



Ансамбли моделей

Занятие №5

Журавлёв Вадим

Показывать

Ближайшие две недели Весь семестр

Дисциплина

Основы машинного обучения

Тип события

Все типы

Группа

Все группы

30 сентября 18:00 — 21:00 среда Основы машинного обучения Смешанное занятие 1 Уточняется ML-11

7 октября 18:00 — 21:00 среда Основы машинного обучения Смешанное занятие 2 Уточняется ML-11

14 октября 18:00 — 21:00 среда Основы машинного обучения Смешанное занятие 3 Уточняется ML-11

21 октября 18:00 — 21:00 среда Основы машинного обучения Смешанное занятие 4 Уточняется ML-11

28 октября 18:00 — 21:00 среда Основы машинного обучения Смешанное занятие 5 Уточняется ML-11

3 ноября 18:00 — 21:00 вторник Основы машинного обучения Смешанное занятие 6 Уточняется ML-11

11 ноября 18:00 — 21:00 среда Основы машинного обучения Смешанное занятие 7 Уточняется ML-11

сентябрь

Пн	Вт	Ср	Чт	Пт	Сб	Вс
1	2	3	4	5	6	
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

октябрь

Пн	Вт	Ср	Чт	Пт	Сб	Вс
		1	2	3	4	
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

ноябрь

Пн	Вт	Ср	Чт	Пт	Сб	Вс

Приходя на лекцию



Содержание занятия

- 1. Ансамбли моделей**
- 2. Стэкинг**
- 3. Бэггинг и бустинг**
- 4. Random Forest**
- 5. Gradient Boosting**
- 6. Автоматический подбор параметров модели**

О значимости ансамблей

Лучшие алгоритмы машинного обучения по точности:

- Градиентный бустинг для классических задач
- Искусственные нейронные сети для изображений, видео, звука

В соревнованиях kaggle всегда* побеждают ансамбли

Ансамбли моделей

Коллективное принятие решений как правило превосходит по качеству индивидуальное принятие решений



Простое голосование



Классификация: класс определяется большинством голосов или усреднением скоров

Регрессия: среднее значение

Взвешенное голосование



Классификация: класс определяется большинством голосов с учетом веса, или усреднением скоров с учетом веса

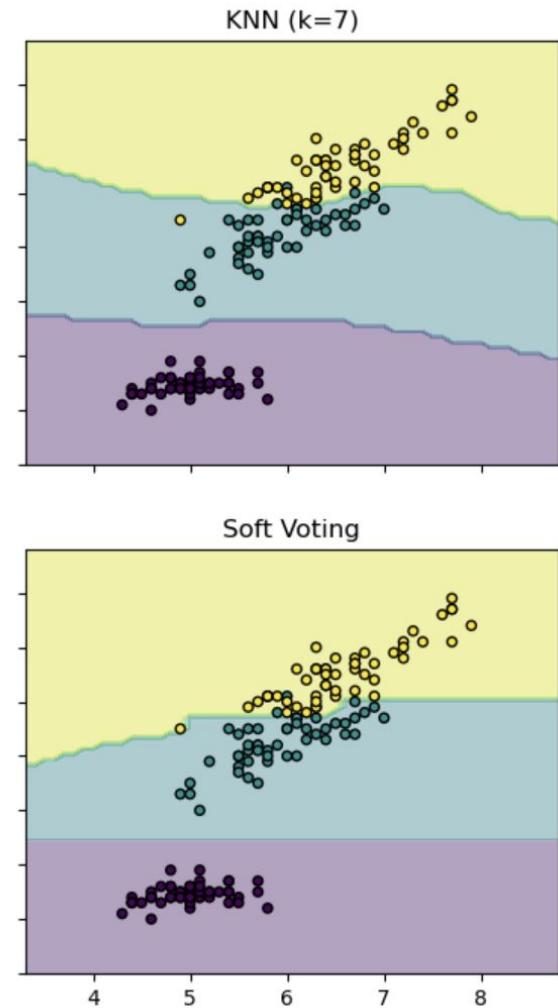
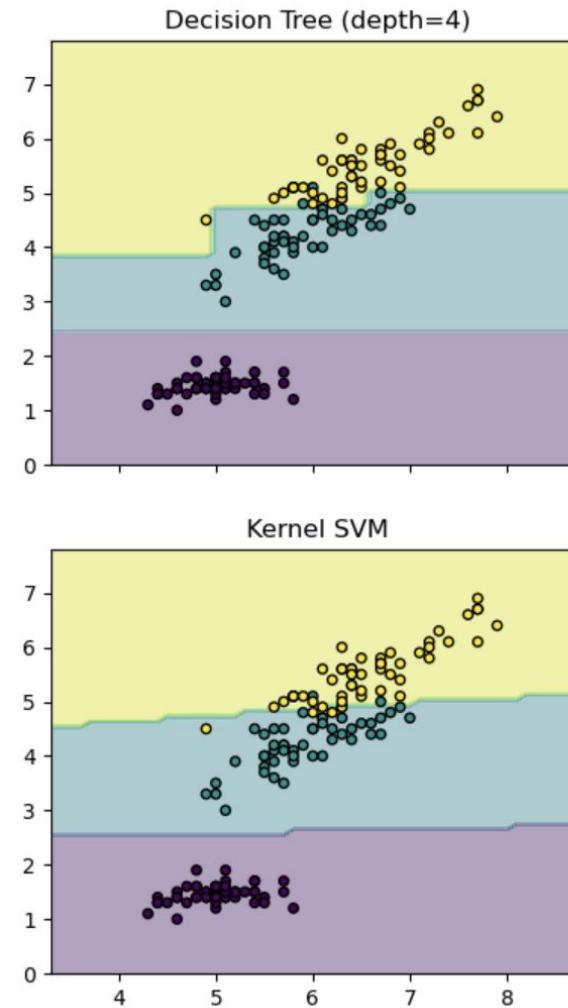
Регрессия: среднее взвешенное значение

Пример голосования

Датасет - [Iris Dataset](#)

Soft Voting - голосование трёх моделей с разным весом (в данном случае веса - [2, 1, 2])

```
from sklearn.ensemble import  
VotingClassifier  
  
clf1 = SVC()  
clf2 = KNeighborsClassifier(n_neighbors=7)  
clf3 = DecisionTreeClassifier(max_depth=4)  
  
eclf = VotingClassifier(estimators=[('dt', clf1), ('knn',  
clf2), ('svc', clf3)], voting='soft', weights=[2, 1, 2])
```



Недостатки голосования

1. Голосование не учитывает особенностей поведения отдельных моделей
2. Голосование по сути является простой моделью

Вопрос

Можно ли использовать ансамбли для линейных моделей?



Стэкинг

Идея:

Построим модель, которая будет предсказывать правильный
ответ на основе предсказаний других моделей

Блендинг

Простейшая схема стекинга – блендинг

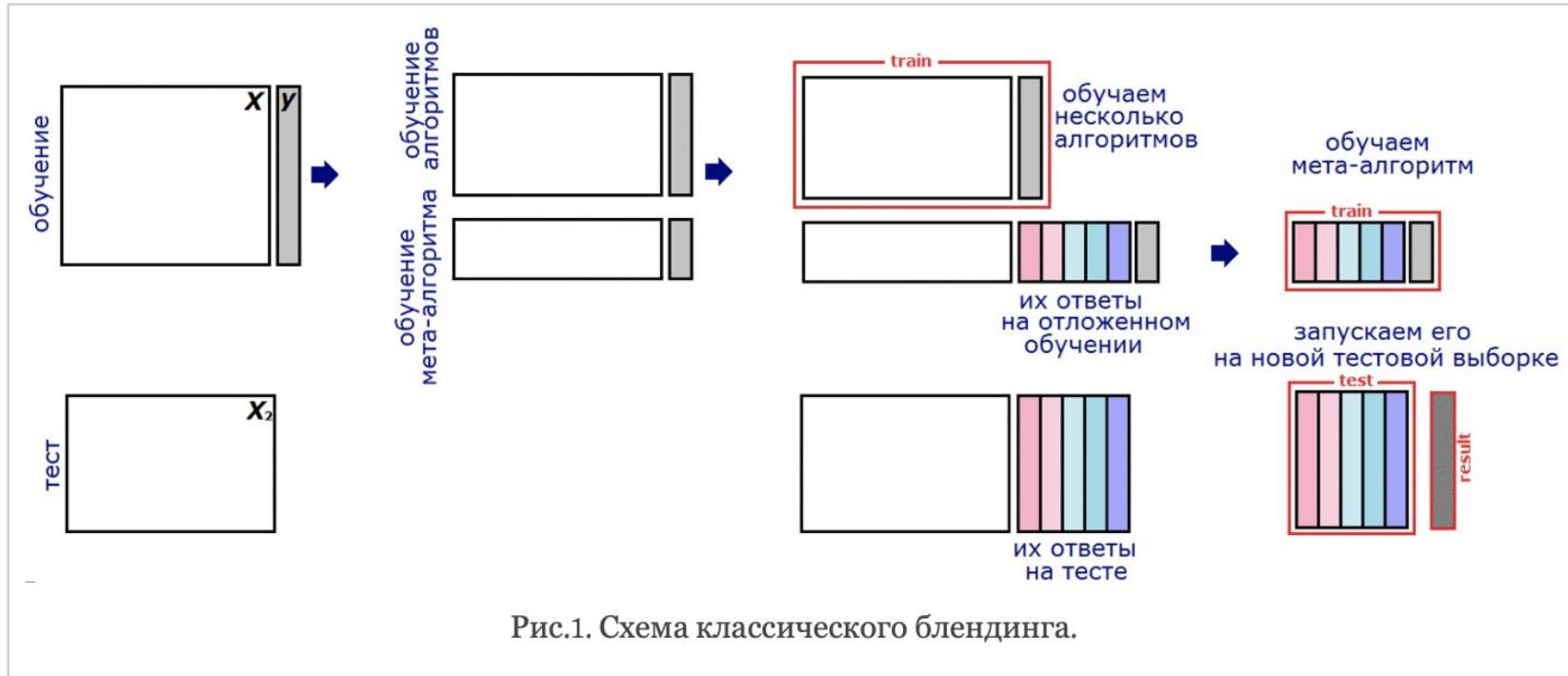


Рис.1. Схема классического блендинга.

Недостаток блендинга

Самый большой недостаток классического блендинга — деление обучающей выборки.

Как бороться?

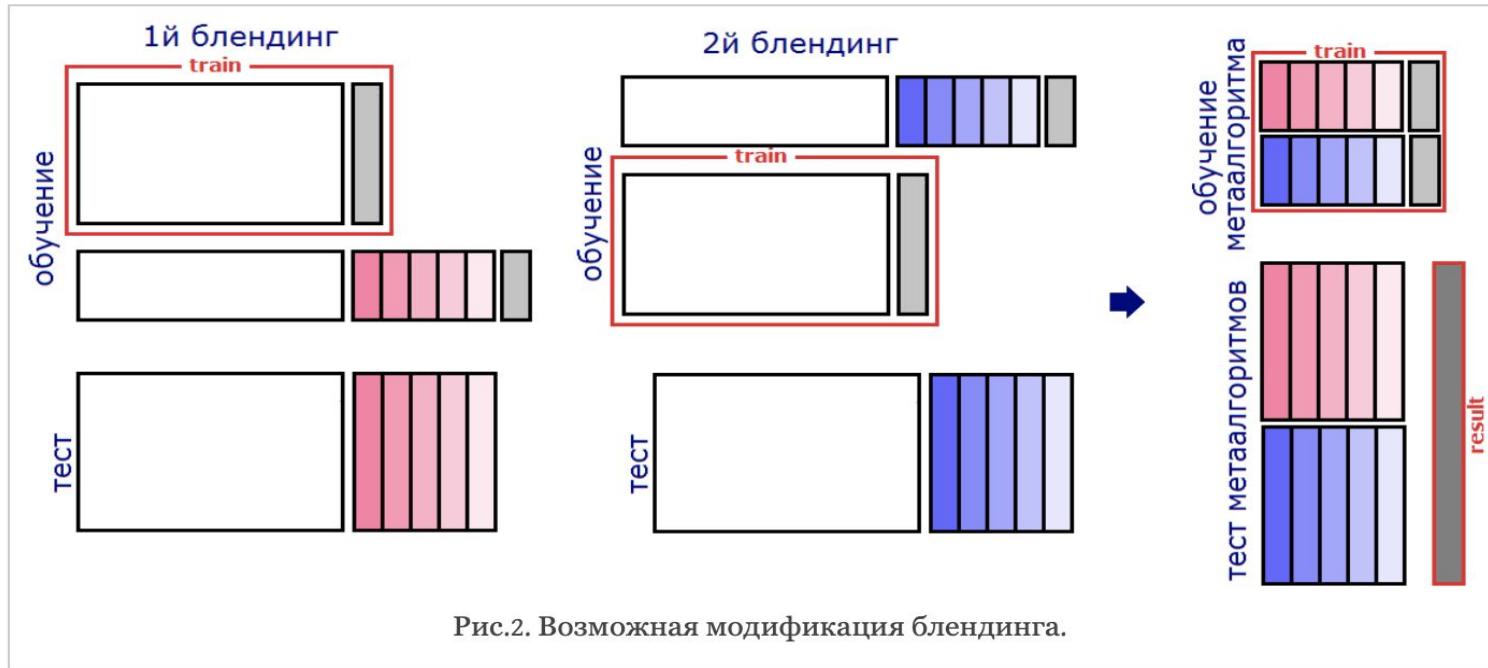
Недостаток блендинга

Самый большой недостаток классического блендинга — деление обучающей выборки.

Как бороться?

Для повышения качества надо усреднить несколько блендингов с разными разбиениями обучения

Блендинг



На практике такая схема блендинга сложнее в реализации и более медленная, а по качеству может не превосходить обычного усреднения.

Стэкинг

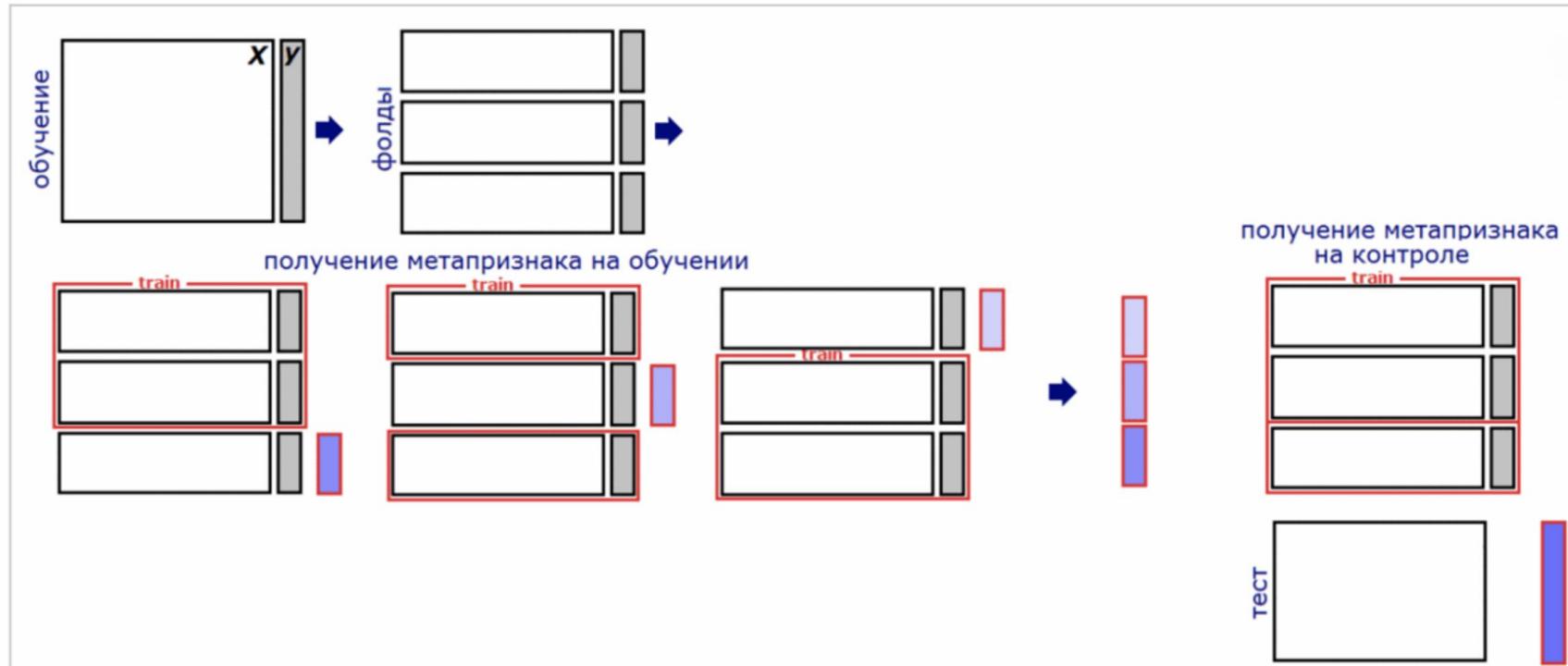


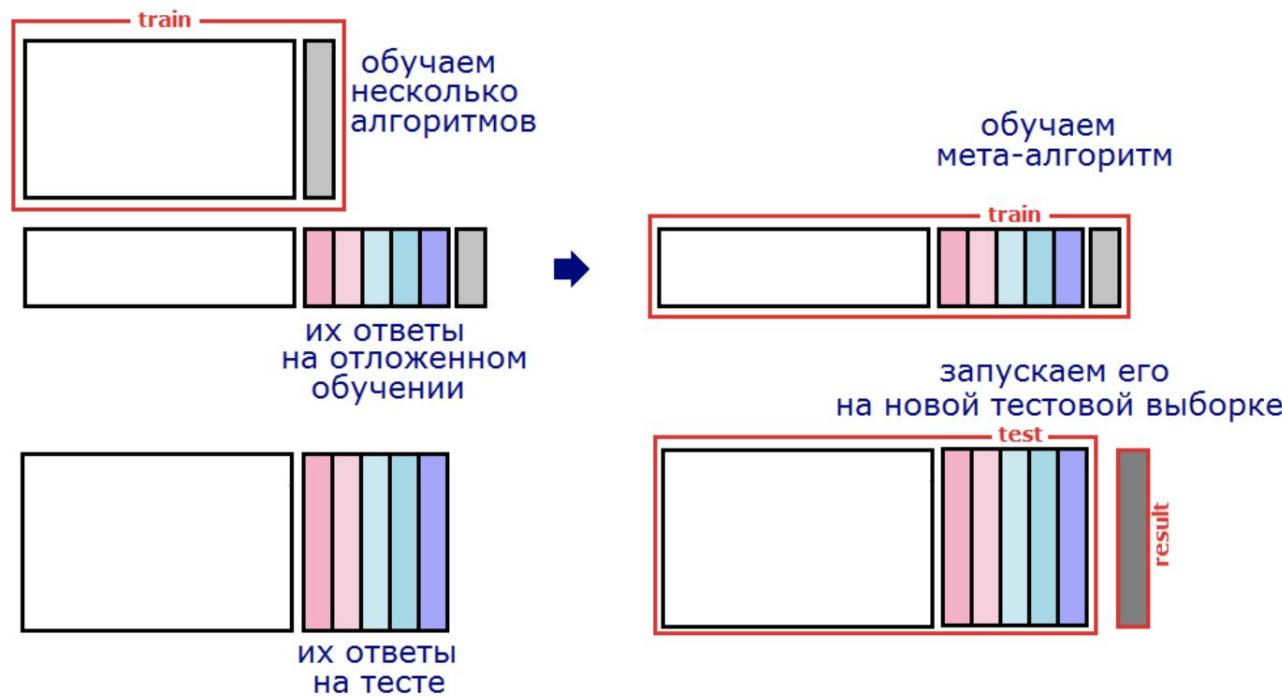
Рис.3. Получение метапризнака в классическом стекинге.

Кросс-валидационное предсказание будем называть
метапризнаком.

Стэкинг

Стэкинг можно делать как на наборе метапризнаков, так и на наборе метапризнаков + набор исходных признаков.

Стэкинг может быть многоуровневым.



Бэггинг и Бустинг

Идея бэггинга:

1. Построим много слегка различающихся моделей
2. Методом усреднения выберем итоговый ответ

Идея бустинга:

Каждая следующая модель в ансамбле пытается предсказать ошибку всех предыдущих моделей ансамбля

Бутстрэп

Бутстрэп (bootstrap) – метод исследования распределения статистик вероятностных распределений, основанный на многократной генерации псевдовыборок на базе имеющейся выборки.

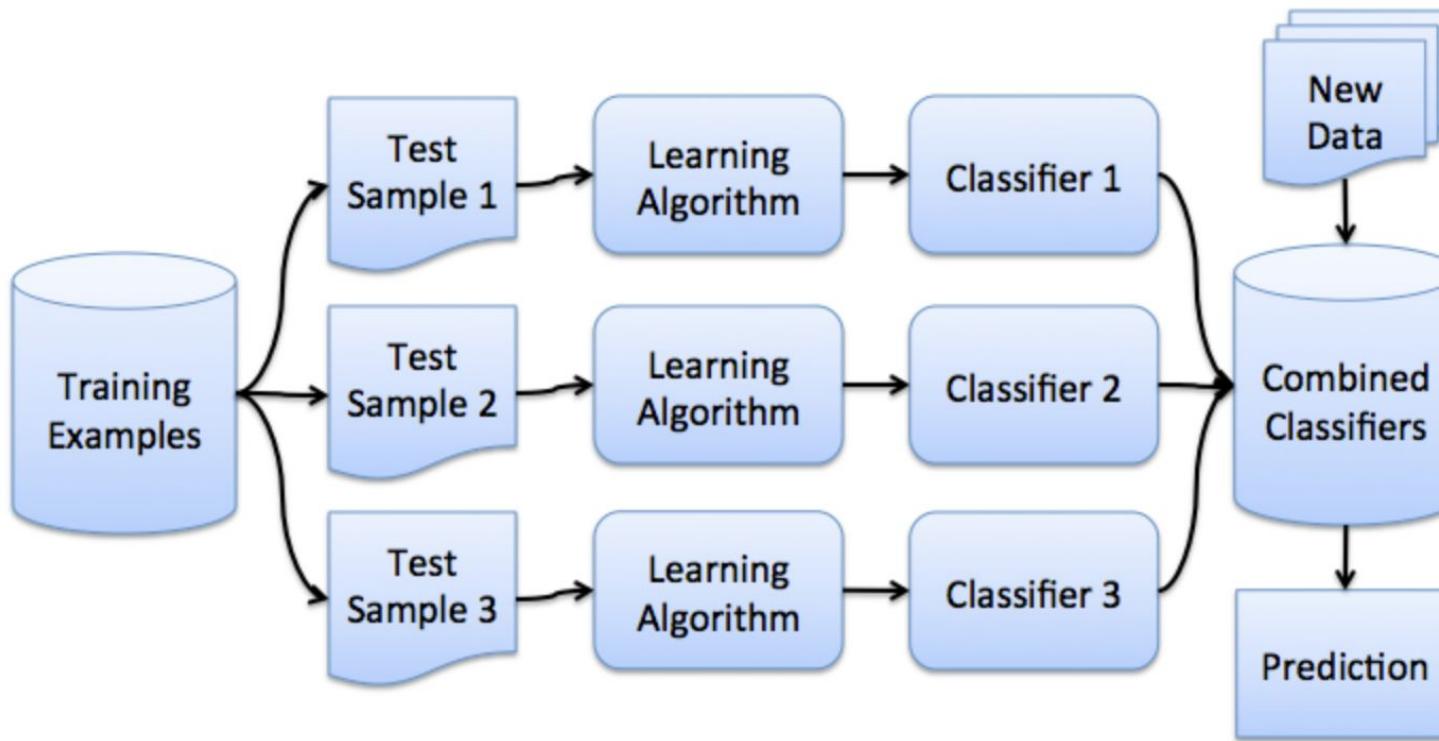
1. Из исходной выборки генерируем псевдовыборки методом случайного выбора с возвращением.
2. На псевдовыборках считаем целевую статистику.
3. Анализируем распределение целевой статистики на псевдовыборках.

Бэггинг

Bagging – Bootstrap aggregating (Leo Breiman, 1994)

- Из Train генерим методом случайного выбора сэмплов с возвращением $\text{Train}'_1 \dots \text{Train}'_N$
- На каждом Train' строим модель
- Итоговое предсказание получаем усреднением предсказаний всех моделей или простым голосованием

БЭГГИНГ



БЭГГИНГ

Рассмотрим задачу регрессии с базовыми алгоритмами $b_1(x), \dots, b_n(x)$

$$\varepsilon_i(x) = b_i(x) - y(x), i = 1, \dots, n$$

$$E_x(b_i(x) - y(x))^2 = E_x \varepsilon_i^2(x)$$

$$E_1 = \frac{1}{n} E_x \sum_{i=1}^n \varepsilon_i^2(x)$$

БЭГГИНГ

Предположим, что ошибки несмещены и некоррелированы:

Построим новую функцию регрессии

Найдем её среднеквадратичную ошибку

$$\begin{aligned} E_x \varepsilon_i(x) &= 0, \\ E_x \varepsilon_i(x) \varepsilon_j(x) &= 0, i \neq j \end{aligned}$$

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

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

Метод случайных подпространств

RSM – Random Subspace Method или feature bagging

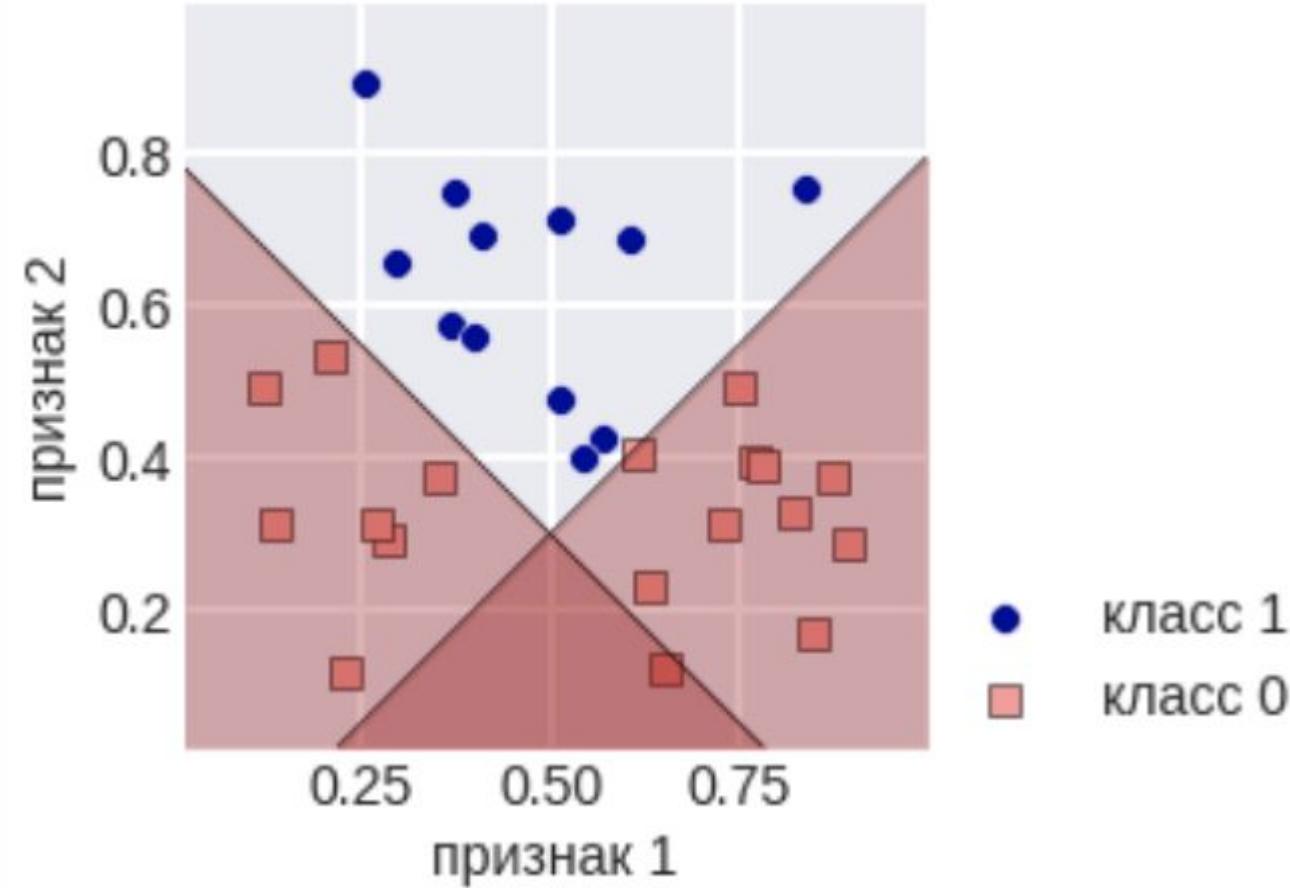
- Из Train генерим методом случайного выбора признаков без возвращения $\text{Train}'_1 \dots \text{Train}'_N$
- На каждом Train' строим модель
- Итоговое предсказание получаем усреднением предсказаний всех моделей

Вернёмся к вопросу

Используются ли ансамбли для линейных моделей?



Вопрос



#026

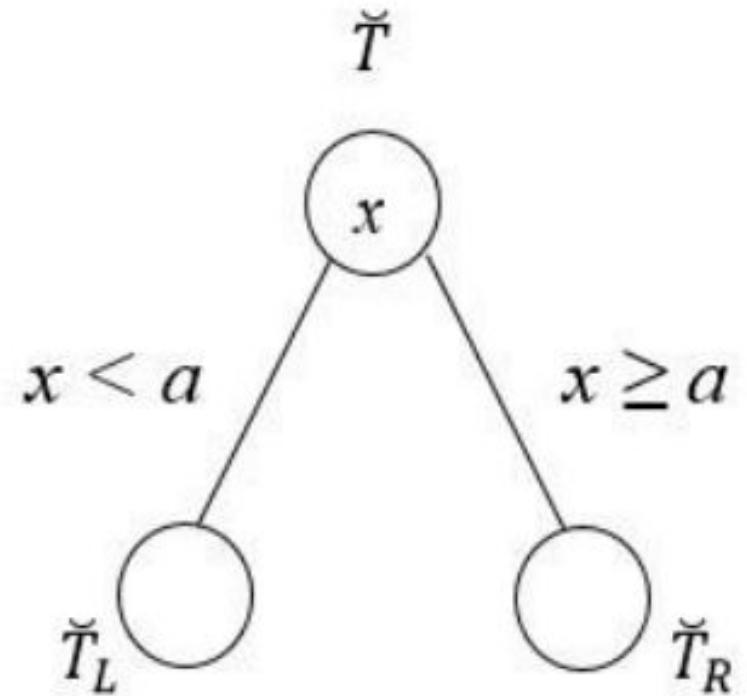
Немного регрессионных деревьев



#027

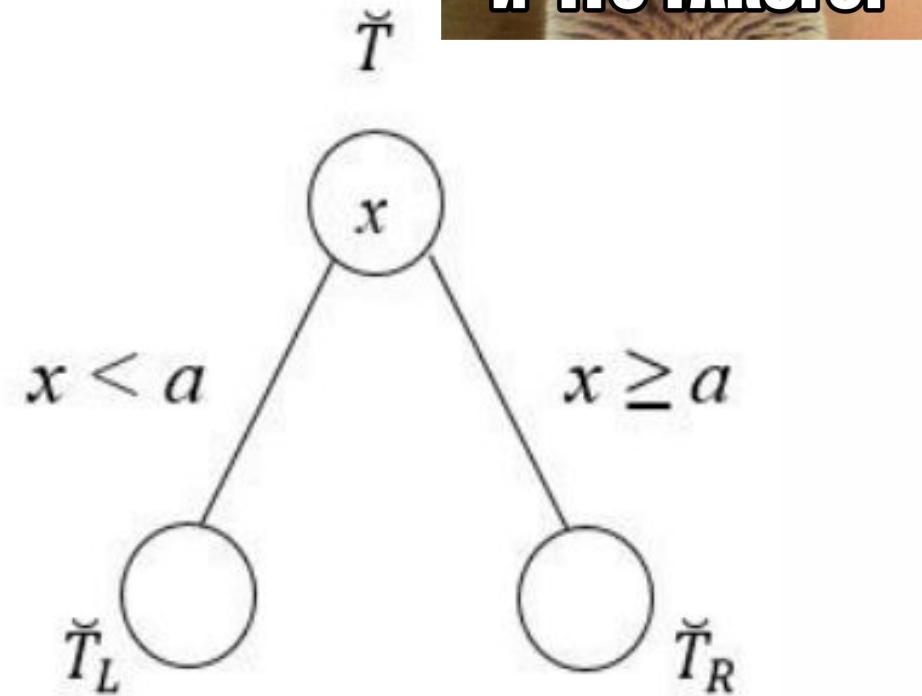
Немного о регрессионных деревьях

$$\check{T} = \{S_{i_1}, \dots, S_{i_u}\}$$
$$\check{T}_L(x, a) = \{S_1^L, \dots, S_p^L\} \quad \check{T}_R(x, a) = \{S_1^R, \dots, S_q^R\}$$



Немного о регрессионных деревьях

$$\check{T} = \{S_{i_1}, \dots, S_{i_u}\}$$
$$\check{T}_L(x, a) = \{S_1^L, \dots, S_p^L\}$$
$$\check{T}_R(x, a) = \{S_1^R, \dots, S_q^R\}$$



Регрессионные деревья

$$\bar{y}_{\check{T}} = \frac{1}{u} \sum_{t=1}^u y_{i_t};$$

$$\text{Variance} = \frac{1}{u} \sum_{t=1}^u (y_{i_t}^2) - \left[\frac{1}{u} \sum_{t=1}^u (y_{i_t}) \right]^2;$$

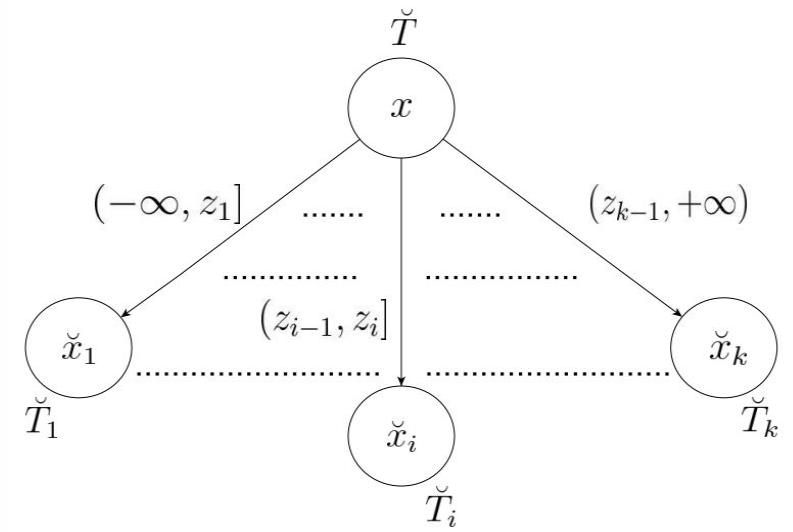
$$\text{SE}(x) = \frac{1}{u} \left\{ \sum_{i=1}^p (y_i^L - \bar{y}_{\check{T}_L})^2 + \sum_{j=1}^q (y_j^R - \bar{y}_{\check{T}_R})^2 \right\};$$

$$C(x) = \text{Variance} - \text{SE}(x).$$

Оптимальным считается разбиение с максимальным значением величины C(x).

k-арные регрессионные деревья

Как будет выглядеть критерий ветвления
для k-арного дерева?

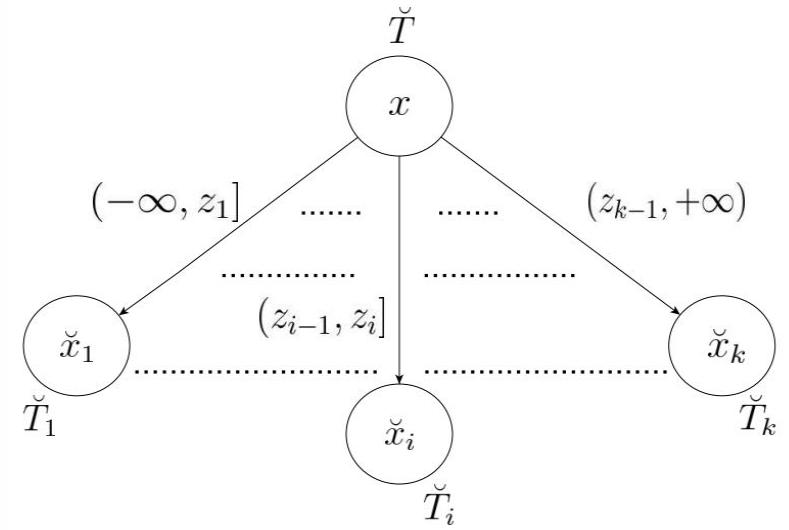


k-арные регрессионные деревья

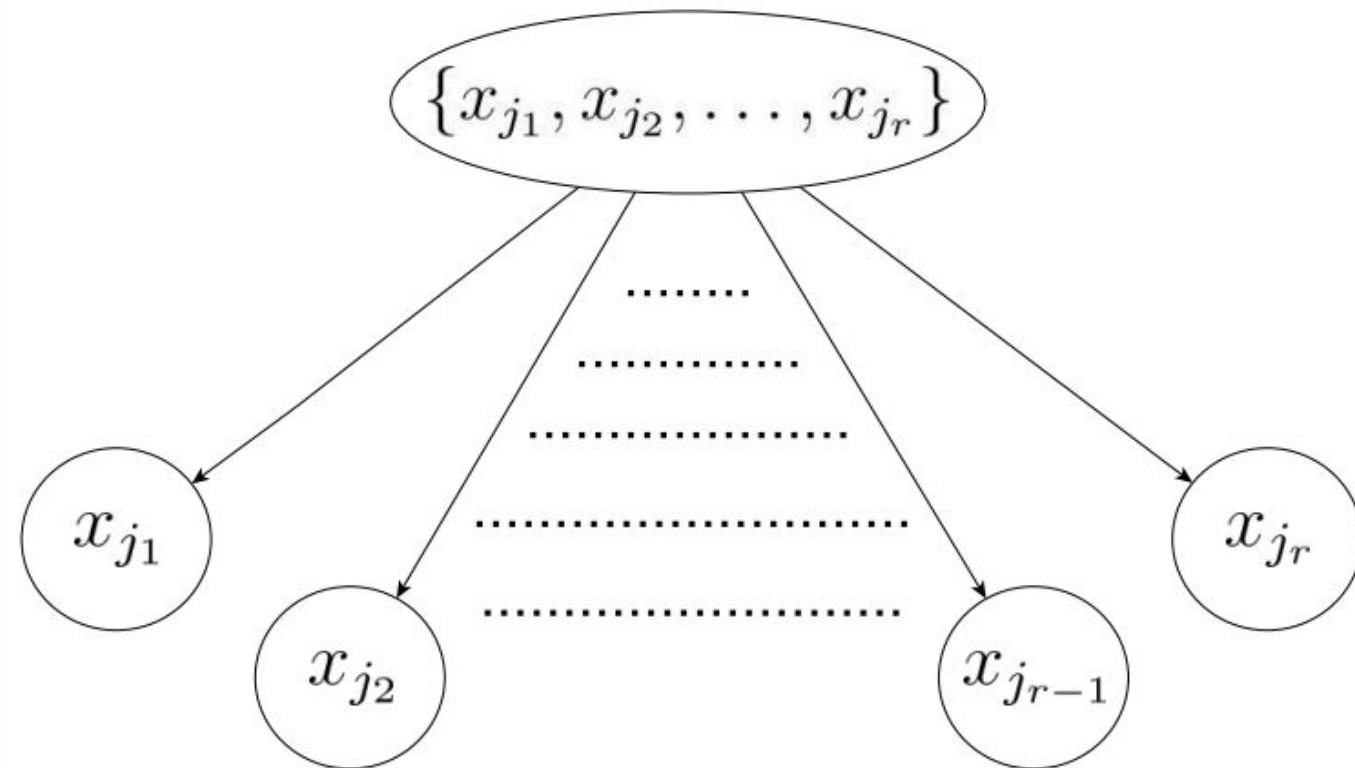
Как будет выглядеть критерий ветвления для k-арного дерева?

$$SE(x, k) = \frac{1}{u_i} \left\{ \sum_{S_t^i \in \check{T}_{i_1}} (y_t^i - \bar{y}_{\check{T}_{i_1}})^2 + \dots + \sum_{S_t^i \in \check{T}_{i_k}} (y_t^i - \bar{y}_{\check{T}_{i_k}})^2 \right\}$$

$$C(k, x) = Variance - SE(x, k).$$



Полные k-арные регрессионные деревья



Полные k-арные регрессионные деревья



#034

Ссылка:

[Построение и исследование полных решающих деревьев ...jmlda.org › papers › doc](https://jmlda.org/papers/doc)

Random Forest

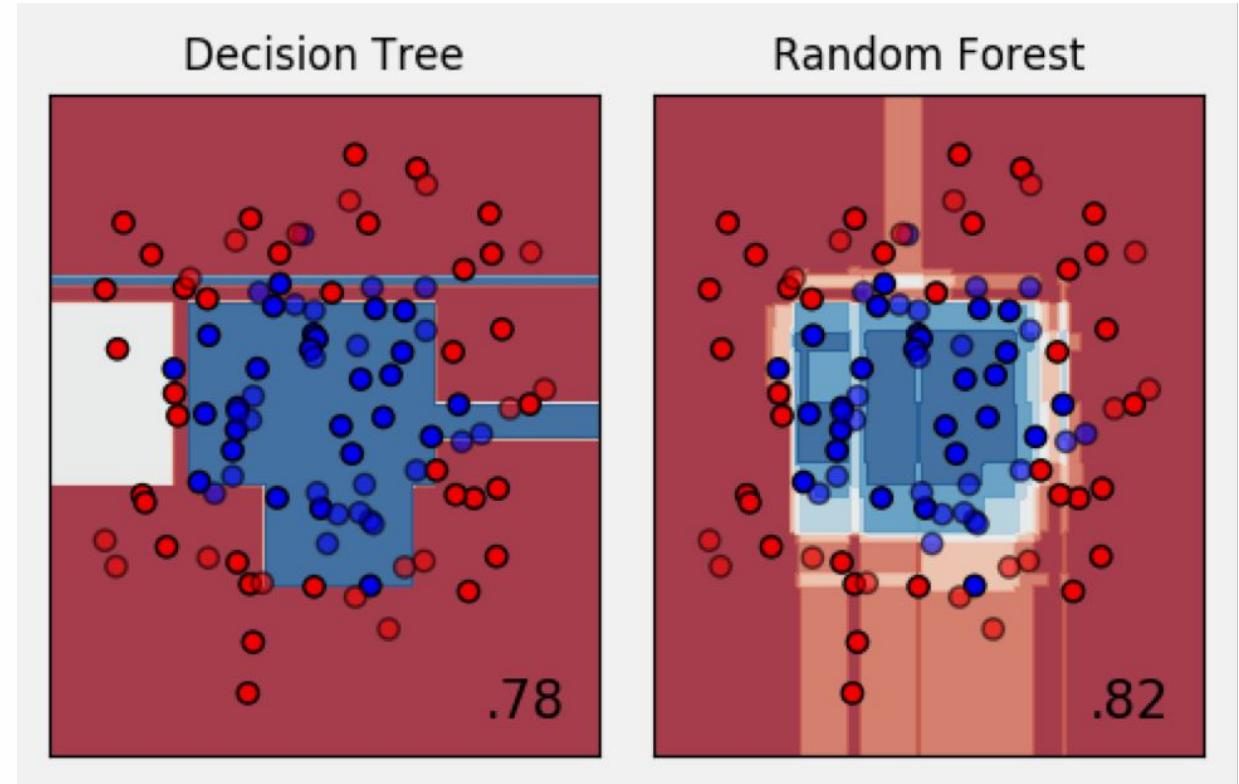
Алгоритм:

1. Выполняем N раз:
 - 1.1. Бутстрэп самплов
 - 1.2. Случайное подпространство признаков
 - 1.3. Построение дерева решений
2. Выбираем ответ модели методом усреднения предсказаний или простого голосования



Random Forest

Random Forest выдает
качество лучше, чем
единичное решающее дерево



Параметры Random Forest

```
RandomForestRegressor(n_estimators, criterion, max_depth,  
min_samples_split, min_samples_leaf, min_weight_fraction_leaf,  
max_features, max_leaf_nodes, min_impurity_decrease,  
min_impurity_split, bootstrap, oob_score, n_jobs, random_state,  
verbose, warm_start)
```

Параметры функции потерь

Параметры ансамбля

Параметры дерева

Параметры технические

Random Forest

Важные параметры:

- `n_estimators` — число деревьев в лесу
- `max_features` — число признаков, по которым ищется разбиение
- `min_samples_leaf` — минимальное число объектов в листе
- `max_depth` — максимальная глубина дерева

Random Forest: Оценка важности признаков

Чем выше в среднем признак в дереве решений, тем он важнее в данной задаче



Random Forest

Плюсы:

1. Алгоритм прост
2. Не чувствителен к выбросам в данных
3. Не переобучается
4. Хорошо параллелизируется
5. Не требует сложной настройки параметров
6. Не требует нормализации данных

Random Forest

Минусы:

1. Модели не интерпретируемые
2. Работает хуже линейных моделей, когда есть разреженные признаки (например, тексты)
3. Большой размер получающихся моделей

Градиентный бустинг

Идея:

1. Представляем итоговую модель $f(x)$ как сумму слабых моделей $h(x)$ (обычно решающие деревья малой глубины).
2. Пусть задана дифференцируемая функция потерь $L(y, f(x))$
3. На каждом шаге мы ищем модель $h(x)$, которая бы аппроксимировала вектор антиградиента L

Градиентный бустинг

#043

1. Инициализировать GBM константным значением $\hat{f}(x) = \hat{f}_0, \hat{f}_0 = \gamma, \gamma \in \mathbb{R}$

$$\hat{f}_0 = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma)$$

2. Для каждой итерации $t = 1, \dots, M$ повторять:

1. Посчитать псевдо-остатки r_t

$$r_{it} = - \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x)=\hat{f}(x)}, \quad \text{for } i = 1, \dots, n$$

2. Построить новый базовый алгоритм $h_t(x)$ как регрессию на псевдо-остатках

$$\{(x_i, r_{it})\}_{i=1, \dots, n}$$

3. Найти оптимальный коэффициент ρ_t при $h_t(x)$ относительно исходной функции потерь

$$\rho_t = \arg \min_{\rho} \sum_{i=1}^n L(y_i, \hat{f}(x_i) + \rho \cdot h(x_i, \theta))$$

4. Сохранить $\hat{f}_t(x) = \rho_t \cdot h_t(x)$

5. Обновить текущее приближение $\hat{f}(x)$

$$\hat{f}(x) \leftarrow \hat{f}(x) + \hat{f}_t(x) = \sum_{i=0}^t \hat{f}_i(x)$$

3. Скомпоновать итоговую GBM модель $\hat{f}(x)$

$$\hat{f}(x) = \sum_{i=0}^M \hat{f}_i(x)$$

Градиентный бустинг

Построили алгоритм $a(x)$, построим алгоритм $b(x)$ такой, что

$$a(x_i) + b(x_i) = y_i, \quad i \in \{1, 2, \dots, m\},$$

Алгоритм $b(x)$ поправляет ошибки алгоритма $a(x)$ на невязку: $\varepsilon_i = y_i - a(x_i)$

$$\sum_{i=1}^m L(y_i - a(x_i), b(x_i)) \rightarrow \min \neq \sum_{i=1}^m L(y_i, a(x_i) + b(x_i)) \rightarrow \min$$

Градиентный бустинг

$$F = \sum_{i=1}^m L(y_i, a(x_i) + b_i) \rightarrow \min_{(b_1, \dots, b_m)}$$

Функция многих переменных максимально убывает

Градиентный бустинг

$$F(b_1, \dots, b_m) = F = \sum_{i=1}^m L(y_i, a(x_i) + b_i) \rightarrow \min_{(b_1, \dots, b_m)}$$

Функция многих переменных максимально убывает в направлении своего антиградиента:

$$-(L'(y_1, a(x_1)), \dots, L'(y_m, a(x_m)))$$

Градиентный бустинг

Выгодно считать

$$b_i = -L'(y_i, a(x_i)), \quad i \in \{1, 2, \dots, m\}$$

а это и есть наши ответы алгоритма b. Получается его следует настраивать на обучающей выборке:

$$(x_i, -L'(y_i, a(x_i)))_{i=1}^m$$

Градиентный бустинг

```
GradientBoostingClassifier(loss, learning_rate, n_estimators,  
subsample, criterion, min_samples_split, min_samples_leaf,  
min_weight_fraction_leaf, max_depth, min_impurity_decrease,  
min_impurity_split, init, random_state, max_features, verbose,  
max_leaf_nodes, warm_start, presort, validation_fraction,  
n_iter_no_change, tol)
```

Параметры функции потерь

Параметры ансамбля

Параметры дерева

Параметры технические

Недостатки градиентного бустинга

- **Проблема переобучения градиентного бустинга**

По мере увеличения числа деревьев ошибка на обучающей выборке постепенно уходит в 0. Ошибка на контрольной выборке существенно больше ошибки на обучающей выборке, достигает минимума примерно на 10 итерации, а затем начинает опять возрастать. Имеет место переобучение.

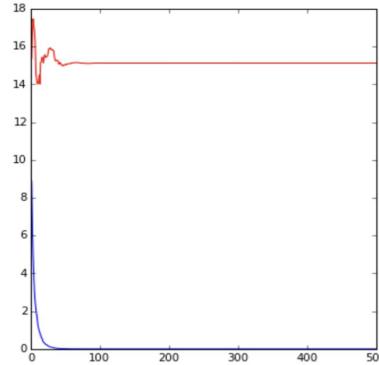
- **Сокращение размера шага**

Чтобы решить эту проблему, нужно «не доверять» направлению, которое построил базовый алгоритм и лишь чуть-чуть смещаться в сторону этого вектора:

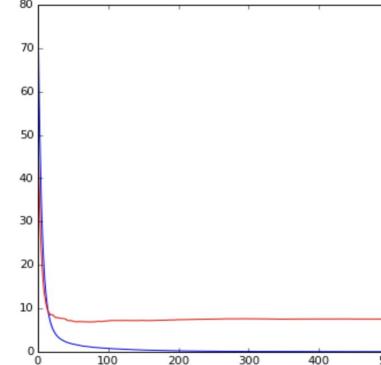
$$a_N(x) = a_{N-1}(x) + \eta b_N(x),$$

где $\eta \in (0, 1]$ — длина шага. Это обеспечивает очень аккуратное движение в пространстве, что делает возможным нахождение локального минимума.

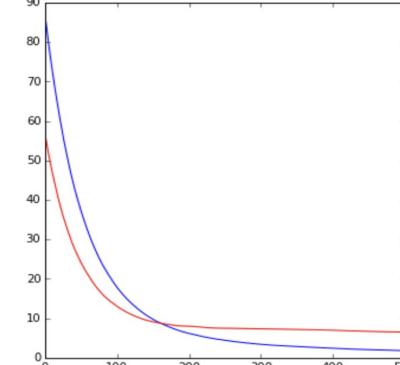
Недостатки градиентного бустинга



(a) Случай $\eta = 1$.



(b) Случай $\eta = 0.1$.



(c) Случай $\eta = 0.01$.

Как видно по графикам, при $\eta = 0.1$ качество на контрольной выборке уже существенно лучше, то есть в некотором смысле удалось побороть переобучение. При еще меньшей длине шага $\eta = 0.01$ градиентному бустингу требуется существенно больше итераций, чтобы достичь чуть-чуть большего качества. Таким образом:

- Чем меньше размер шага, тем больше нужно базовых алгоритмов, чтобы достичь хорошего качества, и тем больше времени занимает процесс.
- Чем меньше размер шага, тем лучшего качества можно достичь.

Другими словами, приходится выбирать: или быстро получить достаточно хорошее качество, или получить качество чуть-чуть лучше за большее время.

Недостатки бустинга

- Идея бустинга обычно плохо применима к построению композиции из достаточно сложных и мощных алгоритмов. Построение такой композиции занимает очень много времени, а качество существенно не увеличивается.
- Результаты работы бустинга сложно интерпретируемы, особенно если в композицию входят десятки алгоритмов.

Преимущества

- Хорошая обобщающая способность. В реальных задачах (не всегда, но часто) удаётся строить композиции, превосходящие по качеству базовые алгоритмы. Обобщающая способность может улучшаться (в некоторых задачах) по мере увеличения числа базовых алгоритмов.
- Простота реализации.
- Собственные накладные расходы бустинга невелики. Время построения композиции практически полностью определяется временем обучения базовых алгоритмов.
- Возможность идентифицировать объекты, являющиеся шумовыми выбросами.

Вопрос на засыпку

Если бы вы могли воспользоваться многоядерным процессором, вы бы предпочли алгоритм бустинга над деревьями случайному лесу? Почему?

Открытые реализации градиентного бустинга

dmlc
XGBoost



Yandex
CatBoost

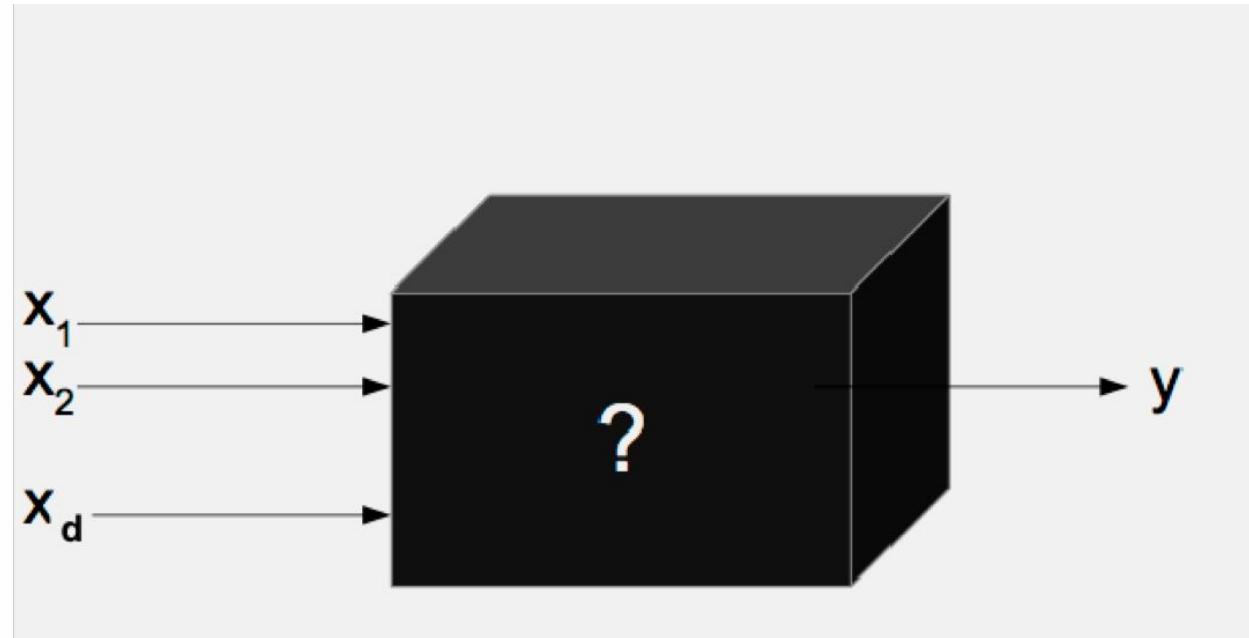


#054

Автоматический подбор гиперпараметров моделей

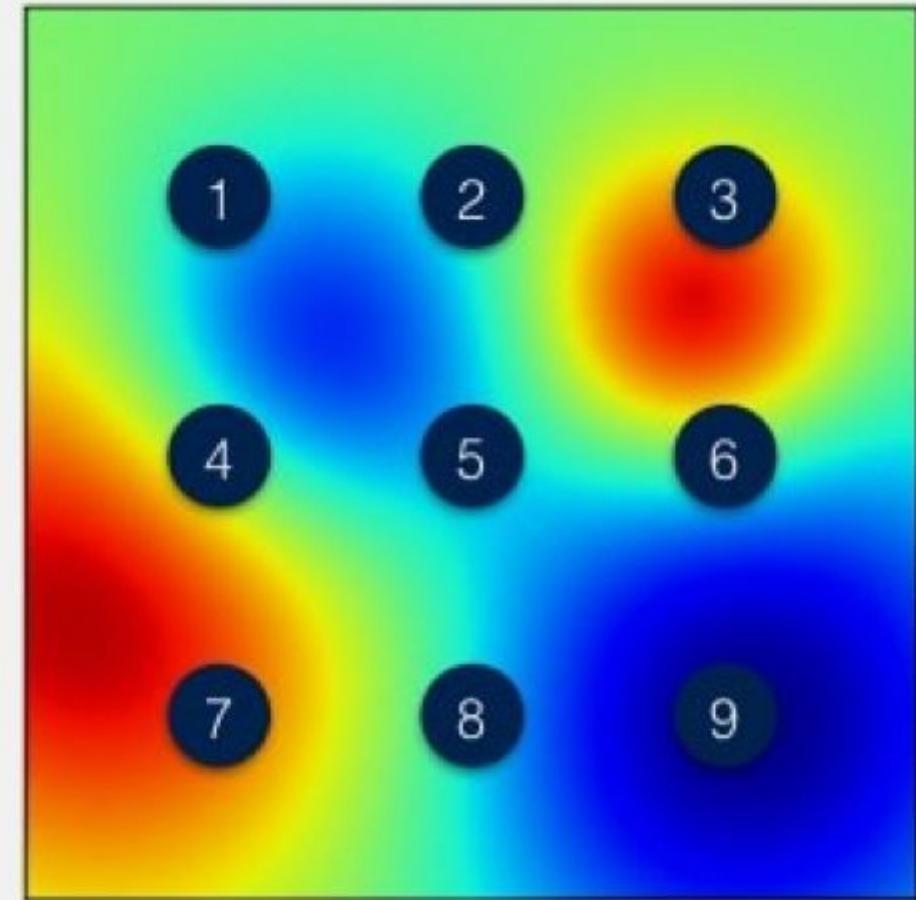
Модель(параметры) -> качество

Задача - максимизировать качество



Grid Search

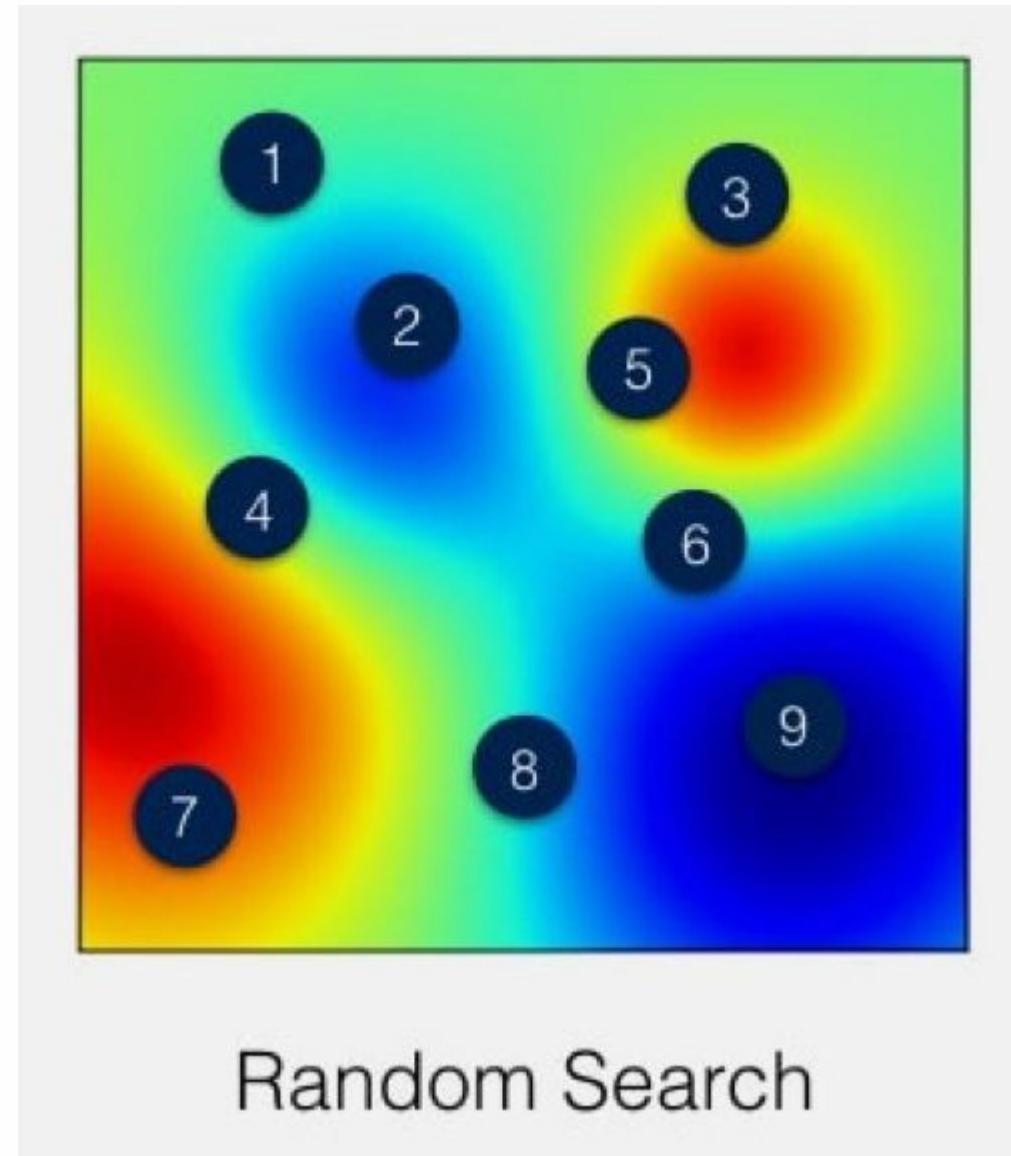
1. Перебираем параметры модели по решетке
2. Выбираем параметры, которые дают самое высокое качество



Grid Search

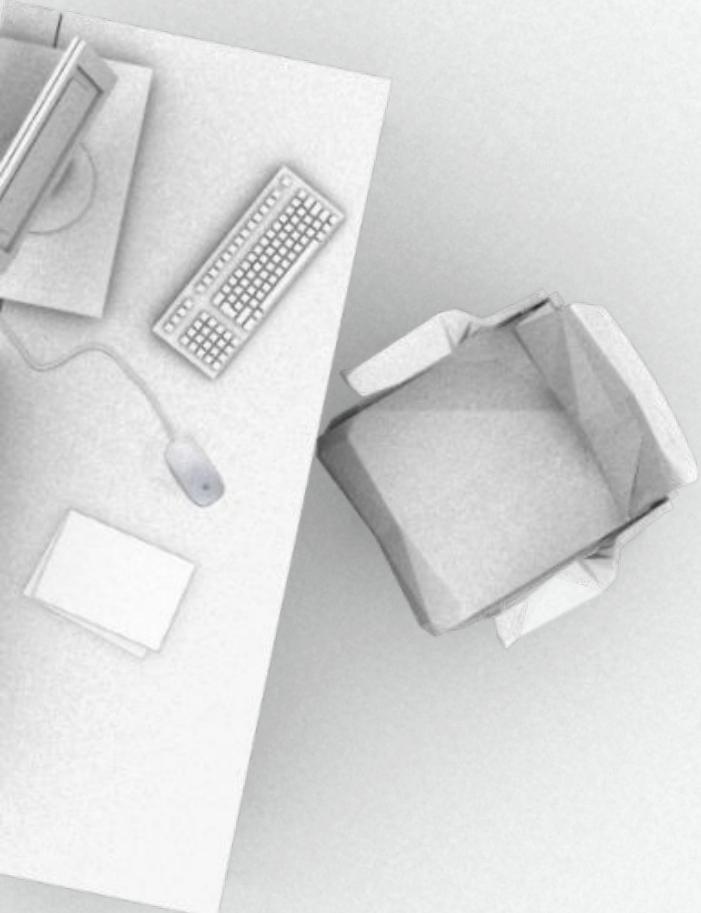
Random Search

1. Сэмплируем параметры модели
2. Выбираем параметры, которые дают самое высокое качество



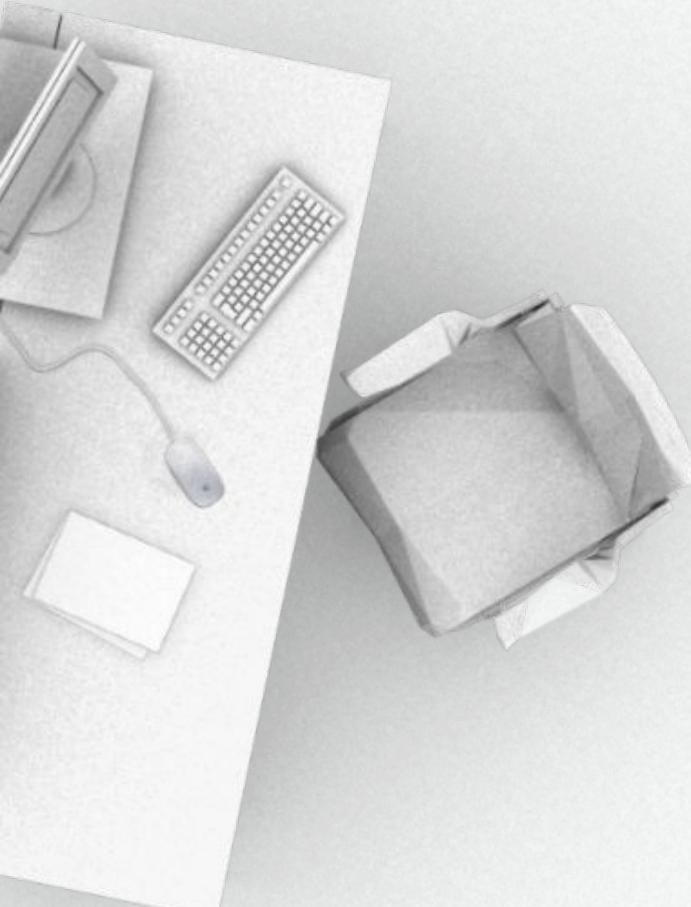
Инструменты для оптимизации гиперпараметров

- sklearn
- Hyperopt
- BayesianOptimization
- Hypropy
- Optunity
- Optuna



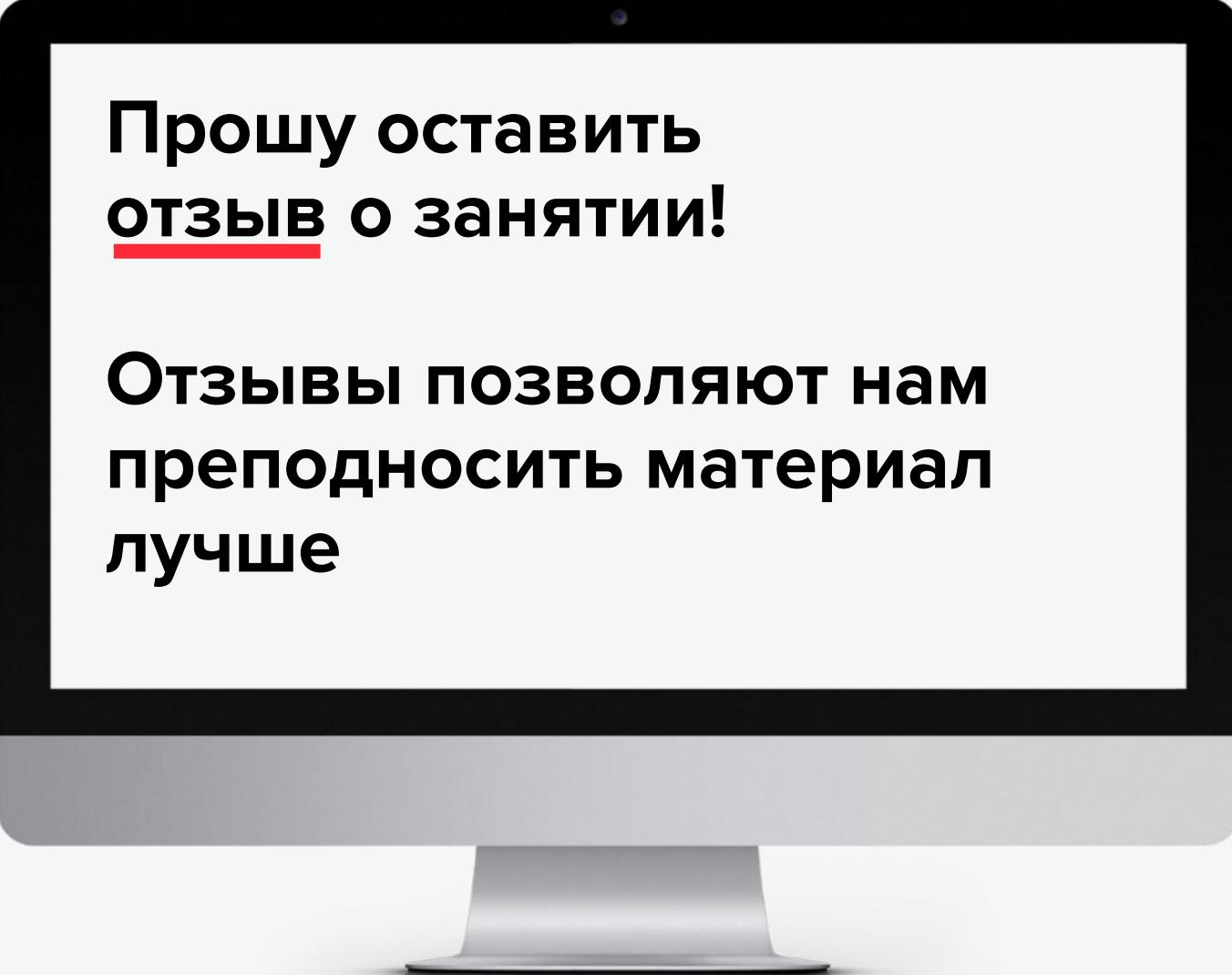
Домашнее задание: соревнование

- Сделать сабмит решения
- Выложить решение на github.com (можно залить jupyter notebook на портале)
- Прислать свой ник kaggle



Конкурс

<https://www.kaggle.com/c/parkml2020/overview>



**Прошу оставить
отзыв о занятии!**

**Отзывы позволяют нам
преподносить материал
лучше**

**СПАСИБО
ЗА ВНИМАНИЕ**

