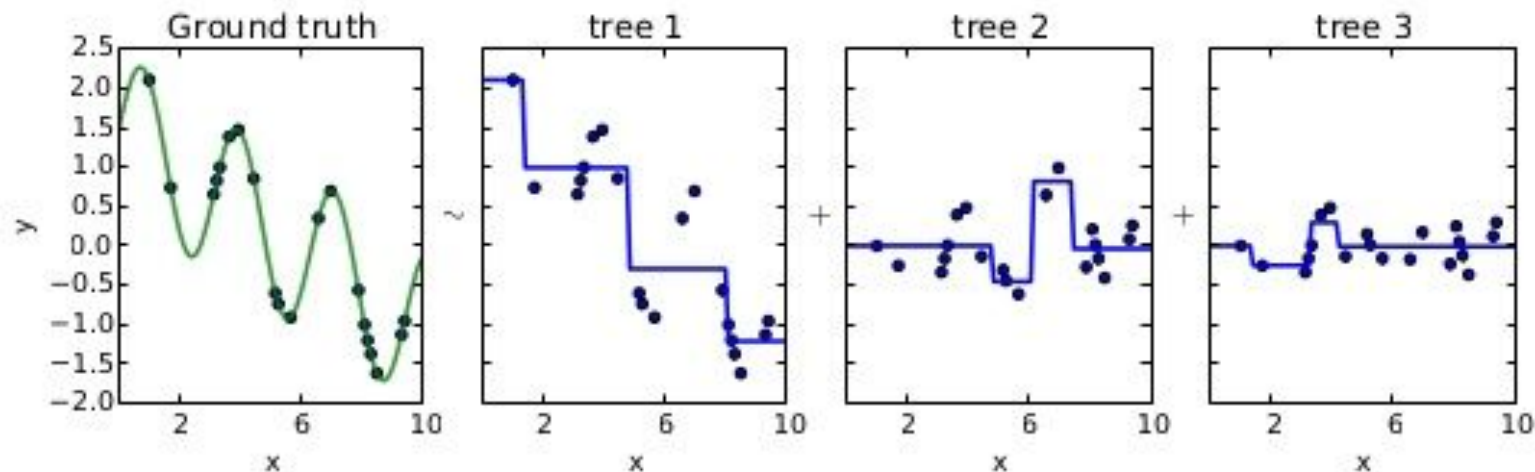
Rownum	x0	x1	x2	x3	y	new pred	old pred
0	0.94	0.27	0.80	0.34	0.2	0.15	0.80
1	0.84	0.79	0.89	0.05	0.25	0.20	0.75
2	0.83	0.11	0.23	0.42	0.35	0.40	0.65
3	0.74	0.26	0.03	0.41	-0.4	-0.30	0.40
4	0.08	0.29	0.76	0.37	-0.55	-0.20	0.55
5	0.71	0.76	0.43	0.95	0.66	0.24	0.34
6	0.08	0.72	0.97	0.04	-0.02	-0.01	0.02

Для примера 0: предсказание будет равным $0.15 + 0.8 = \mathbf{0.95}$

Boosting. Residual based

Prediction = prediction_0 + pred_1*eta + ... + pred_N*eta

Residual fitting



Boosting

Почему бустинг работает?

Нельзя ли придумать для ансамблирования что-то сложнее чем усреднение или голосование?

Смеси экспертов

$$\sum_{t=1}^T g_t(x) b_t(x)$$

Это сумма взвешенных оценок алгоритмов

Вес g_t - функция от положения в признаковом пространстве

Линейная комбинация - частный случай смеси экспертов ($g_t = \text{const}$)

Как учить функции весов?

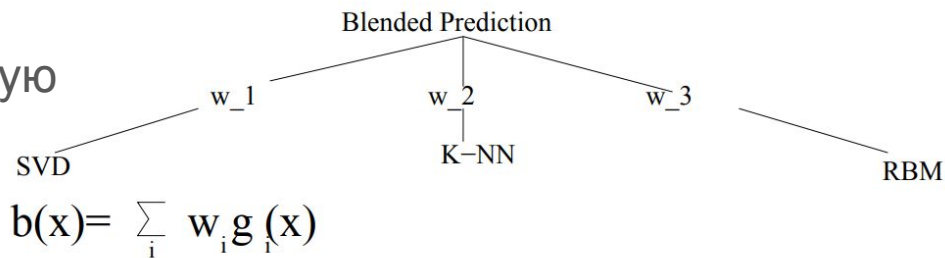
- ограничиться некоторым семейством функций
- подбирать параметры этого семейства

Важный частный случай -- линейное семейство весовых функций

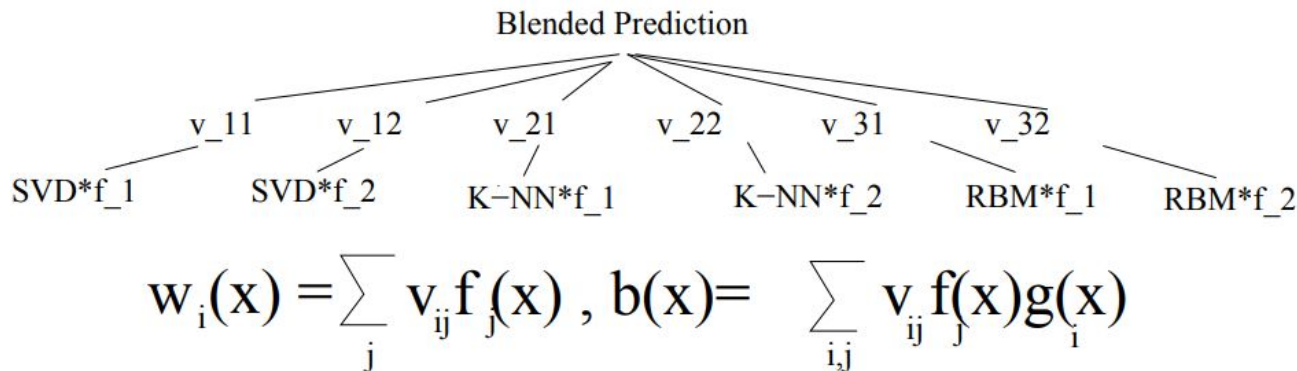
Feature-Weighted Linear Stacking

Настраиваем веса взвешенной суммы ответов базовых алгоритмов как линейную комбинацию нескольких специально отобранных признаков

Standard Linear Stacking



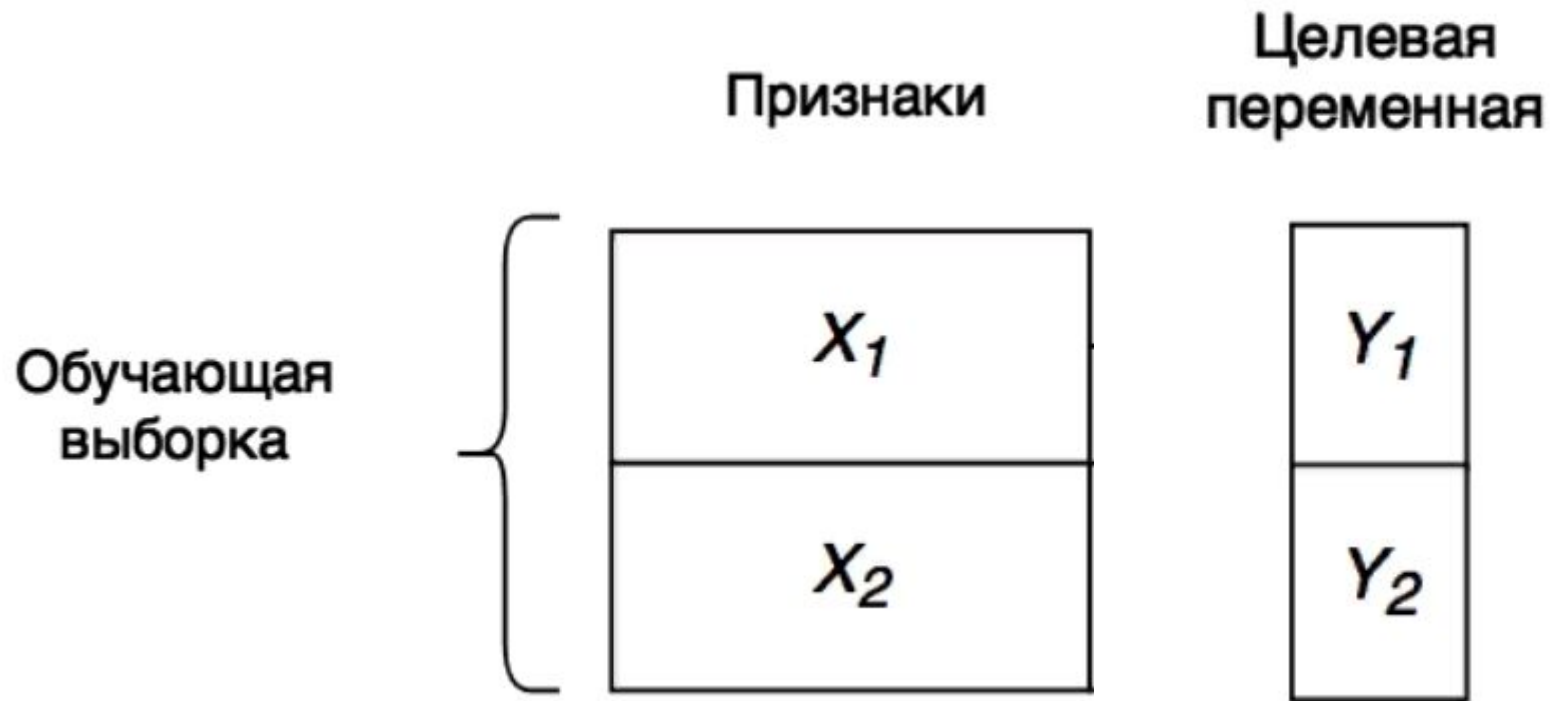
Feature-Weighted Linear Stacking



Нельзя ли доверять ансамблирование другому алгоритму?

Blending (Holdout stacking)

Делим выборку на две части



Blending (Holdout stacking)

Делим выборку на две части

На одной обучаем базовые алгоритмы

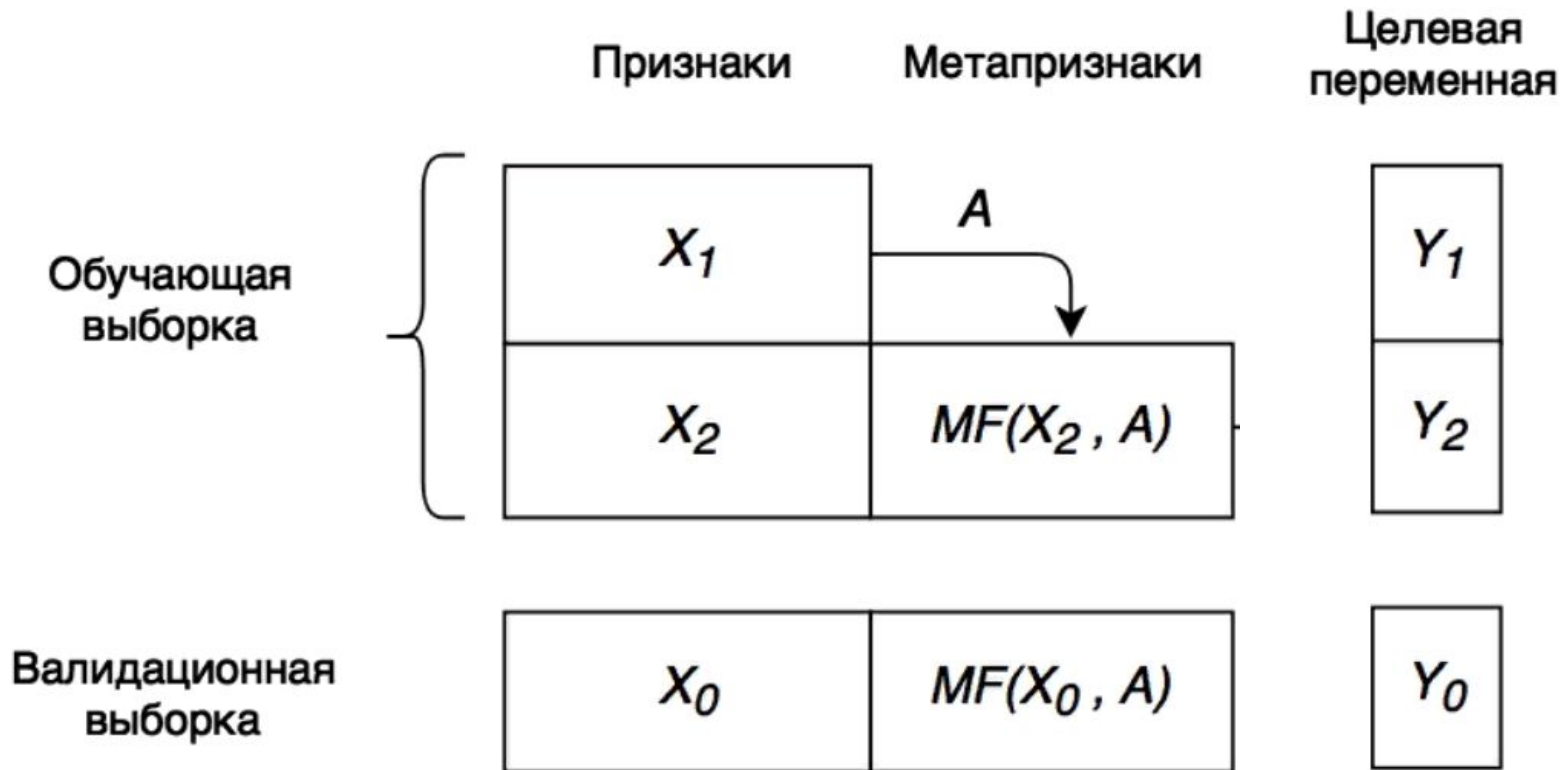


Blending (Holdout stacking)

Делим выборку на две части

На одной обучаем базовые алгоритмы

На другой части и на тестовой выборке рассчитываем пространство оценок



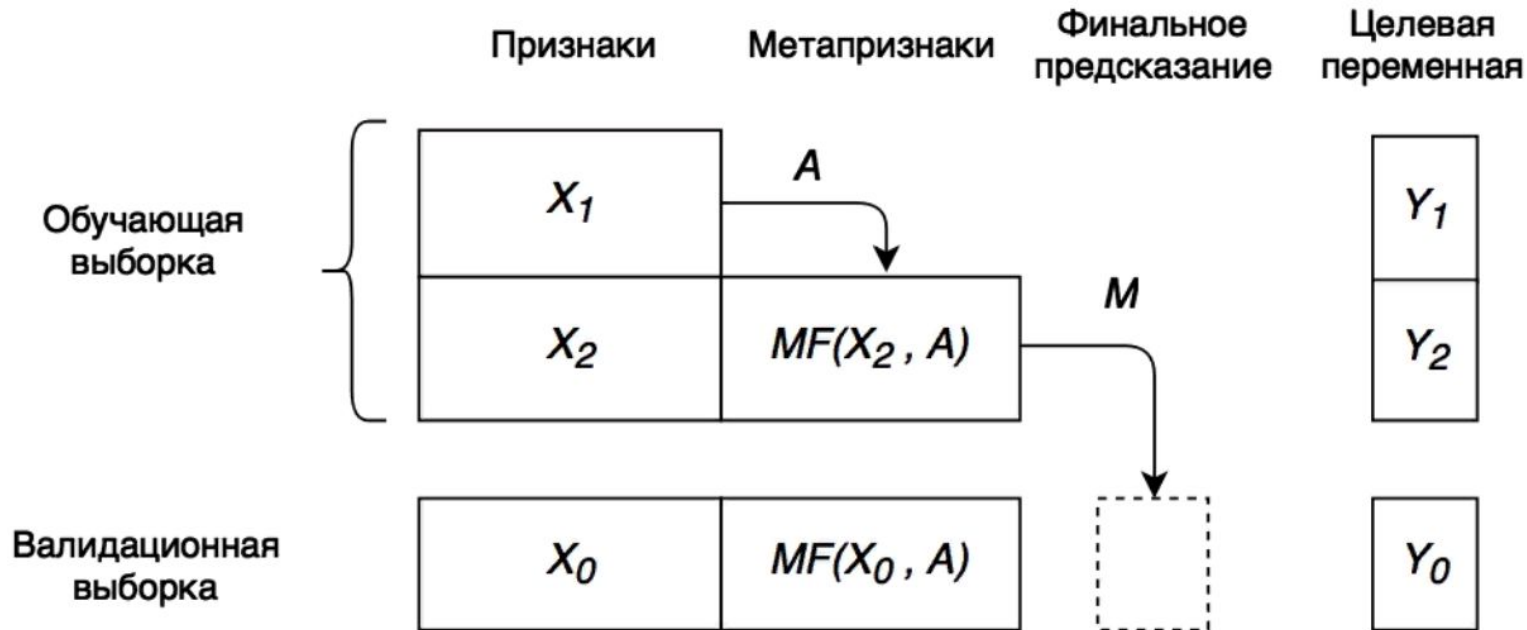
Blending (Holdout stacking)

Делим выборку на две части

На одной обучаем базовые алгоритмы

На другой части и на тестовой выборке рассчитываем пространство оценок

На метапризнаках второй части настраиваем метаалгоритм



На метапризнаках теста получаем финальный ответ

Blending (Holdout stacking)

Что не так?

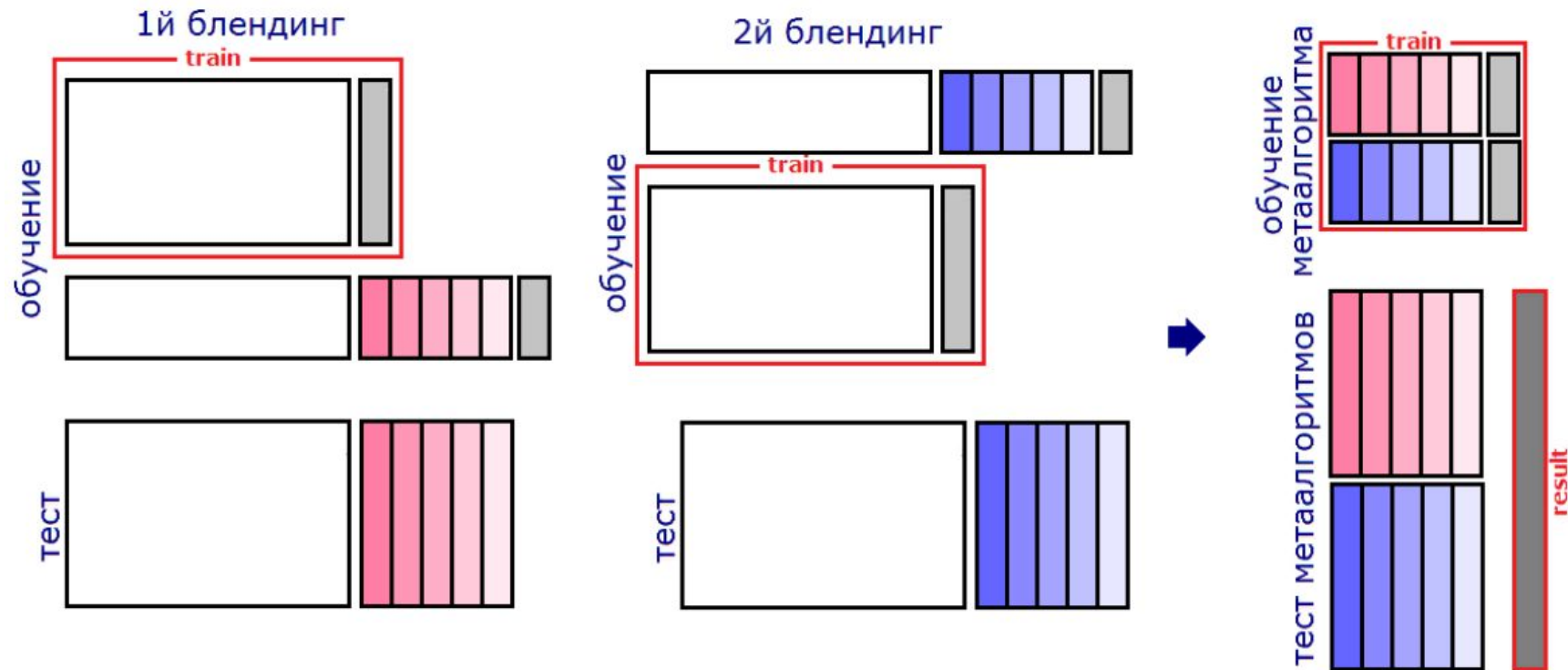
Blending (Holdout stacking)

Что не так?

Ни базовые алгоритмы, ни метаалгоритм не используют всю обучающую выборку

Blending (Holdout stacking). Надо получше

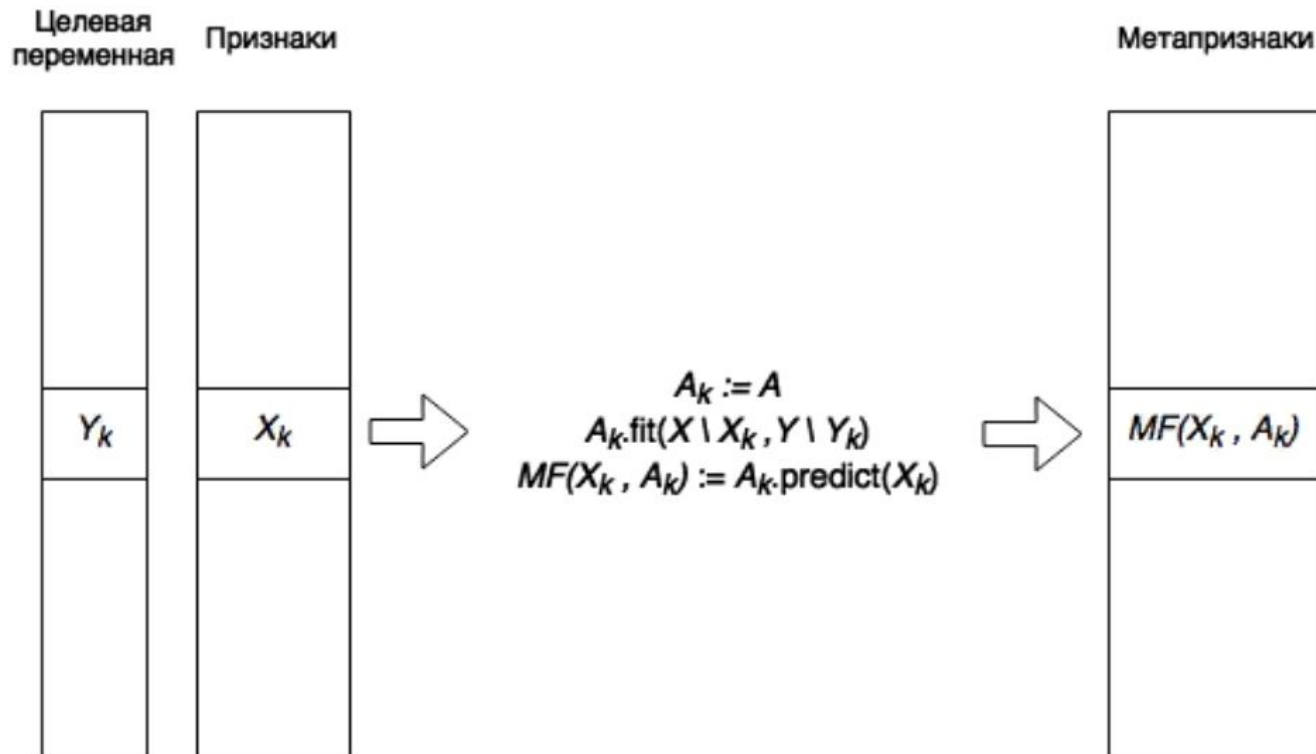
- усредним несколько блендингов с разными разбиениями
- или сконкатенируем



Stacking

Боремся за использование всей обучающей выборки

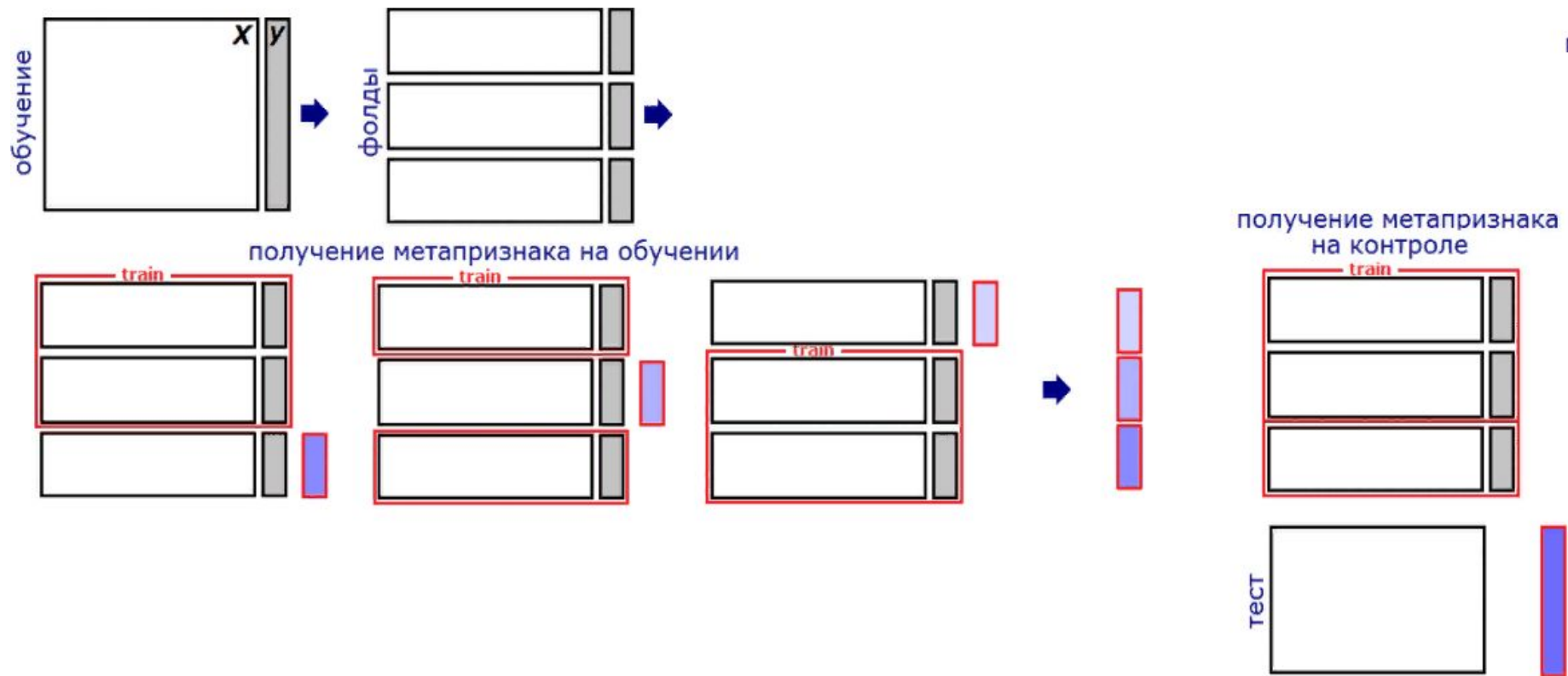
Хотим метапризнаки



Stacking

- разобьем выборку на фолды
- перебирая фолды обучаем базовый алгоритм на всех кроме одного
- на оставшемся получаем оценки (метапризнаки)
- для получения аналогичных оценок на тестовой выборке базовые алгоритмы обучают на всей обучающей выборке

Stacking



Stacking

Основная идея

пытаемся восстановить отображение $F(b_1, \dots, b_T)$ методами машинного обучения

Можем реализовывать нелинейные отображения

RF / GBM / NN

Один из самых эффективных способов

Самый трудоемкий

Легко переобучиться

аккуратно работаем с данными
расчетляем методы регуляризации

Stacking

Основная идея

пытаемся восстановить отображение $F(b_1, \dots, b_T)$ методами машинного обучения

Можем реализовывать нелинейные отображения

RF / GBM / NN

Один из самых эффективных способов

Самый трудоемкий

Легко переобучиться

аккуратно работаем с данными
расчетляем методы регуляризации

Stacking. О разбиении данных

Нельзя допускать банальных ошибок

обучение и валидация в каждый момент не должны пересекаться

Если данных было бы много - не было бы проблем

их мало, пользуемся техникой аналогичной кроссвалидации

Обязателен честный holdout при тестировании идей

хотя бы простое разбиение 40 / 30 / 30



Stacking. Еще полезно

Разбиение лучше держать фиксированным для всех моделей ($5 < k < 10$)

Можем переразбить фолды

усреднить метапризнаки

Дополнительная регуляризация

добавим нормальный шум к оценкам / метапризнакам (обучение vs тест)

Out-of-fold(2)*M

случайное разбиение каждый раз

на целевом множестве усредняем значение двух моделей для каждого из M шагов

надо помнить $2 \cdot M$ моделей

есть случаи когда работает лучше



Stacking. Еще полезно

Важно и нужно использовать базовые алгоритмы разной природы

Разные признаковые пространства (включая RSM)

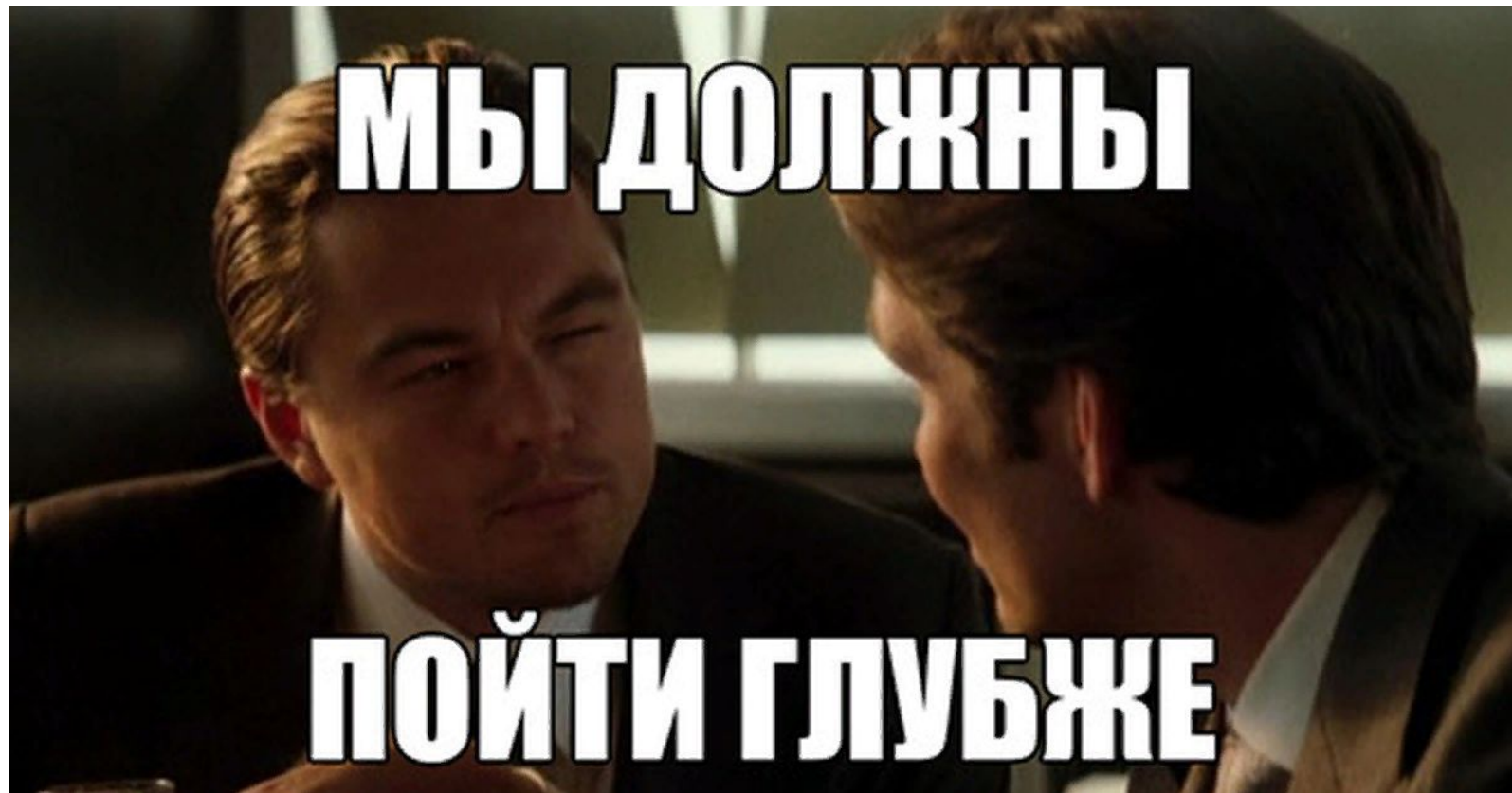
Недооптимизация базовых алгоритмов

Сильная регуляризация в базовых алгоритмах

Иногда хорошо работает обучение не на целевой признак, а на разницу между целевым и каким-то другим

Feature Engineering над метапризнаками (попарные произведения)

Stacking



Stacking. Глубже

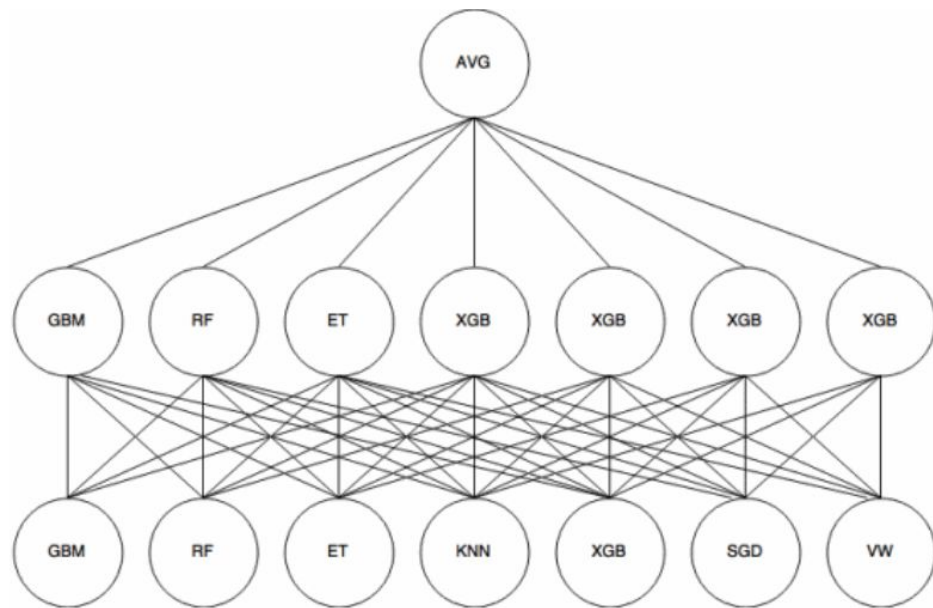
3 уровня обучения - вполне реально

Можно ли 4?

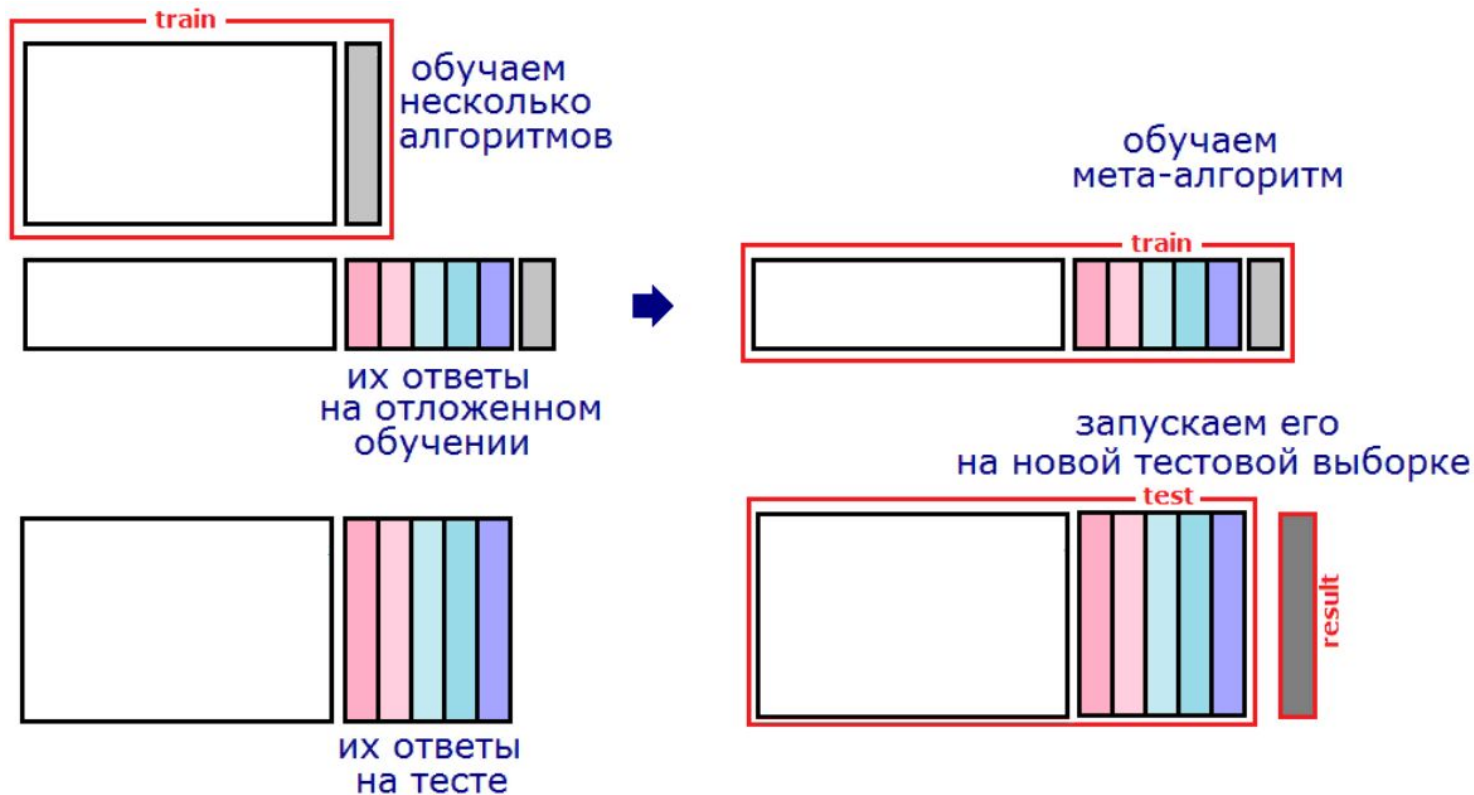
Можно ли автоматизировать построение ансамбля?

StackNet (by KazAnova)

<https://github.com/kaz-Anova/StackNet>

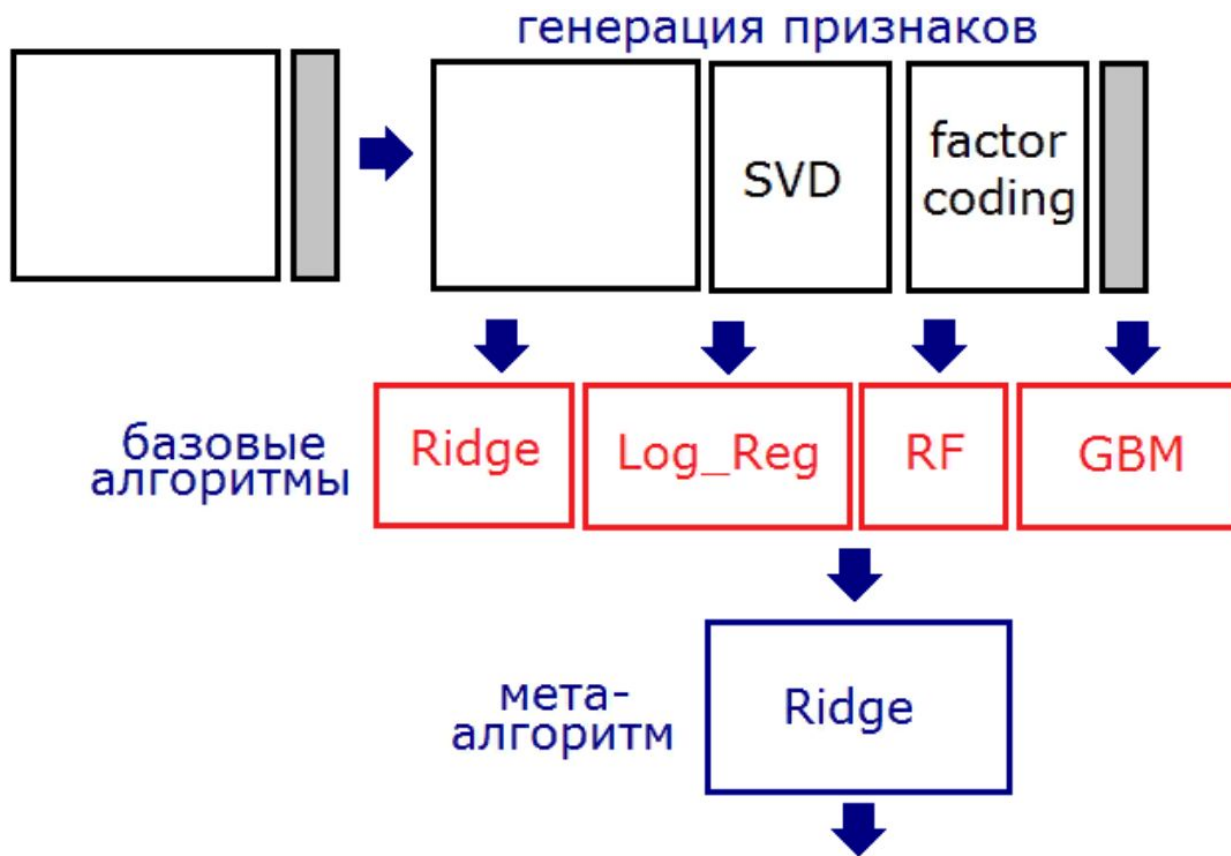


Stacking. Признаки и метапризнаки



Не всегда полезно

Stacking. Связь с другими техниками



Stacking

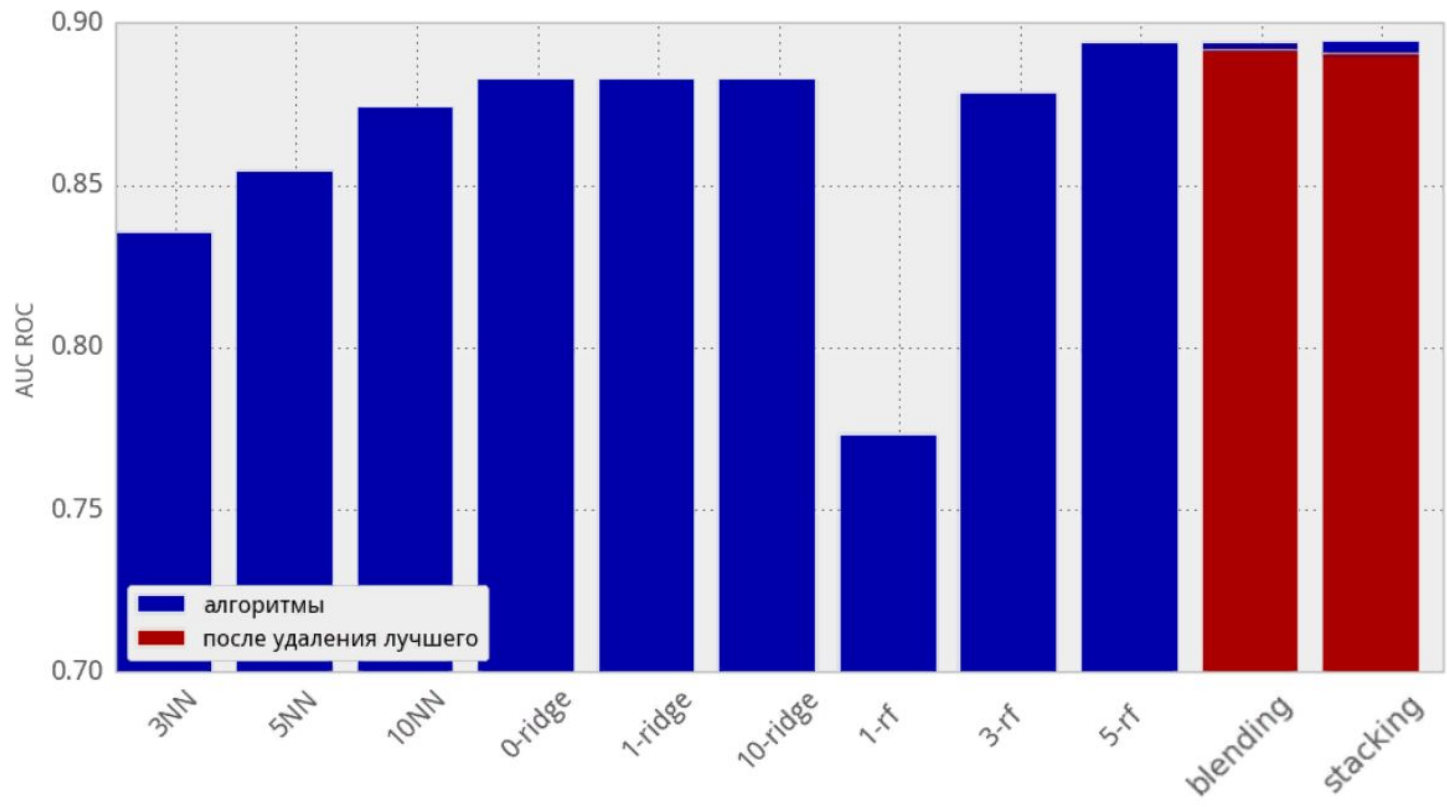
Типичные модели первого уровня

- ❑ 2-3 gradient boosted trees (lightgb, xgboost, H2O, catboost)
- ❑ 2-3 Neural nets (keras, pytorch)
- ❑ 1-2 ExtraTrees/Random Forest (sklearn)
- ❑ 1-2 linear models as in logistic/ridge regression, linear svm (sklearn)
- ❑ 1-2 knn models (sklearn)
- ❑ 1 Factorization machine (libfm)
- ❑ 1 svm with nonlinear kernel if size/memory allows (sklearn)

На каждые 7-8 базовых моделей - модель второго уровня

Stacking. Все хорошо но ...

Нужно понимать, что не всегда результат существенно превышает качество лучшего базового алгоритма



Stacking. Все хорошо но ...

Нужно понимать, что не всегда существенно повышает качество лучшего базового алгоритма

Выборка должна быть достаточно большого объема

Временные ряды - ок?

Stacking

Характерный прирост качества при использовании ансамблей 1-3%

[kaggle-avito-prohibited-content](#)

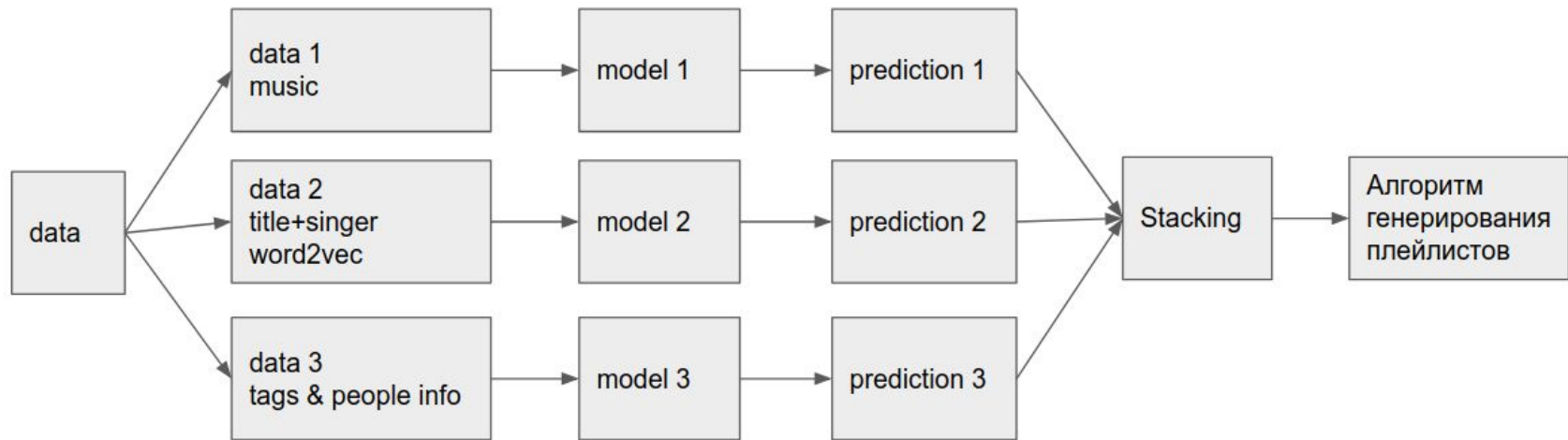
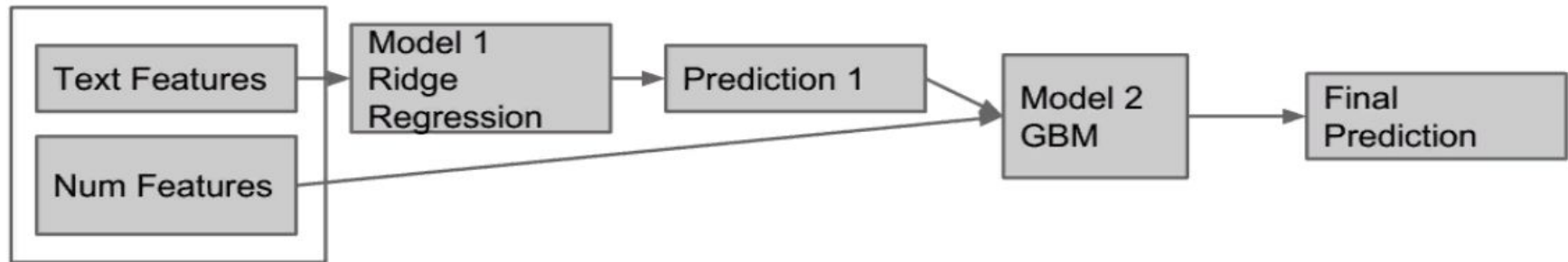
Алгоритм	Результат (MAP@10)
SVM	0.980
FM	0.981
2*SVM	0.982
2*FM	0.983
LR(FM, SVM)	0.984
RF(FM, SVM)	0.987

[kaggle-shoppers*](#)

Алгоритм	Результат (ROC-AUC)
Vowpal Wabbit A	0.59962
Vowpal Wabbit B	0.59957
Vowpal Wabbit C	0.59954
GLMNet	0.59665
Усреднение	0.60031
Усреднение рангов	0.60187

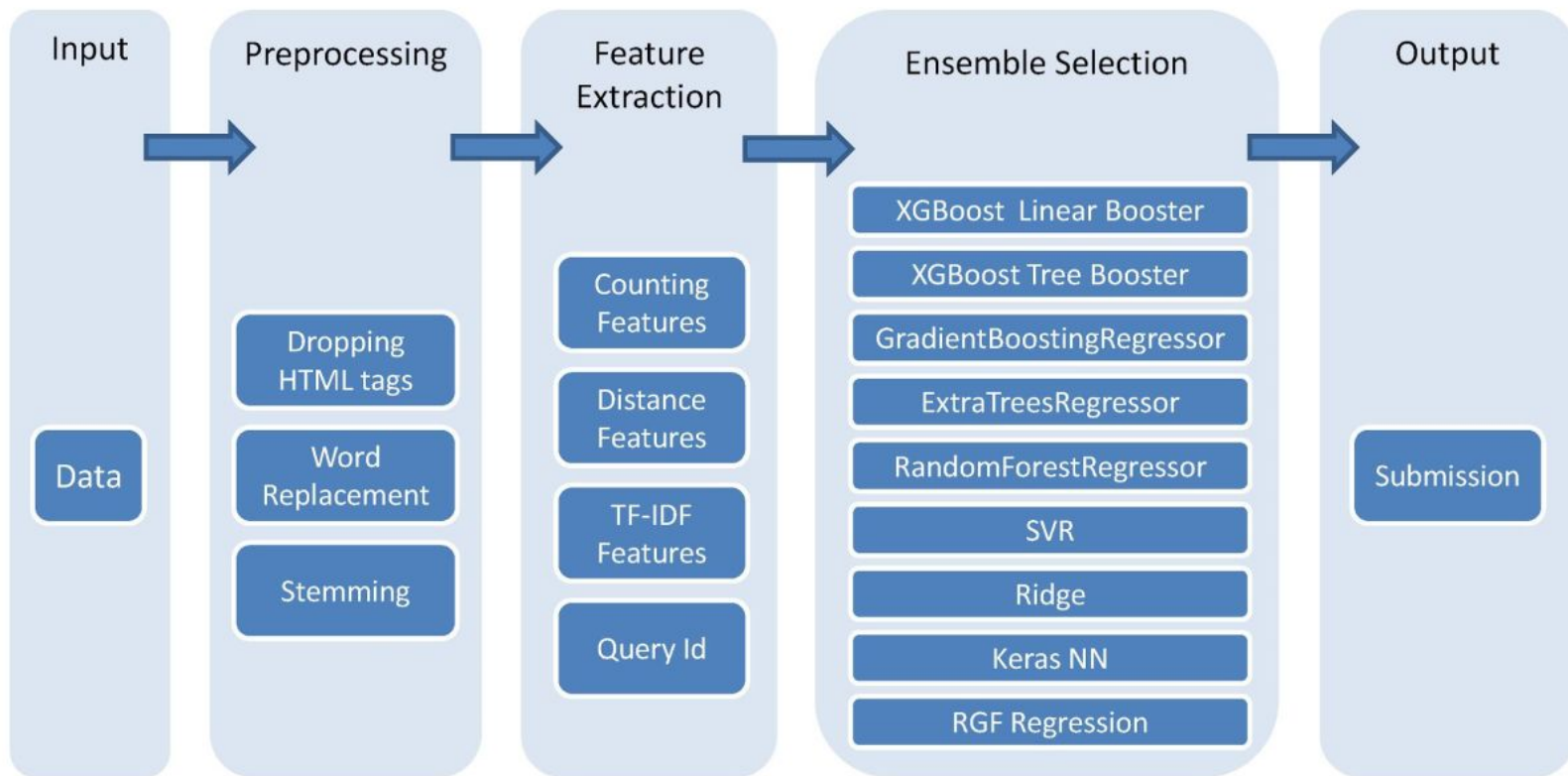
Stacking. Примеры

Типичные архитектуры



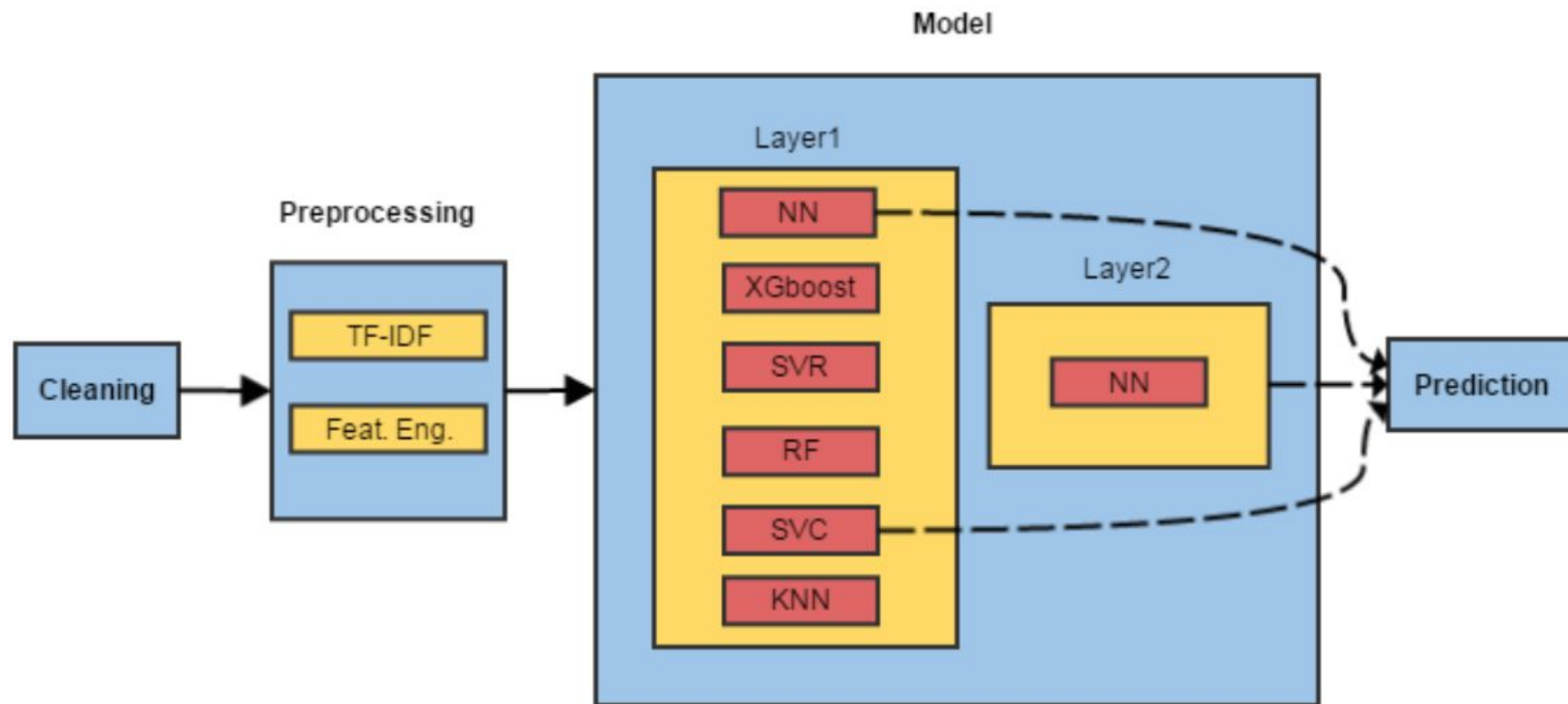
Stacking. Примеры

CrowdFlower (top 1)



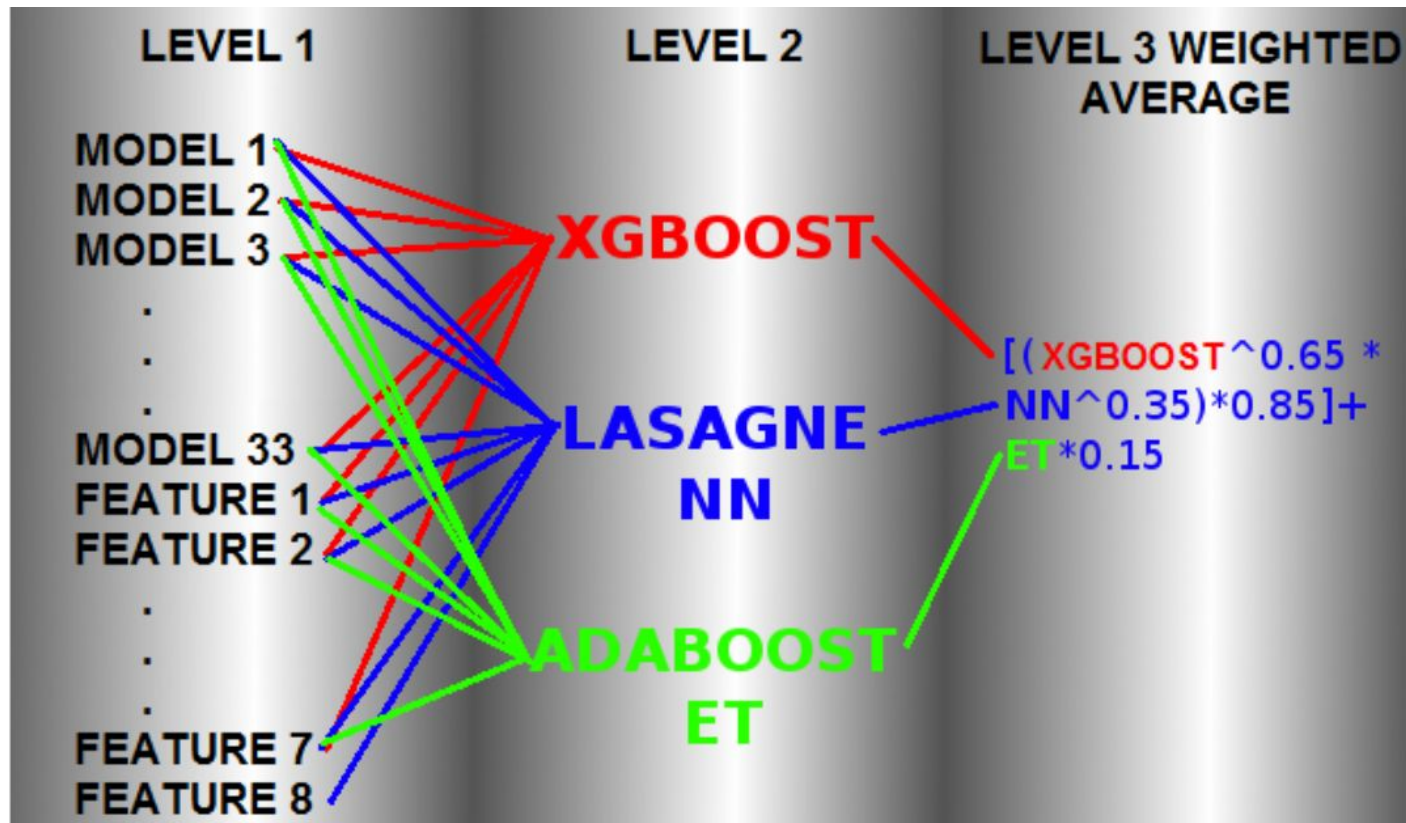
Stacking. Примеры

CrowdFlower (top 3)



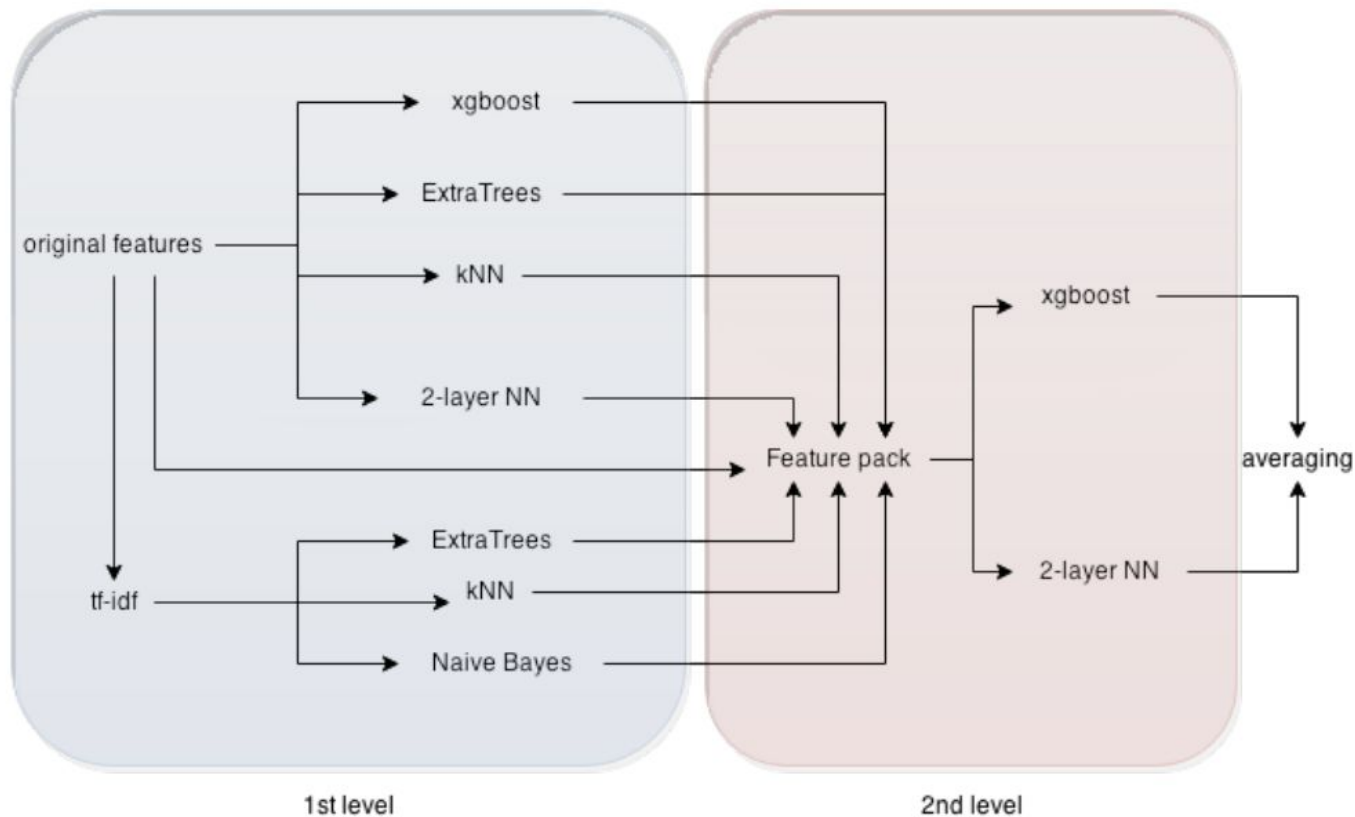
Stacking. Примеры

Otto (top 1)



Stacking. Примеры

Otto (top 5)





Data Mining In Action