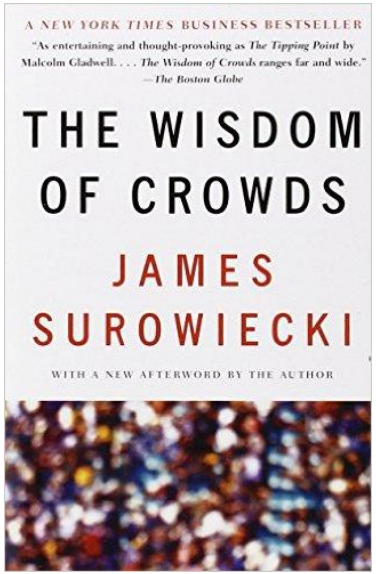
Ensemble Learning in ML

Кирилл Данилюк, Yandex SDC Team

Интуиция–1

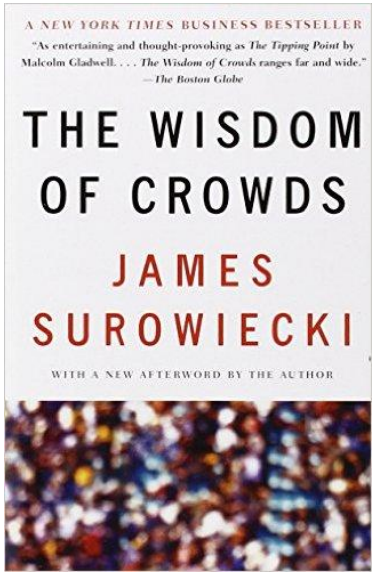
- Человек мыслит аналогиями
- В **определённых** ситуациях решения, принятые группой, превосходят по точности индивидуальные решения
- Люди **иногда** принимают решения, взвешивая мнения других
- Пример — консилиумы врачей и учёных
- Не подходит для творческих решений



https://www.amazon.com/Wisdom-Crowds-James-Surowiecki/dp/0385721706

Интуиция–2

- Различия в точках зрения участников
- Различия в интерпретациях, допускаются эксцентричные идеи и взгляды
- Независимость принятия решений
- Специализация участников, доменные знания
- Наличие алгоритма агрегации отдельных решений в коллективное решение



https://www.amazon.com/Wisdom-Crowds-James-Surowiecki/dp/0385721706

Голосование большинством

3 нескоррелированных классификатора (допустим, бинарная классификация) с точностью 70% каждый.

Как изменится их точность, если мы устроим среди них голосование и возьмём выбранное большинством значение?

Голосование — решение

- 3 классификатора с точностью 70% каждый (“0” — ошибка)
- При большинстве голосов улучшают точность до 78%
- 5 clf. Точность: 83%

```
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
```

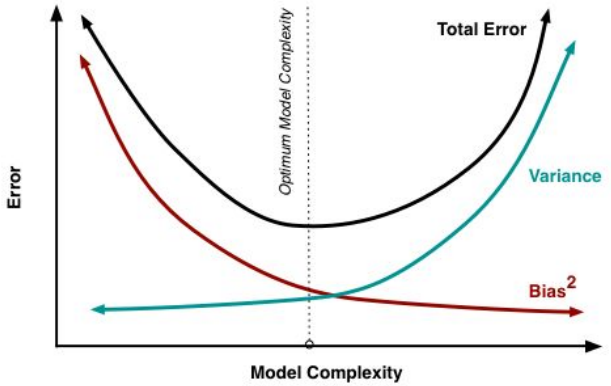
https://mlwave.com/kaggle-ensembling-guide/

Bias-variance tradeoff

$$Err(x) = E\left[(Y - \hat{f}(x))^2\right]$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right] + \sigma_e^2$$

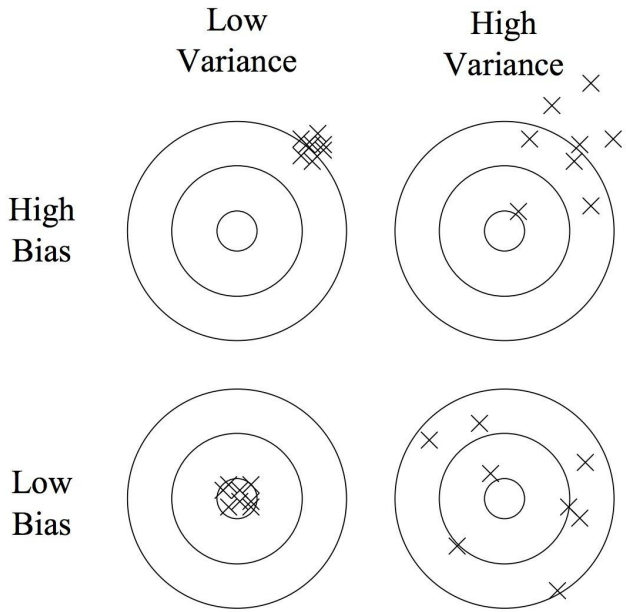


- Смещение — среднее отклонение прогноза модели от прогноза идеальной модели
- Разброс — дисперсия ответов модели, обученных на разных подвыборках
- Неустранимая ошибка внутри самих данных

http://scikit-learn.org/stable/auto_examples/ensemble/plot_bias_variance.html

Bias-variance tradeoff

- Компромисс между качеством на train и силой модели на test
- Low bias — сложные модели => **переобучение**
- Low variance — модели, не учитывающие сложность данных => **недообучение**

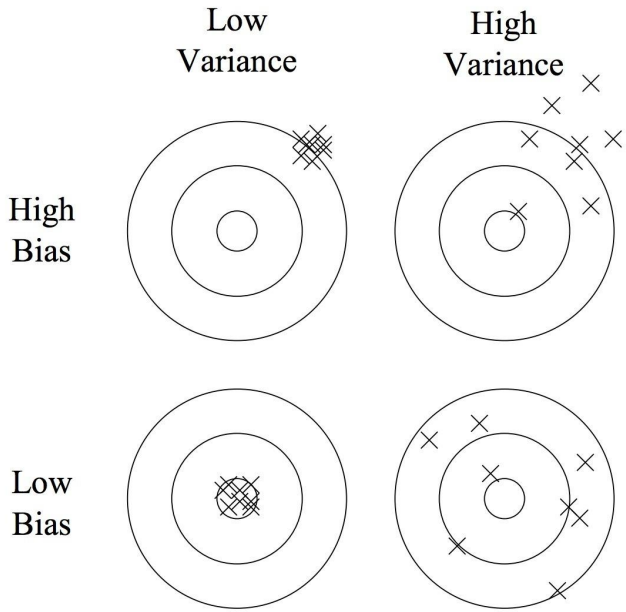


http://scott.fortmann-roe.com/docs/BiasVariance.html

Bias-variance tradeoff

Резюме:

- Ошибка модели раскладывается на смещение и разброс + шум в данных
- Корреляция между предсказаниями моделей играет важную (и негативную) роль
- Разброс можно уменьшить, снизив корреляцию между моделями в ансамбле



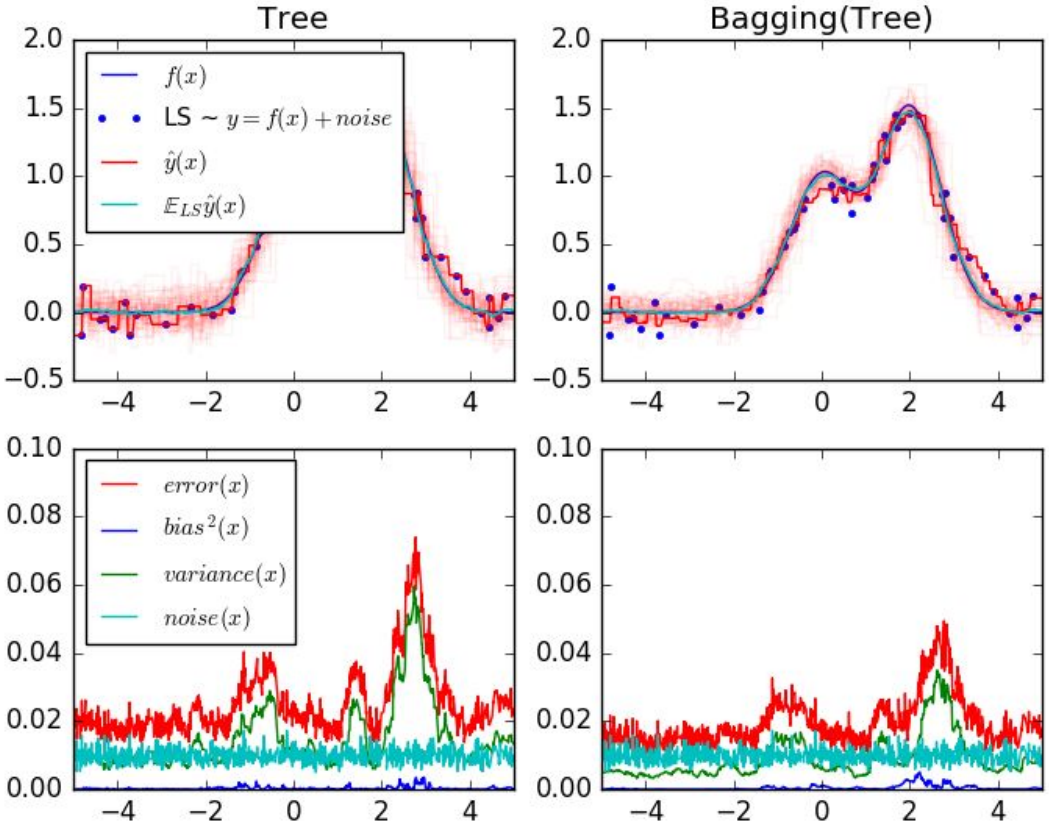
http://scott.fortmann-roe.com/docs/BiasVariance.html

Ансамбли моделей

- Помогают снизить ошибку моделей за счёт снижения разброса или смещения
- Виды:
  - Бутстрап (или subsampling) + усреднение — бэггинг
  - Использование случайных подпространств
  - Направленное построение ансамбля (бустинг)
  - Метамодели (Wolpert (1992) Stacked Generalization)

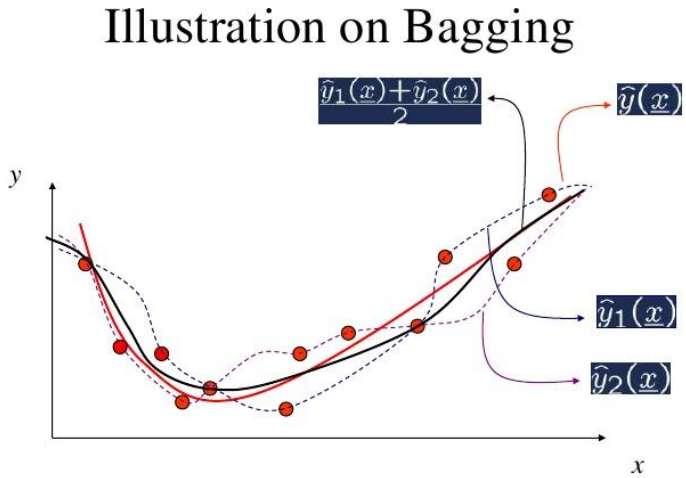
ftp://nic.funet.fi/pub/sci/neural/neuroprose/wolpert.stack_gen.ps.Z

Ансамбли моделей



Бэггинг–1

- Bagging: **b**ootstrap **agg**regat**ing**
- Используем бутстрап / подмножества тренировочной выборки **с возвращением**
- Часто этого недостаточно. Тогда дополнительно используем рандомизированные подмножества признаков
- Агрегируем модели одного типа
- Каждая модель считается в ансамбле с одинаковым весом
- `BaggingClassifier` в sklearn



https://www.kaggle.com/c/overfitting

Бэггинг–2: Бутстрап

- Непараметрический метод.
- Позволяет оценить эмпирическую функцию распределения на основе многократной генерации псевдовыборок с возвращением (методом Монте-Карло)
- По эмпирической функции распределения можно вычислять нужные статистики: дисперсию, квантили и т.д.

Бэггинг–3: OOBE

- Out of Bag Error: «встроенная» в бутстрап возможность кросс-валидироваться на не попавших в выборку примерах
- В выборку не попадает $(1 - \frac{1}{\ell})^{\ell}$ примеров, что в пределе означает, что дерево обучается на 63% выборки, а 37% можно оставить на валидацию.

Случайный лес

- Ещё более рандомизированный вариант бэггинга
- Алгоритм:
  - Выделяем бутстрапом N случайных подвыборок размера X_n
  - Для каждой подвыборки независимо строим CART-дерево заданной глубины (либо до тех пор, пока в листе не окажется заданное число объектов)
  - При каждом сплите рандомизируем подпространство признаков
  - Объединяем в ансамбль для классификации: $a(x) = \text{sign}\, \frac{1}{N} \sum_{n=1}^{N} b_n(x).$

https://www.analyticsvidhya.com/blog/2014/06/introduction-random-forest-simplified/

Бустинг

- Концепция weak learners (базовых алгоритмов): натренировать слабые классификаторы намного проще (bias-variance tradeoff)
- Из множества weak сделать один strong learner
- Направленное построение ансамбля, в отличие от RF
- Стратегия:
  - Базовые алгоритмы строятся последовательно, один за другим
  - Каждый следующий алгоритм строится так, чтобы исправлять общую ошибку текущего ансамбля

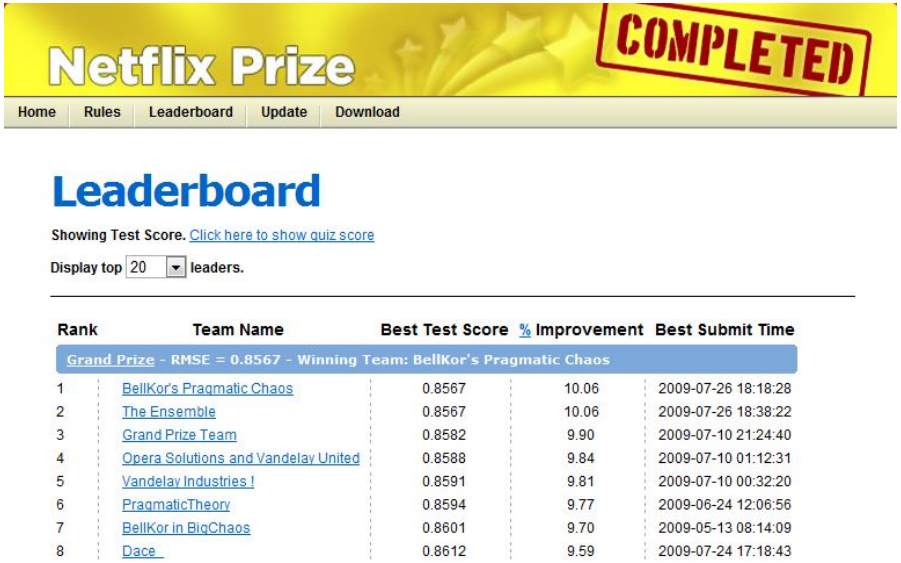
Градиентный бустинг

- Дана обучающая выборка $\{(x_i, y_i)\}_{i=1}^n$, дифференцируемый лосс $L(y, F(x))$ и M (число итераций). Найти F_M(x).
- Инициализация базового алгоритма (регрессия): $F_0(x) = \arg\min_\gamma \sum_{i=1}^n L(y_i, \gamma).$
- На каждом шаге m = 1 .. M:
  - Вычисляем псевдо-остатки: $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$
  - Фитим базовые алгоритмы $h_m(x)$ на псевдо-остатки, используя фичи из трейна: $\{(x_i, r_{im})\}_{i=1}^n$
  - Вычисляем коэффициент линейной комбинации:

$$\gamma_m = \arg\min_\gamma \sum_{i=1}^n L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right)$$

- Обновим модель: $F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$
- Выведем F_M(x).

Метамодели

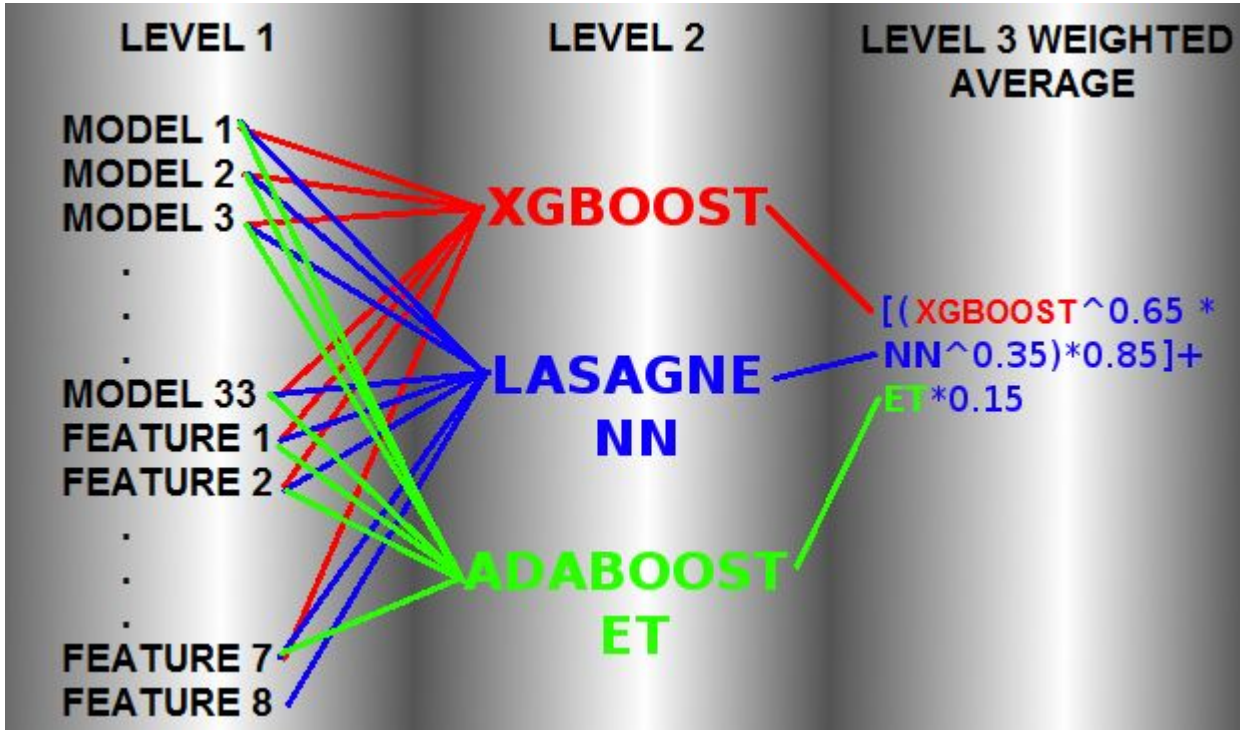


“This is a truly impressive compilation and culmination of years of work, blending hundreds of predictive models to finally cross the finish line.

We evaluated some of the new methods offline but the additional accuracy gains that we measured did not seem to justify the engineering effort needed to bring them into a production environment.”

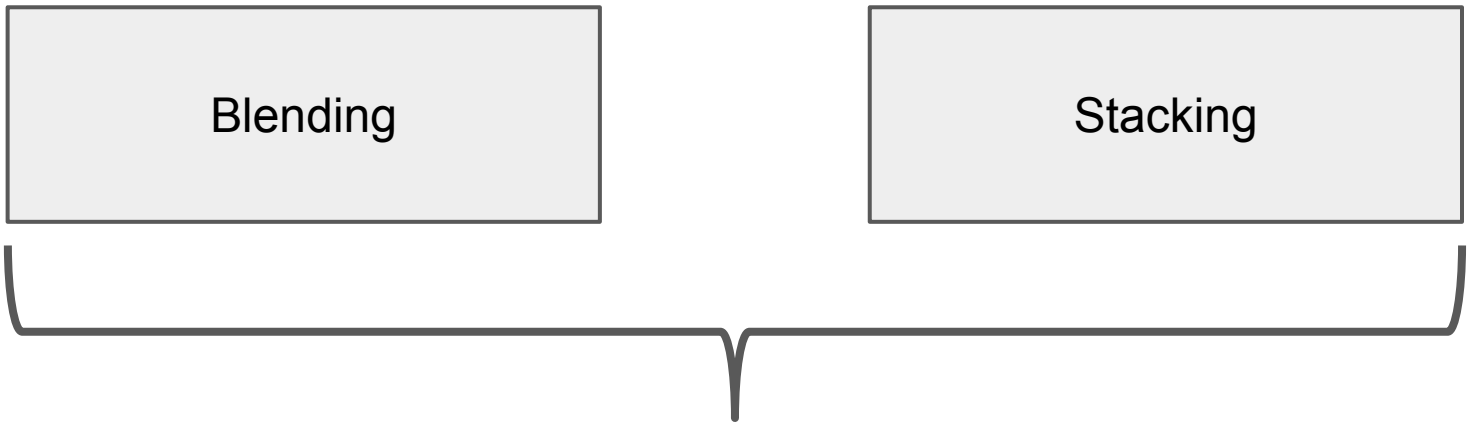
Netflix Engineers

Метамодели



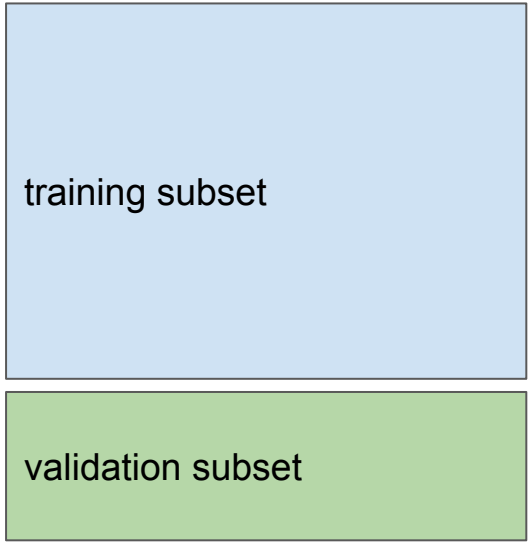
https://www.kaggle.com/c/otto-group-product-classification-challenge/discussion/14335

Метамодели



Blending

- Как и стекинг, тесно связан с понятием кросс-валидации
- Большой train-датасет делится на 2 части: training и validation
- Лучшая по валидации модель делает предсказания для [1] validation subset, [2] test set
- Строим несколько моделей и используем validation subset как train для метамодели

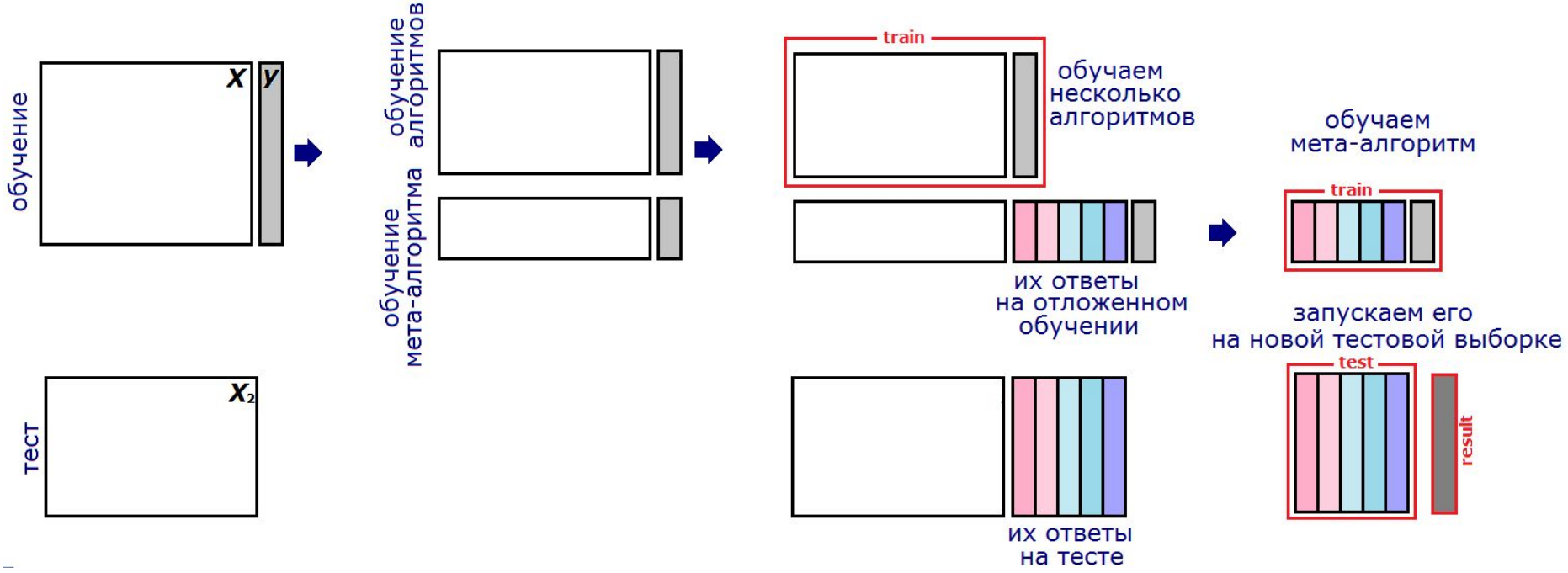


https://en.wikipedia.org/wiki/AdaBoost

Blending

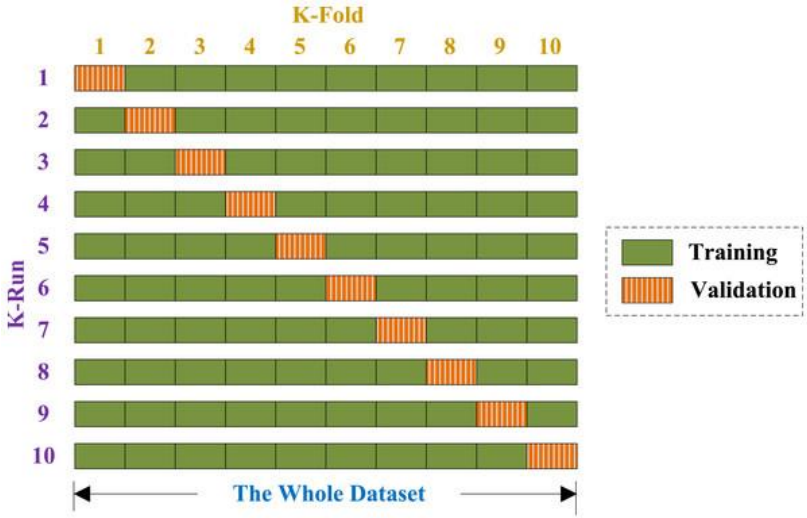
- +: Просто реализовать (можно даже в Spark RDD, используйте `pyspark.RDD.randomSplit`)
- +: Защищает от дата-лика
- +: Один валидационный сет можно использовать для любых классификаторов — удобно для командной работы
- —: validation subset не используется в обучении, этими данными придётся пожертвовать
- —: возможность переобучения на validation subset
- —: нужно проверять соответствие распределений в validation и training subsets

Blending


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

Stacking

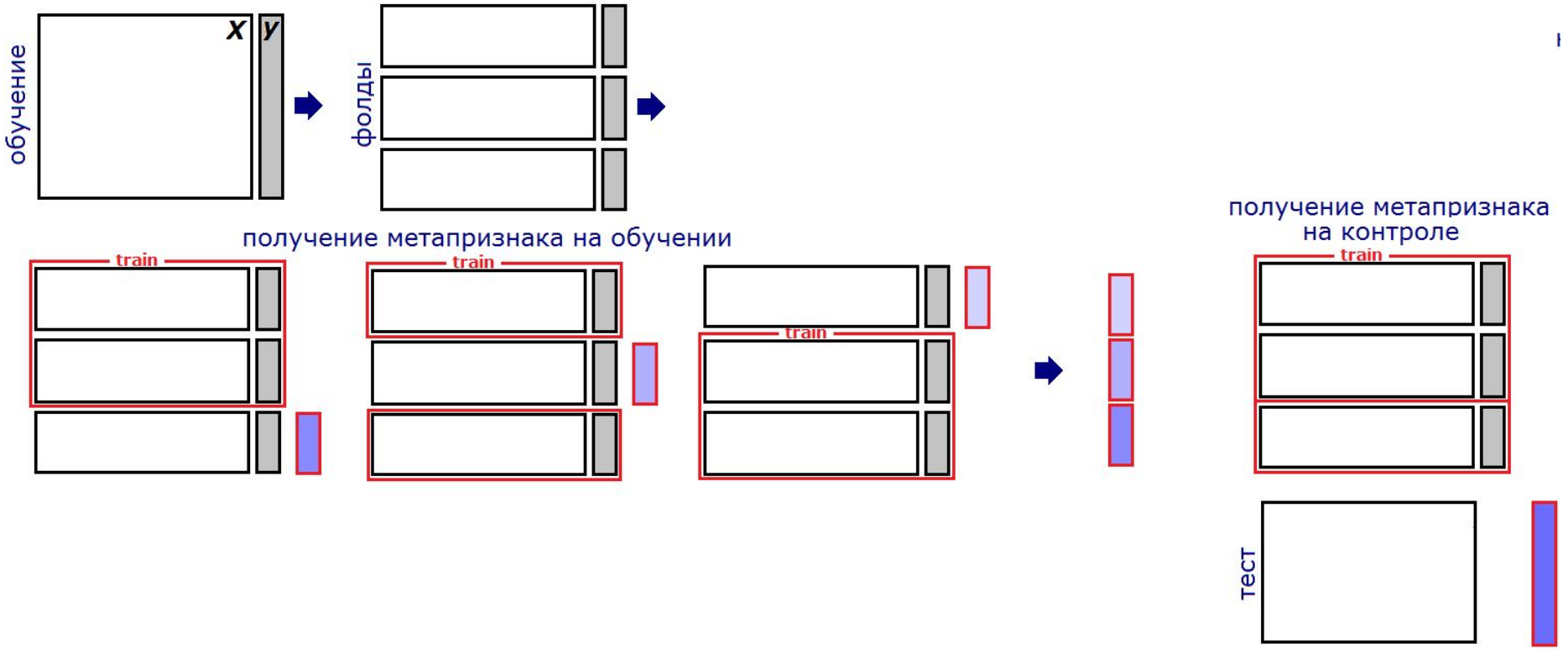
- Думайте в терминах k-fold cross-validation
- В k раз сложнее для расчёта, чем простой блендинг
- Задача — сделать предсказание всего train-сета, используя k-валидационных фолдов; каждый oof-классификатор предсказывает **весь** test-сет



Stacking

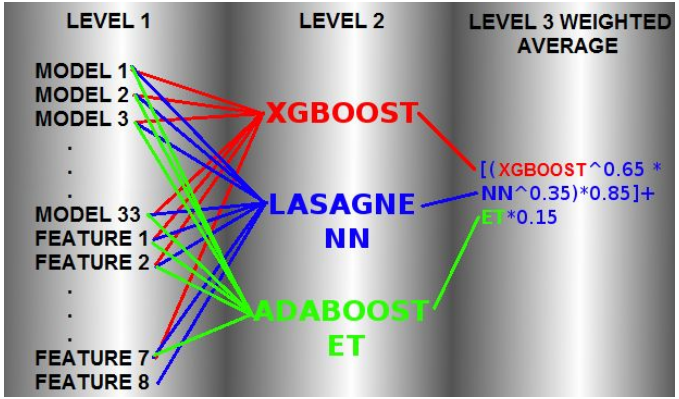
- +: Использует все данные из train-сета => обычно даёт лучше результат, чем блендинг
- +: Можно сделать StratifiedKFoldSplit
- +: Лучше защищён от переобучения на валидации
- —: значительно тяжеловеснее, чем блендинг
- —: у всех в команде должны быть одинаковые фолды train-сета
- —: обычно включает ещё и bagged folds, что ещё **на порядок** увеличивает сложность

Stacking


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

Уровни

- L1: уровень отдельных моделей и агрегированных фич => фичи для L2
- L2: метамодели, обучающиеся на выходах валидации / oof L1
- L3: метамодели, обучающиеся на выходах моделей L2



На вход метамоделям можно подавать фичи, полученные через unsupervised learning: k-means кластеры, pca components, t-sne вектора

http://blog.kaggle.com/2015/06/09/otto-product-classification-winners-interview-2nd-place-alexander-guschin/
https://www.kaggle.com/c/otto-group-product-classification-challenge/discussion/14295