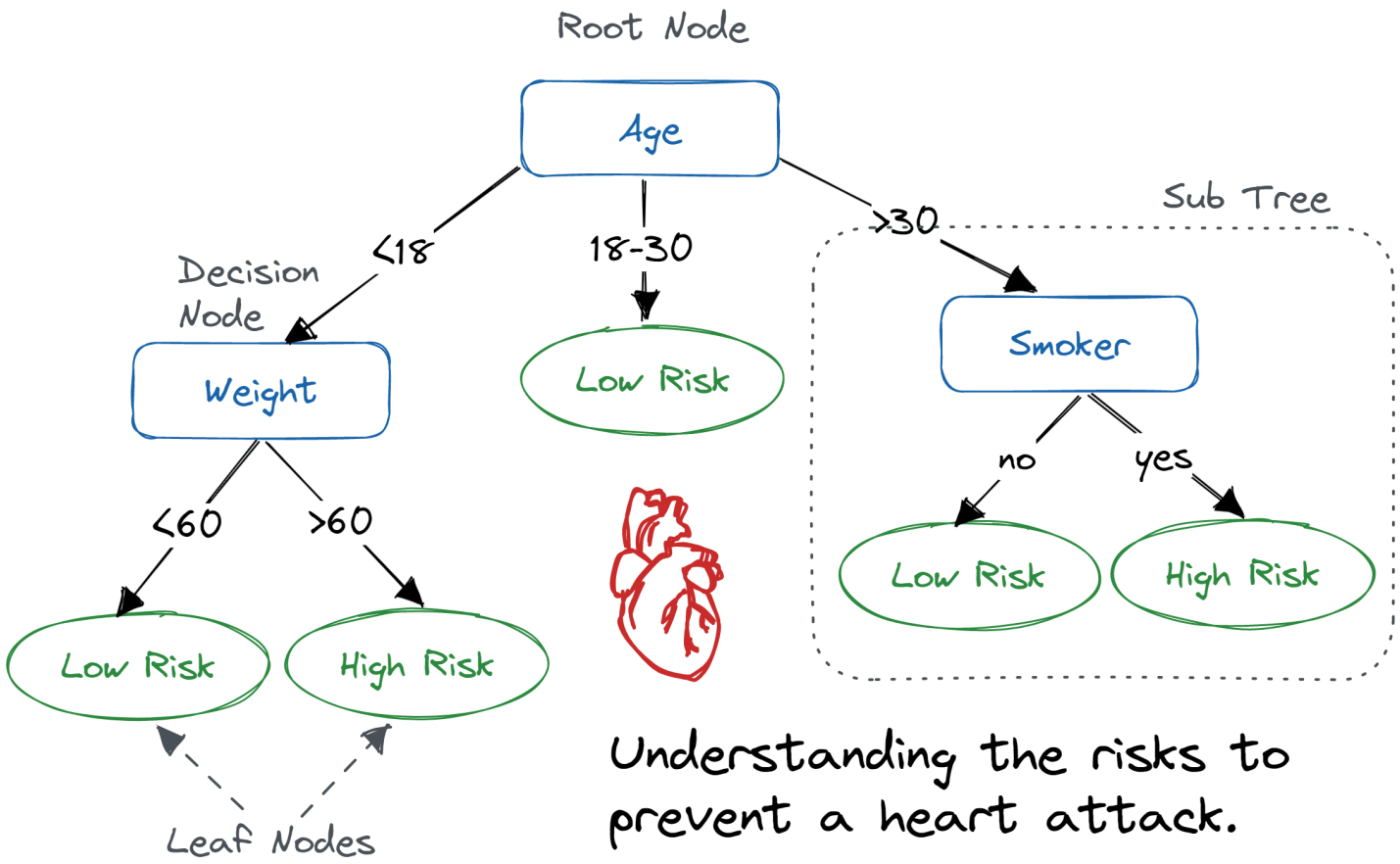
Ансамбли. Часть 2

Лекция 6

Повторение. Решающие деревья



Повторение. Разложение ошибки

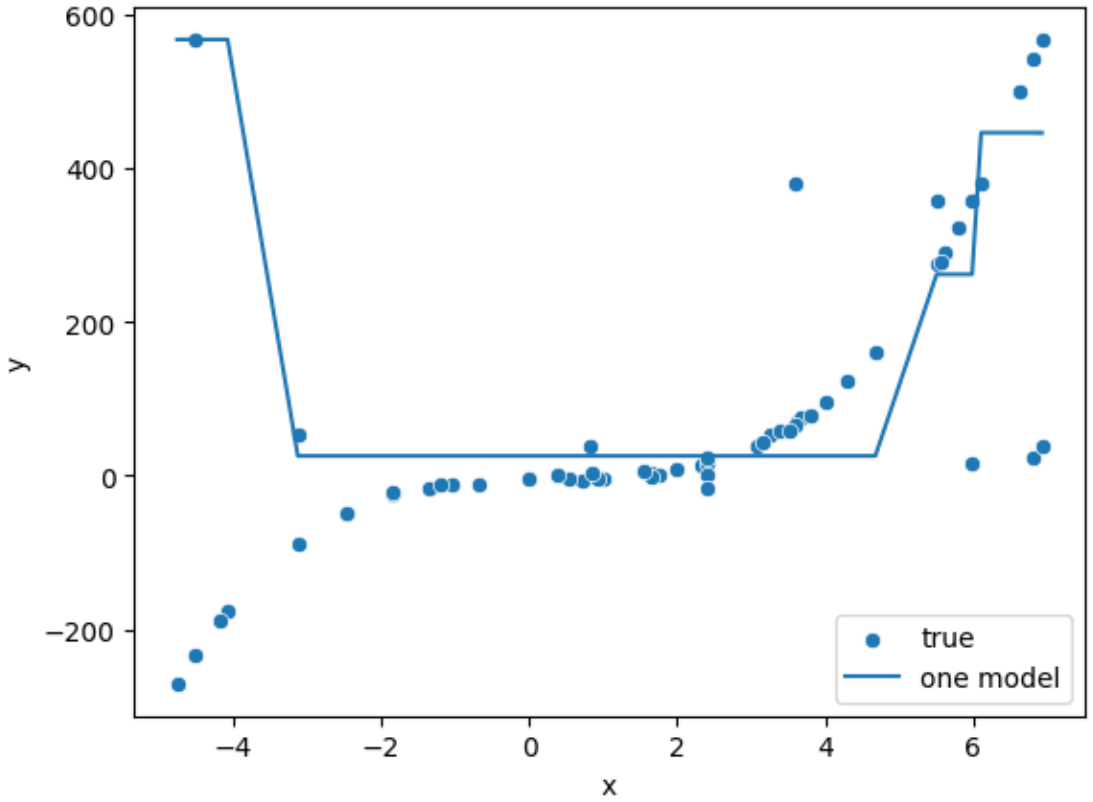
$$Q(a) = \mathbb{E}_x bias_X^2 a(x, X) + \mathbb{E}_x \mathbb{V}_X[a(x, X)] + \sigma^2$$

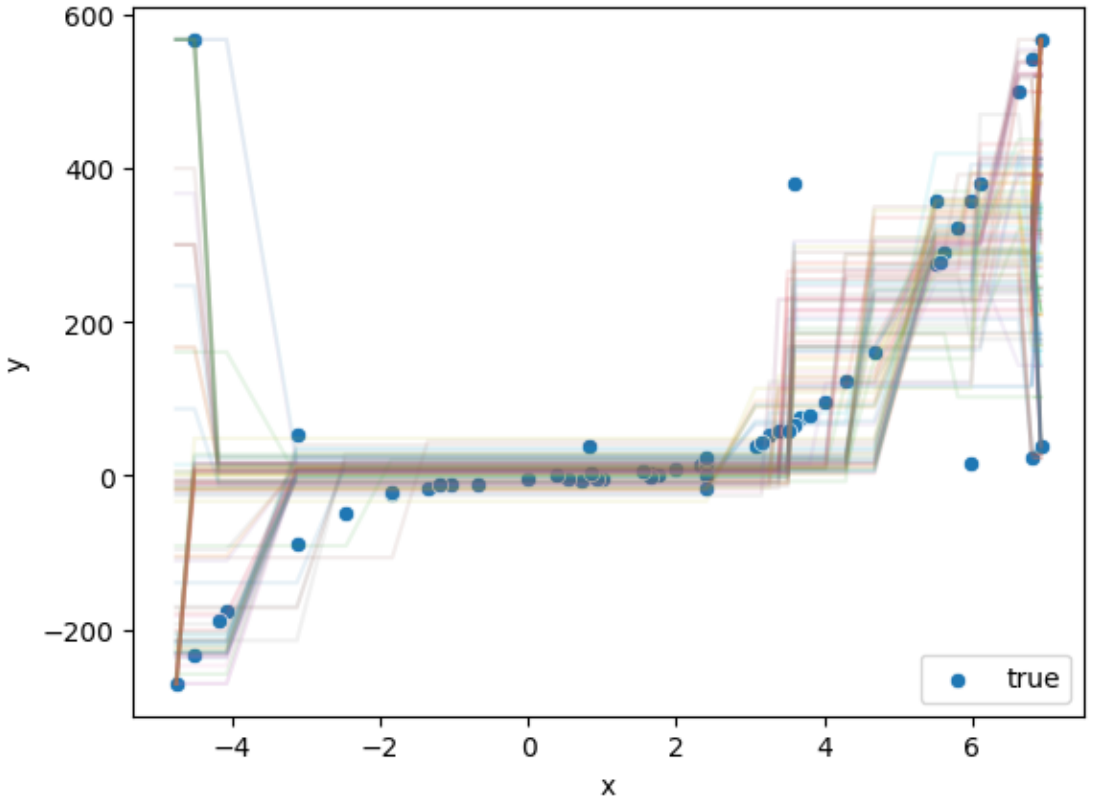
где $bias_X\ a(x, X) = f(x) - \mathbb{E}_x[a(x, X)]$ – **смещение**, показывает, насколько усредненная модель далека от истинной зависимости

$\mathbb{V}_X[a(x, X)] = \mathbb{E}_X\left[a(x, X) - \mathbb{E}_x[a(x, X)]\right]^2$ – **дисперсия**, показывает, насколько велико отклонение каждой отдельной модели от усредненной

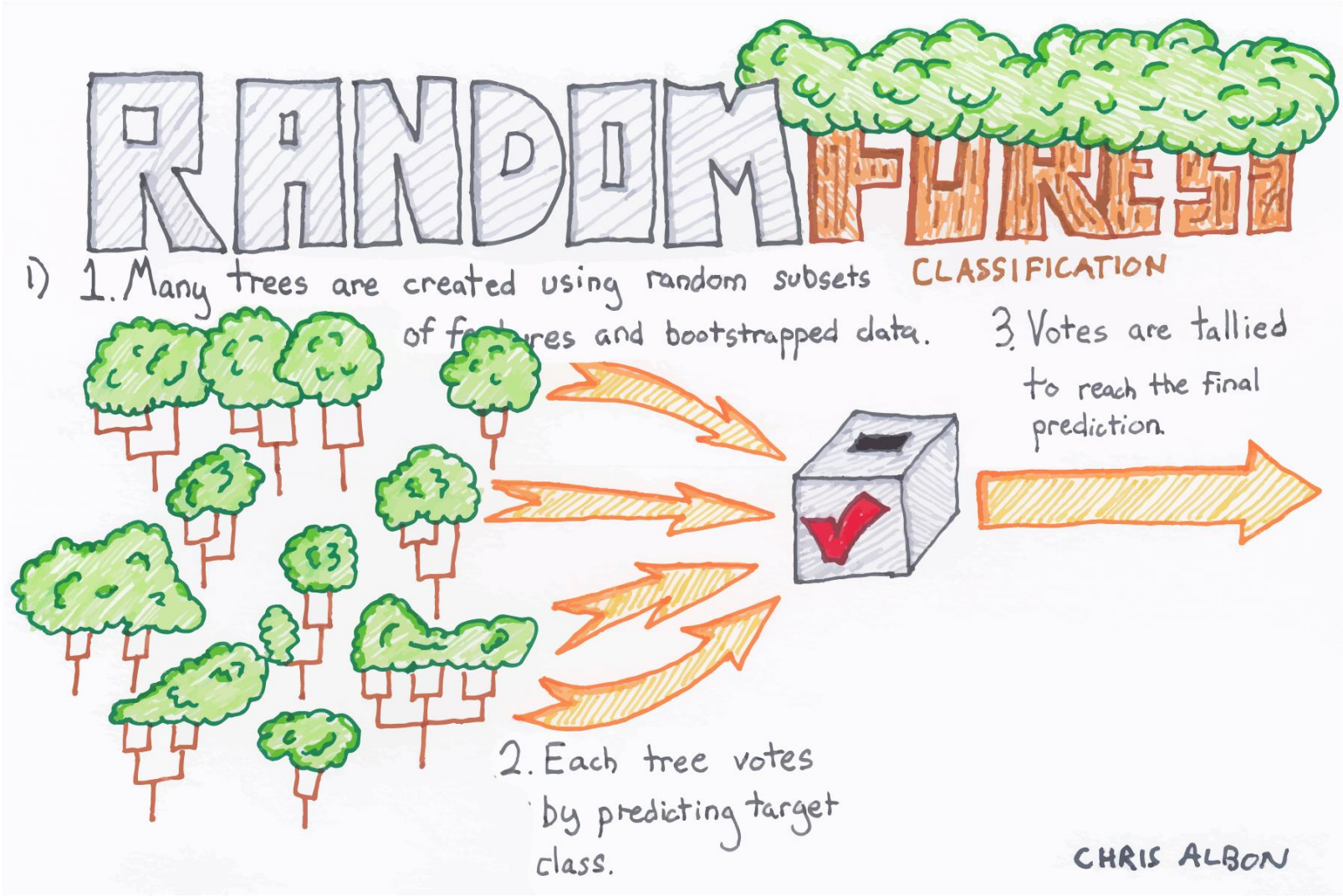
$\sigma^2 = \mathbb{E}_X \mathbb{E}_\epsilon[y(x, \epsilon) - f(x)]^2$ – **неустранимый шум** в данных

Повторение. Разложение ошибки наглядно





Повторение. Случайный лес



Пример. Стоимость квартиры. Лес

$\mathbb{X} = (x_i, y_i)_{i=1}^{l}$ – исходная выборка
$y \in R$

Случайный лес:

Каждое дерево обучается на ~67% квартирах и 1/3 случайных признаках в каждой вершине

В результате дает свое предсказание, оно усредняется с оставшимися деревьями и получается $\hat{y}$

Деревья обучаются параллельно

Пример. Стоимость квартиры. Бустинг

$\mathbb{X} = (x_i, y_i)_{i=1}^{l}$ – исходная выборка
$y \in R$

Идея:

Обучать каждое дерево последовательно

Первое дерево аппроксимирует $\boldsymbol{y}$ как сможет, а следующее постарается исправить ошибки предыдущего

Пример. Стоимость квартиры. Бустинг

Первое дерево $b_1(x)$

Ошибка на первом дереве $Q = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - b_1(x_i)\right)^2$

Выходит, что наше дерево на каждом объекте ошибается на $res_{\text{i}} = y_i - b_1(x_i)$

Тогда если мы будем предсказывать не $b_1(x_i)$, а $b_1(x_i) + res_i$, то получим наилучшую аппроксимацию

Давайте обучим модель $b_2(x_i)$, которая бы предсказывала res_i.

Тогда ошибка ансамбля (двух деревьев) будет $Q = \frac{1}{N}\sum_{i=1}^{N}(y_i - b_1(x_i) - b_2(x_i))^2$

Пример. Стоимость квартиры. Бустинг

Перепишем предсказание ансамбля как:

$$a(x) = \sum_{j=1}^{M} b_j(x)$$

В решающем лесе было вот так:

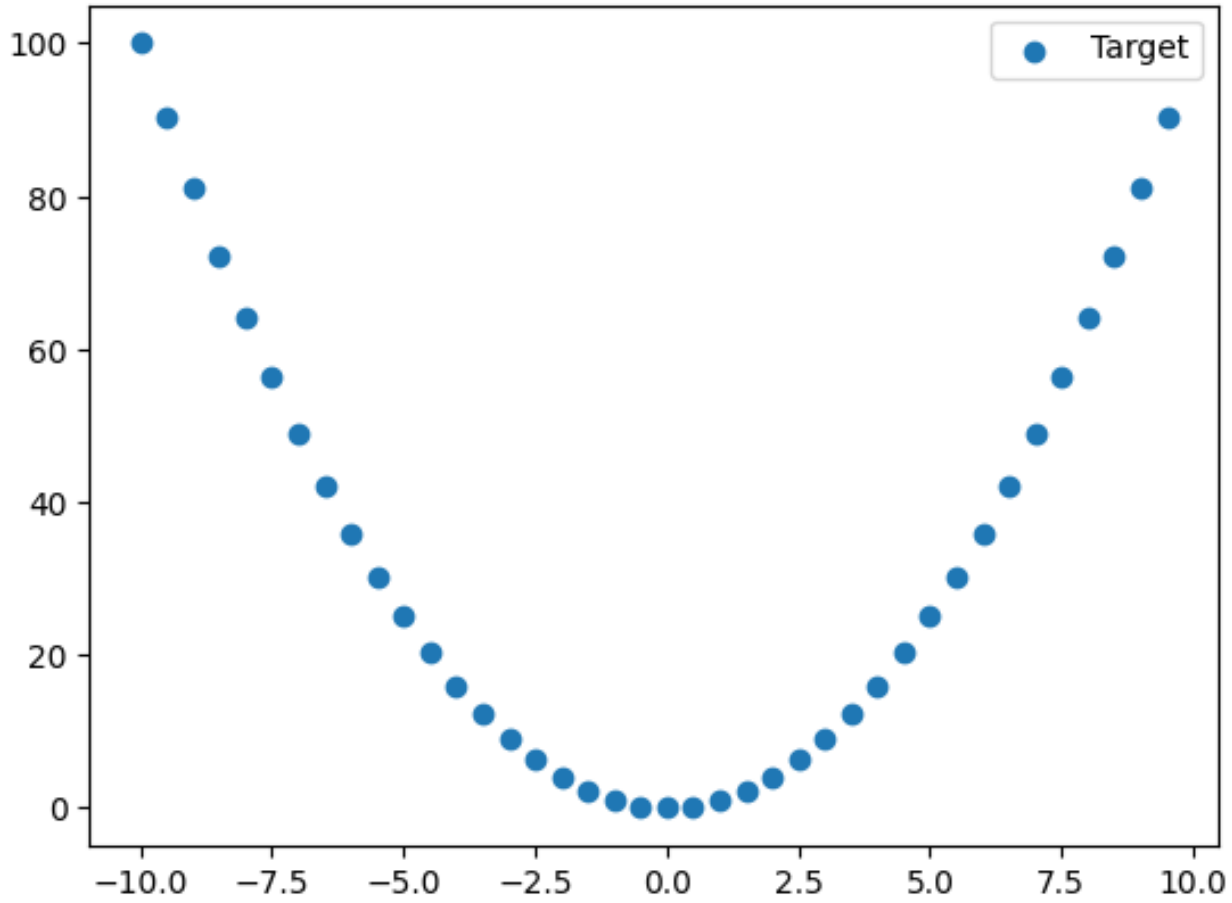
$$a(x) = \frac{1}{M} \sum_{j=1}^{M} b_j(x)$$

Важно: отличаются они не только лишь формулой

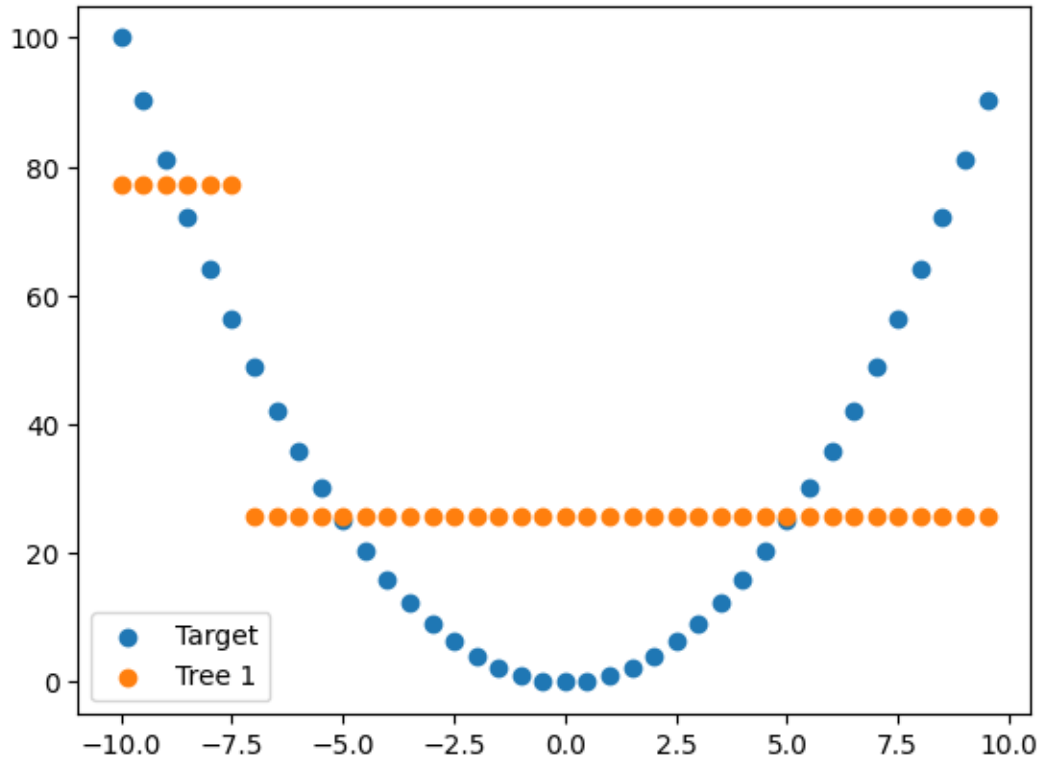
Пример. $y = x^2$

Будем поэтапно обучать решающие пеньки (деревья глубины 1) так, чтобы каждое следующее дерево исправляло ошибку предыдущего

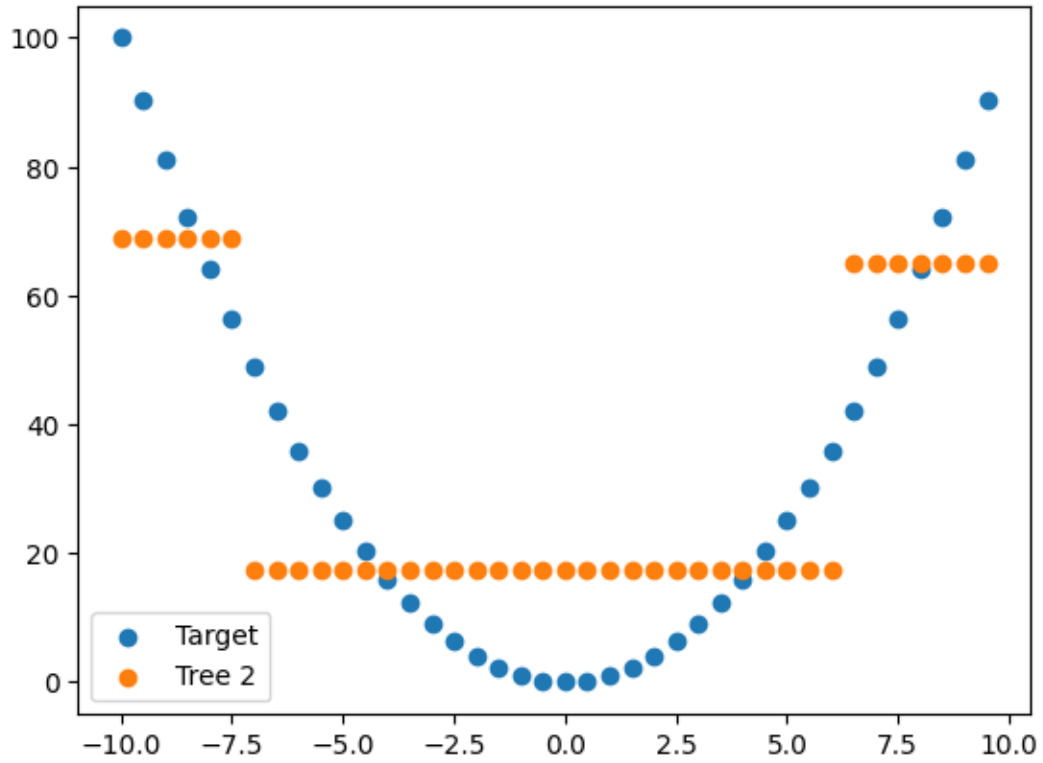
```
X = np.arange(-10, 10, 0.5)
y = X ** 2
X = X.reshape(-1, 1)
predicts = [np.zeros_like(y)]
residuals = []
n_trees = 3500
for _ in range(n_trees):
    residual = y - predicts[-1]
    residuals.append(residual)
    tree = DecisionTreeRegressor(max_depth=1)
    tree.fit(X, residual)
    new_predict = predicts[-1] + tree.predict(X)
    predicts.append(new_predict)
```



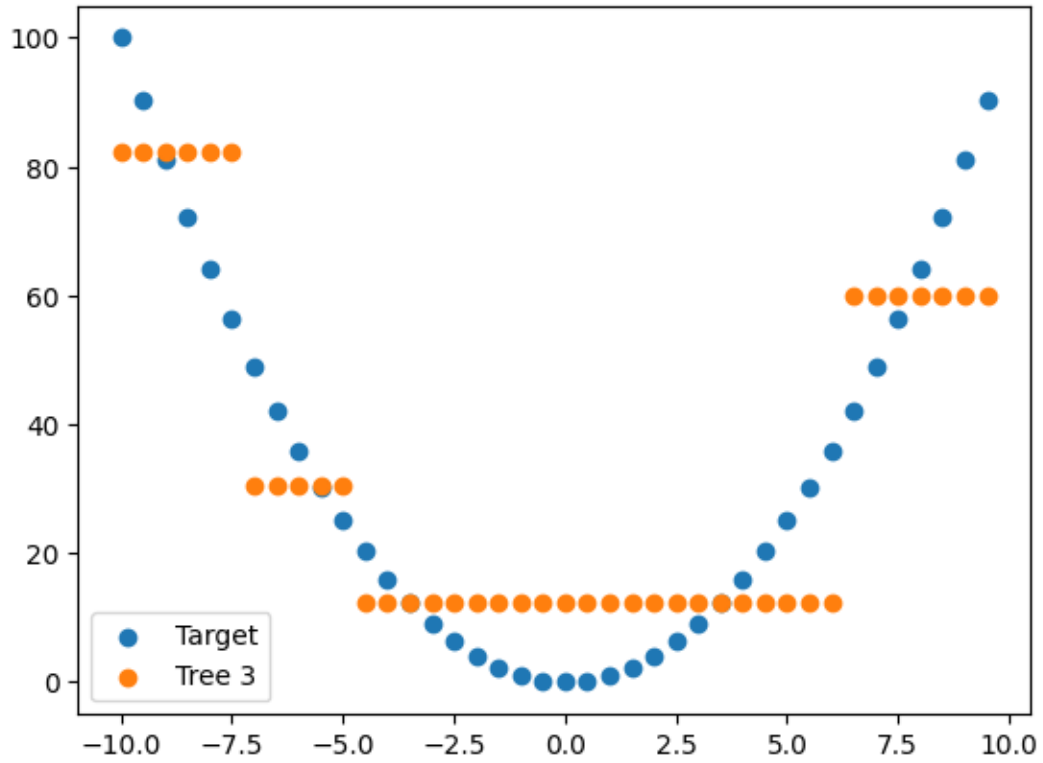
Пример. $y = x^2$. Дерево 1



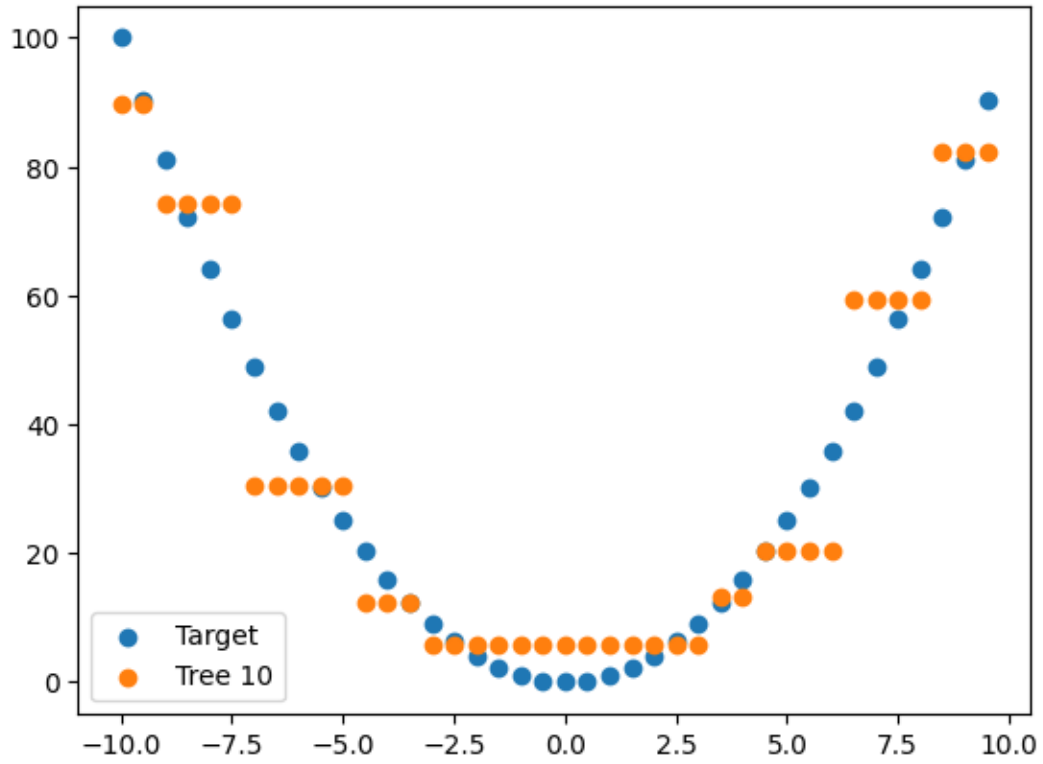
Пример. $y = x^2$. Дерево 2



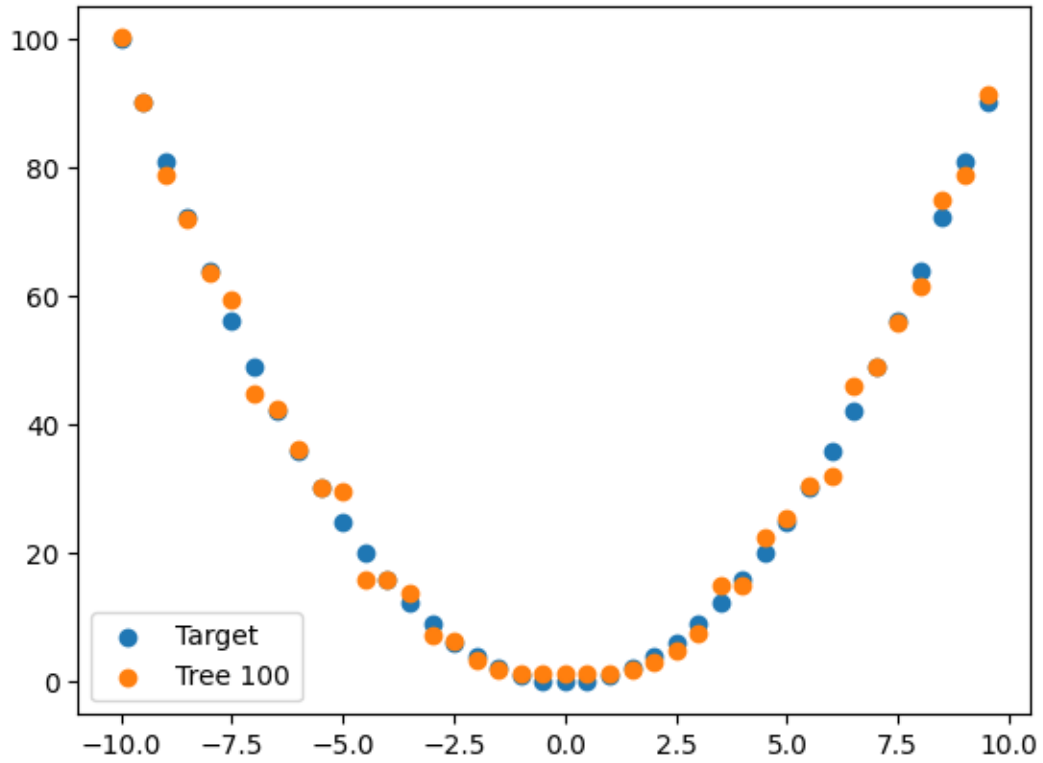
Пример. $y = x^2$. Дерево 3



Пример. $y = x^2$. Дерево 10



Пример. $y = x^2$. Дерево 100



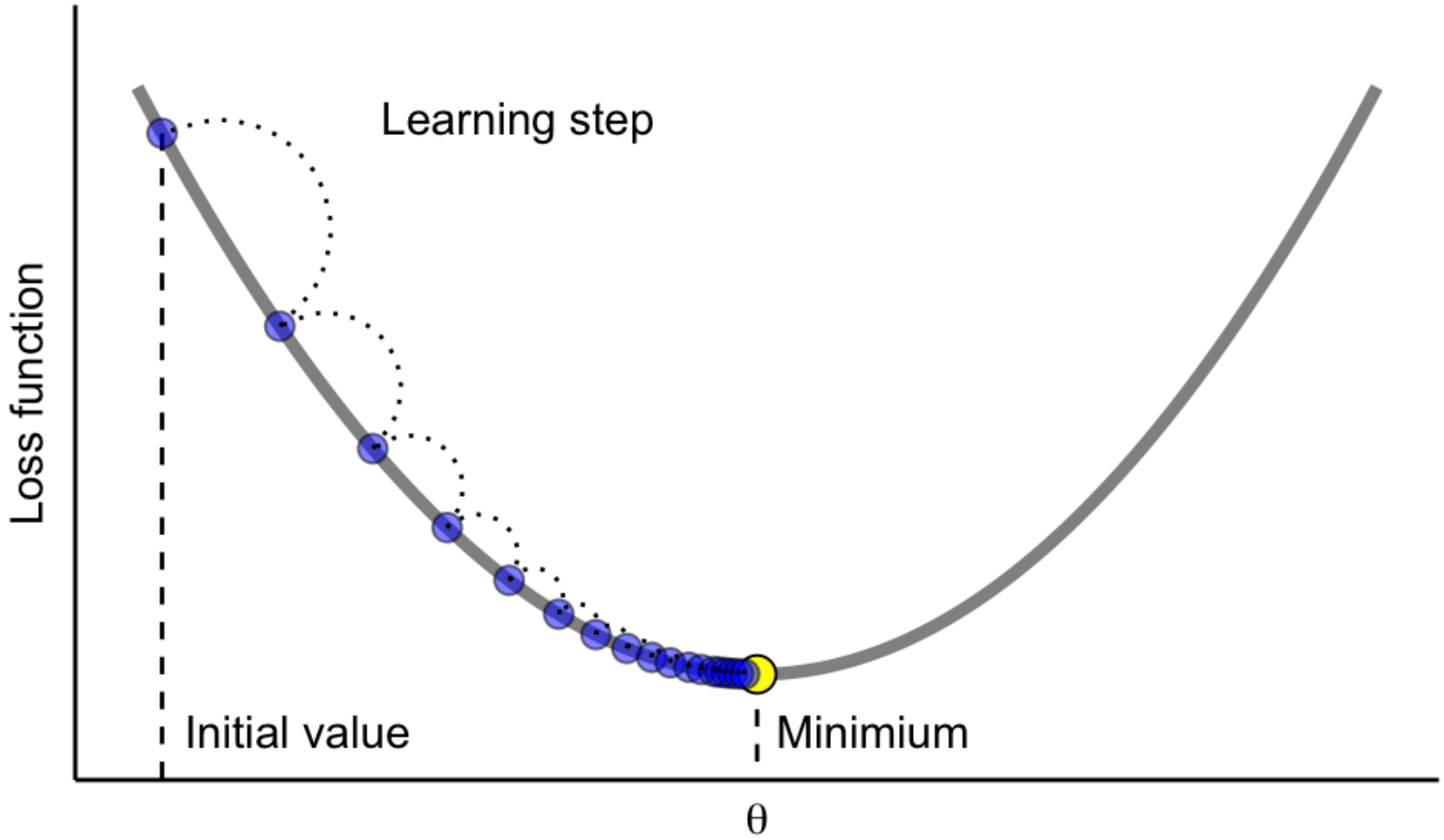
Почему именно $res_{\text{i}} = y_i - b_1(x_i)$? Где градиент если градиентный бустинг?

На самом деле:

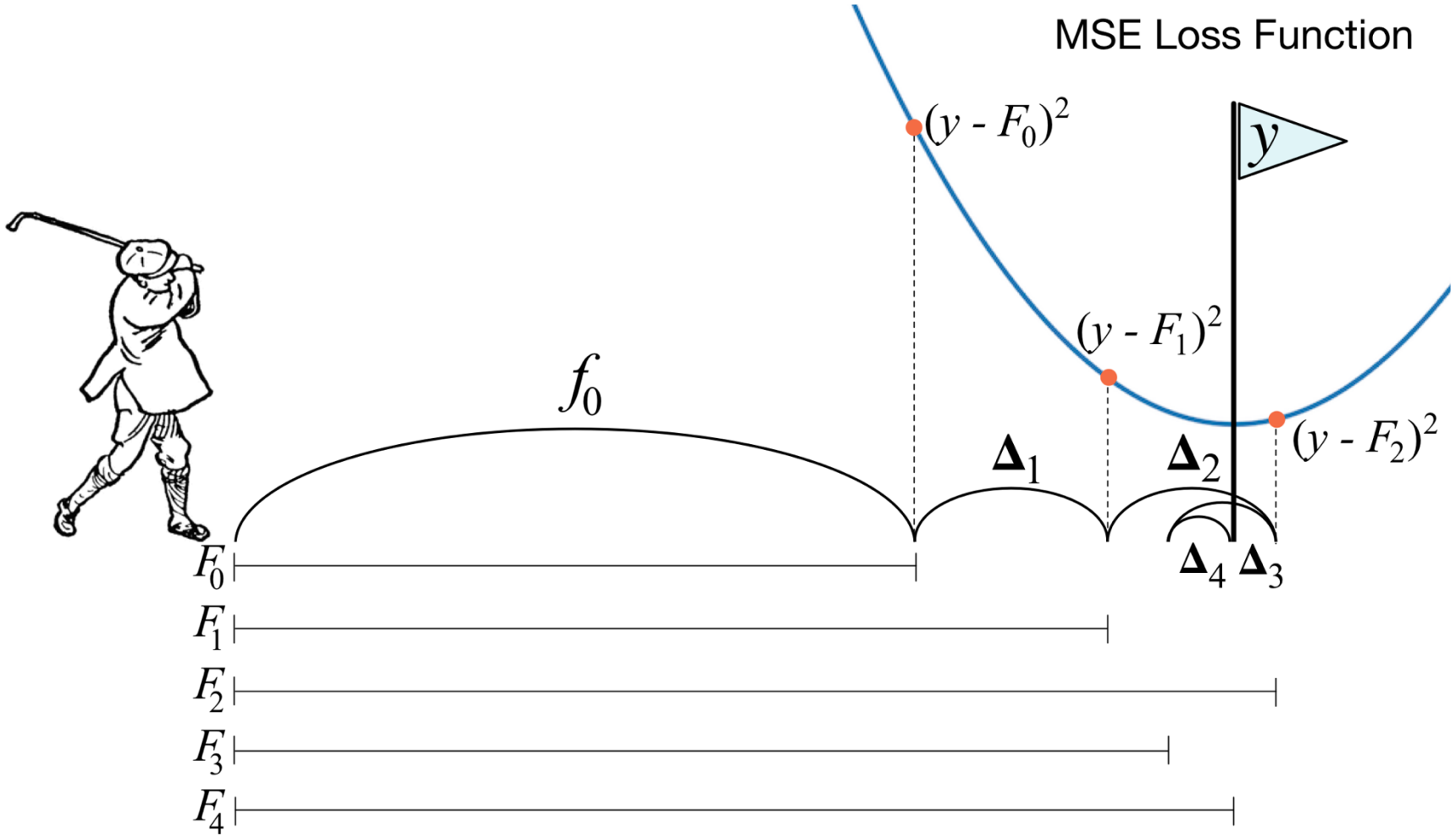
$$res = -\frac{dL}{da_{M-1}}$$

$$\boldsymbol{res} = -\frac{dL}{da_{M-1}} = -\frac{d\left(\frac{1}{N}\sum_{i=0}^{N}\left(y_i - a_{i,M-1}\right)^2\right)}{da_{M-1}} = \frac{2}{N}\left(\boldsymbol{y} - \boldsymbol{a_{M-1}}\right)$$

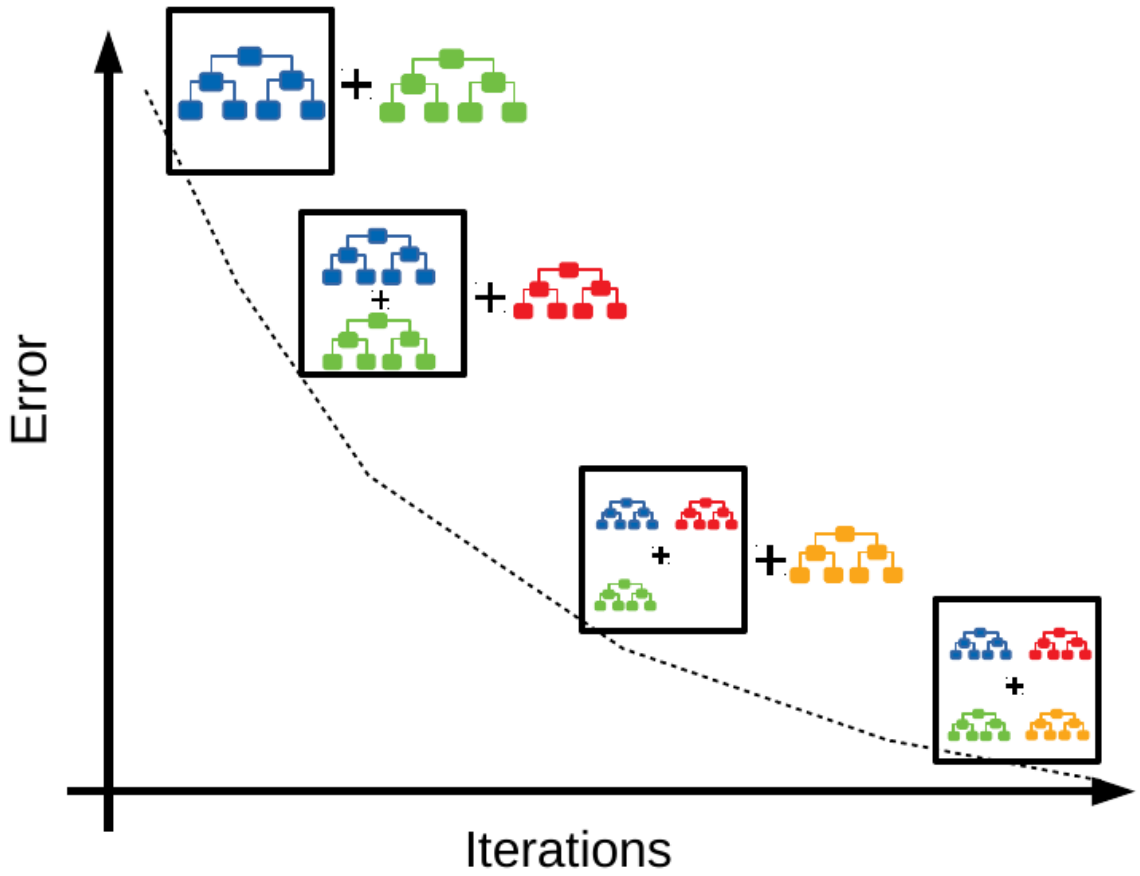
Градиентный бустинг в картинках



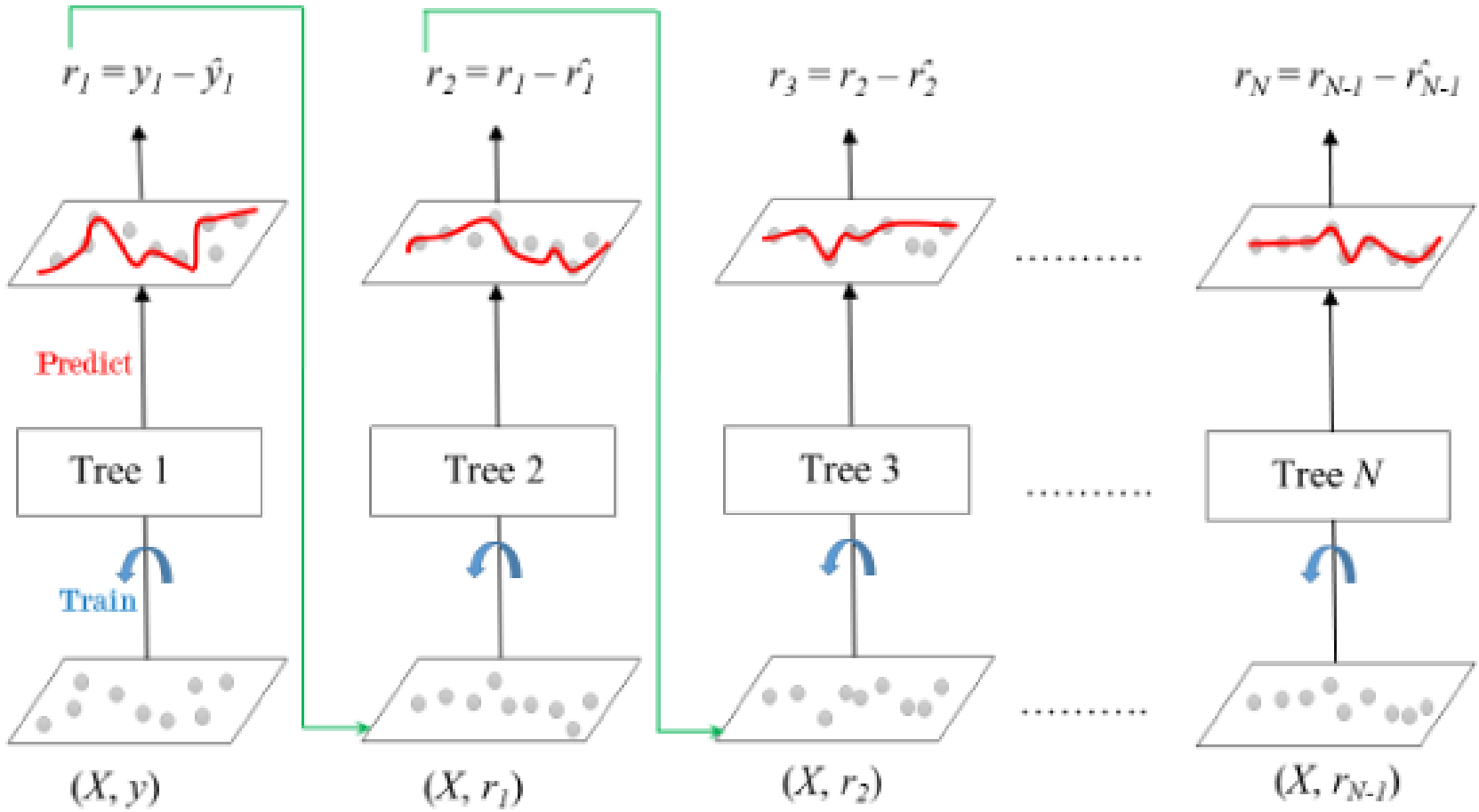
Градиентный бустинг в картинках



Градиетный бустинг в картинках



Градиетный бустинг в картинках



Градиентный бустинг

Каждое следующее дерево будет пытаться аппроксимировать антиградиент функции потерь

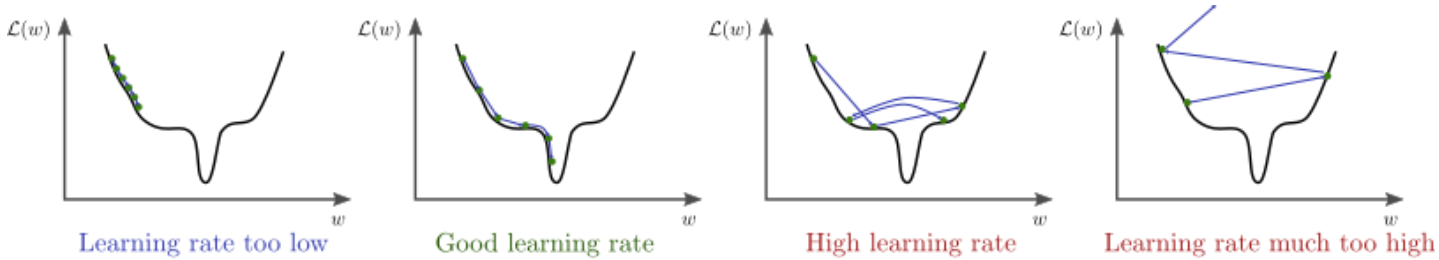
Какие алгоритмы выбрать в качестве базового?

1. Сложные алгоритмы (например, глубокие деревья) – за пару шагов мы сможем идеально приблизить все градиенты и получим 0 ошибку. В остатках будем аппроксимировать в основном случайный шум
2. Простые алгоритмы (например, неглубокие деревья) – на каждом шаге будем предсказывать что-то приближенное на на градиент (но не в точности его)

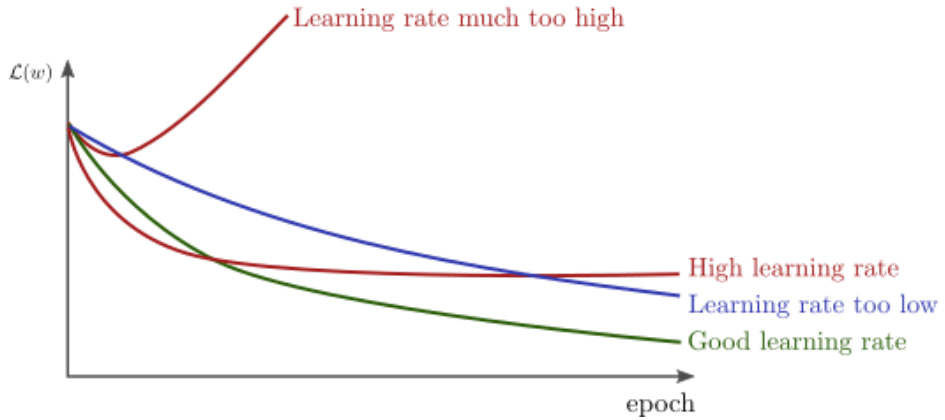
Поэтому используем **неглубокие деревья** в бустинге

Градиентный бустинг. Регуляризация

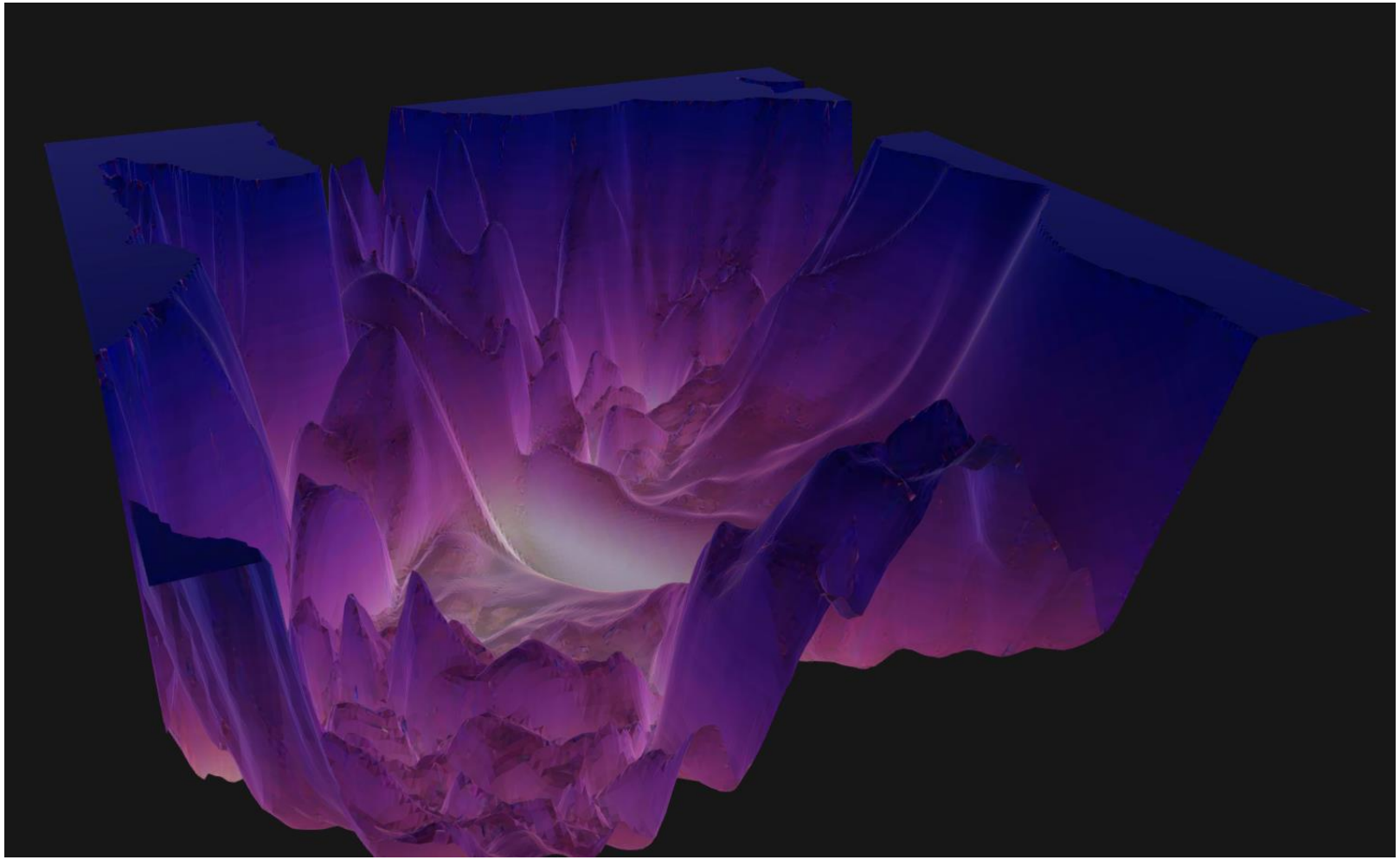
Вспоминая градиетный спуск, мы можем ходить не на полное значение антиградиента, а на какую-то его долю (шаг), которую называем learning rate (η):



$$a_M(x) = a_{M-1}(x) - \eta \cdot b_M(x)$$



Немного красоты



https://losslandscape.com/explorer

Смещение и разброс

В бустинге используем **неглубокие деревья**, которые обладают ?? смещением и ?? разбросом

В композиции бустинга смещение (растет ?, падает ?), разброс (растет ?, падает ?)

Реализации бустинга

Реализации бустинга

	CatBoost		LightGBM		XGBoost		H2O	
	Tuned	Default	Tuned	Default	Tuned	Default	Tuned	Default
Adult	**0.26974**	0.27298 +1.21%	0.27602 +2.33%	0.28716 +6.46%	0.27542 +2.11%	0.28009 +3.84%	0.27510 +1.99%	0.27607 +2.35%
Amazon	**0.13772**	0.13811 +0.29%	0.16360 +18.80%	0.16716 +21.38%	0.16327 +18.56%	0.16536 +20.07%	0.16264 +18.10%	0.16950 +23.08%
Click prediction	**0.39090**	0.39112 +0.06%	0.39633 +1.39%	0.39749 +1.69%	0.39624 +1.37%	0.39764 +1.73%	0.39759 +1.72%	0.39785 +1.78%
KDD appetency	0.07151	**0.07138** –0.19%	0.07179 +0.40%	0.07482 +4.63%	0.07176 +0.35%	0.07466 +4.41%	0.07246 +1.33%	0.07355 +2.86%

https://catboost.ai/

Реализации бустинга

KDD churn	**0.23129**	0.23193 +0.28%	0.23205 +0.33%	0.23565 +1.89%	0.23312 +0.80%	0.23369 +1.04%	0.23275 +0.64%	0.23287 +0.69%
KDD internet	**0.20875**	0.22021 +5.49%	0.22315 +6.90%	0.23627 +13.19%	0.22532 +7.94%	0.23468 +12.43%	0.22209 +6.40%	0.24023 +15.09%
KDD upselling	**0.16613**	0.16674 +0.37%	0.16682 +0.42%	0.17107 +2.98%	0.16632 +0.12%	0.16873 +1.57%	0.16824 +1.28%	0.16981 +2.22%
KDD 98	**0.19467**	0.19479 +0.07%	0.19576 +0.56%	0.19837 +1.91%	0.19568 +0.52%	0.19795 +1.69%	0.19539 +0.37%	0.19607 +0.72%
Kick prediction	**0.28479**	0.28491 +0.05%	0.29566 +3.82%	0.29877 +4.91%	0.29465 +3.47%	0.29816 +4.70%	0.29481 +3.52%	0.29635 +4.06%

https://catboost.ai/

Гиперпараметры бустинга

```
class CatBoostClassifier(iterations=None,
                         learning_rate=None,
                         depth=None,
                         l2_leaf_reg=None,
                         model_size_reg=None,
                         rsm=None,
                         loss_function=None,
                         border_count=None,
                         feature_border_type=None,
                         per_float_feature_quantization=None,
                         input_borders=None,
                         output_borders=None,
                         fold_permutation_block=None,
                         od_pval=None,
                         od_wait=None,
                         od_type=None,
                         nan_mode=None,
                         counter_calc_method=None,
                         leaf_estimation_iterations=None,
                         leaf_estimation_method=None,
                         thread_count=None,
                         random_seed=None,
                         use_best_model=None,
                         verbose=None,
                         logging_level=None,
                         metric_period=None,
                         ctr_leaf_count_limit=None,
                         store_all_simple_ctr=None,
                         max_ctr_complexity=None,
                         has_time=None,
                         allow_const_label=None,
                         classes_count=None,
                         class_weights=None,
                         auto_class_weights=None,
                         one_hot_max_size=None,
                         random_strength=None,
                         name=None,
                         ignored_features=None,
                         train_dir=None,
                         custom_loss=None,
                         custom_metric=None,
                         eval_metric=None,
                         bagging_temperature=None,
                         save_snapshot=None,
                         snapshot_file=None,
                         snapshot_interval=None,
                         fold_len_multiplier=None,
                         used_ram_limit=None,
```

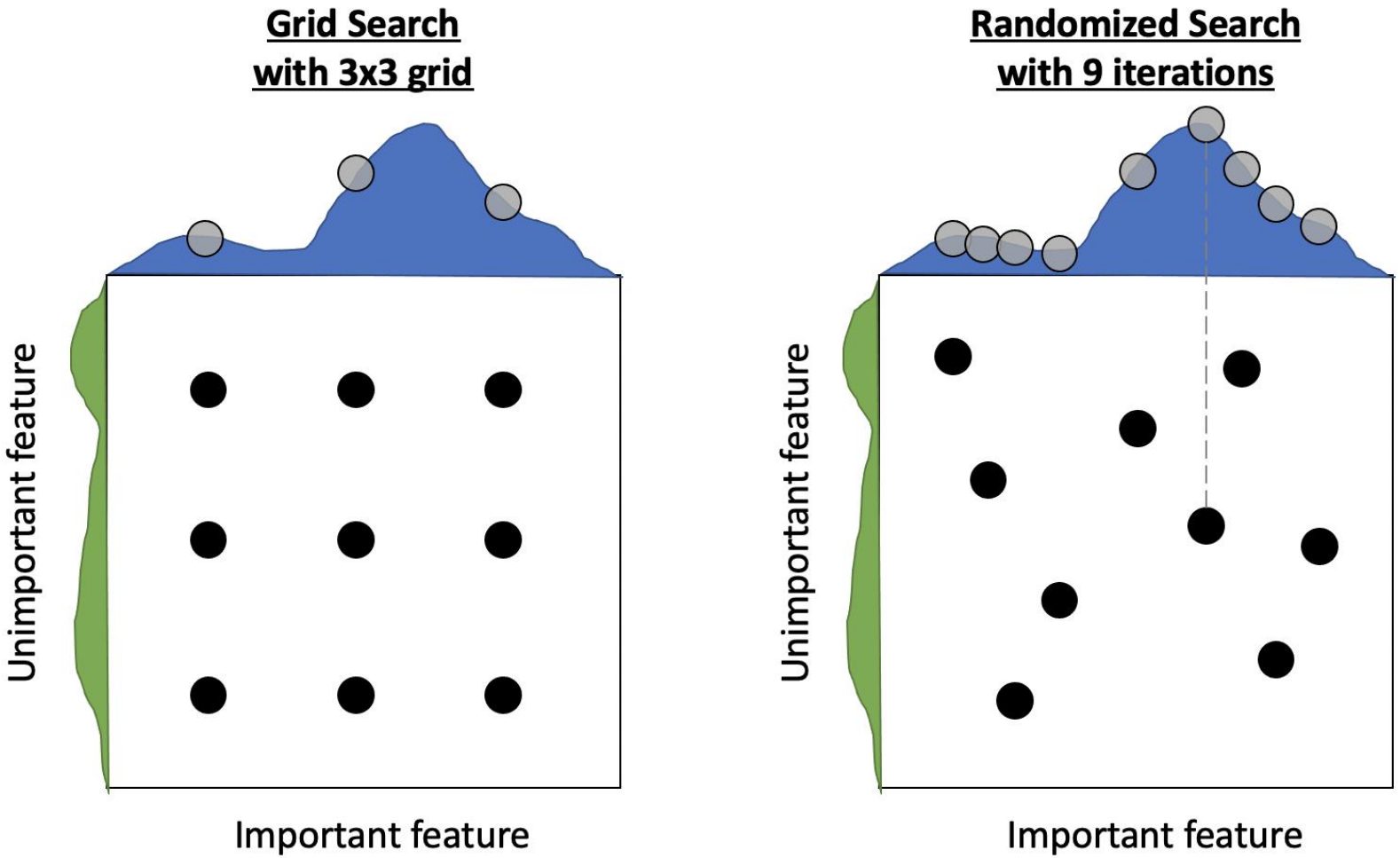
Гиперпараметры бустинга

```
used_ram_limit=None,
gpu_ram_part=None,
allow_writing_files=None,
final_ctr_computation_mode=None,
approx_on_full_history=None,
boosting_type=None,
simple_ctr=None,
combinations_ctr=None,
per_feature_ctr=None,
task_type=None,
device_config=None,
devices=None,
bootstrap_type=None,
subsample=None,
sampling_unit=None,
dev_score_calc_obj_block_size=None,
max_depth=None,
n_estimators=None,
num_boost_round=None,
num_trees=None,
colsample_bylevel=None,
random_state=None,
reg_lambda=None,
objective=None,
eta=None,
```

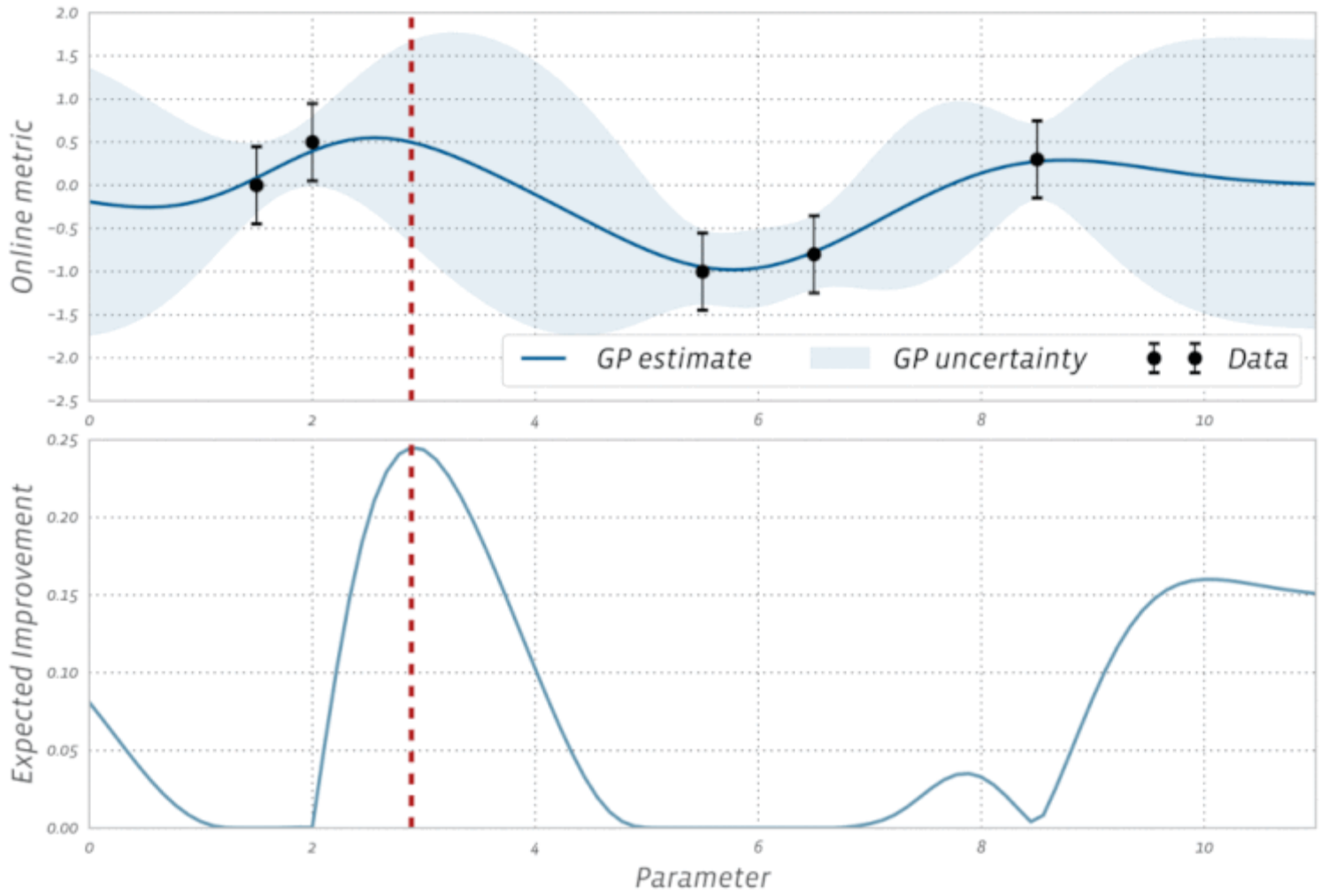
```
max_bin=None,
scale_pos_weight=None,
gpu_cat_features_storage=None,
data_partition=None
metadata=None,
early_stopping_rounds=None,
cat_features=None,
grow_policy=None,
min_data_in_leaf=None,
min_child_samples=None,
max_leaves=None,
num_leaves=None,
score_function=None,
leaf_estimation_backtracking=None,
ctr_history_unit=None,
monotone_constraints=None,
feature_weights=None,
penalties_coefficient=None,
first_feature_use_penalties=None,
```

```
model_shrink_rate=None,
model_shrink_mode=None,
langevin=None,
diffusion_temperature=None,
posterior_sampling=None,
boost_from_average=None,
text_features=None,
tokenizers=None,
dictionaries=None,
feature_calcers=None,
text_processing=None,
fixed_binary_splits=None)
```

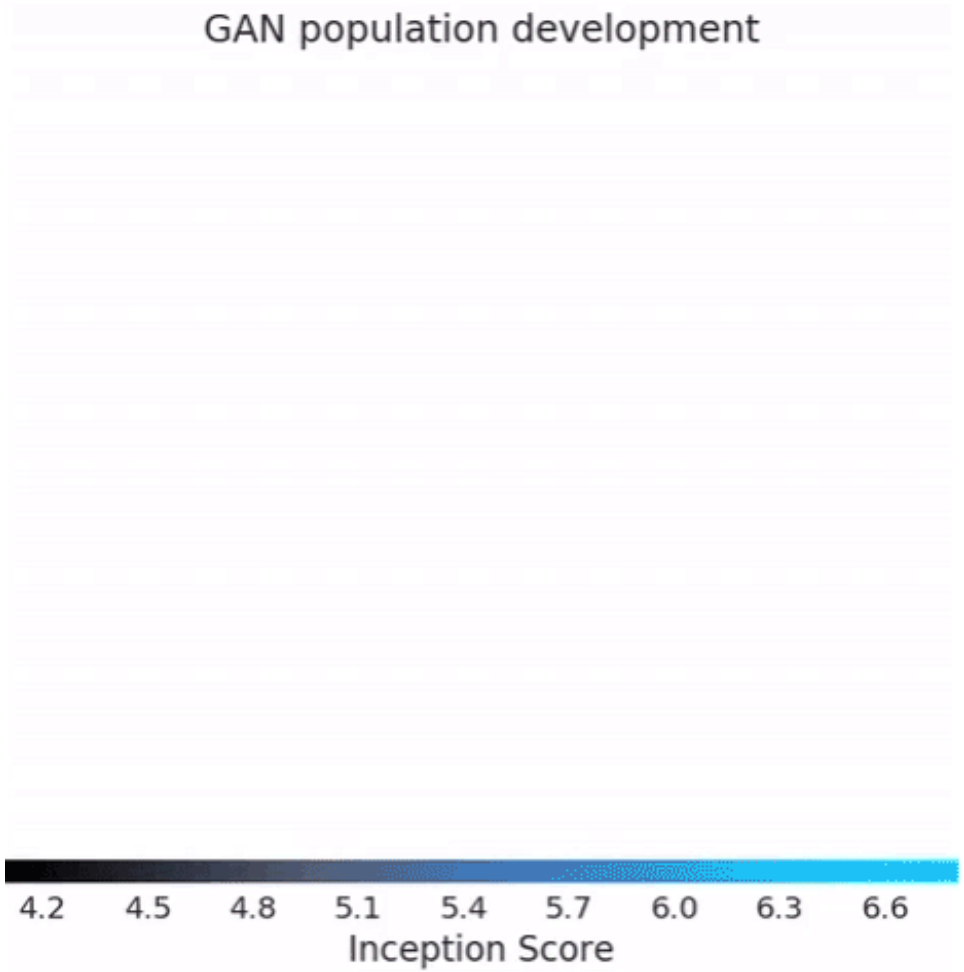
Как подбирать гиперпараметры?

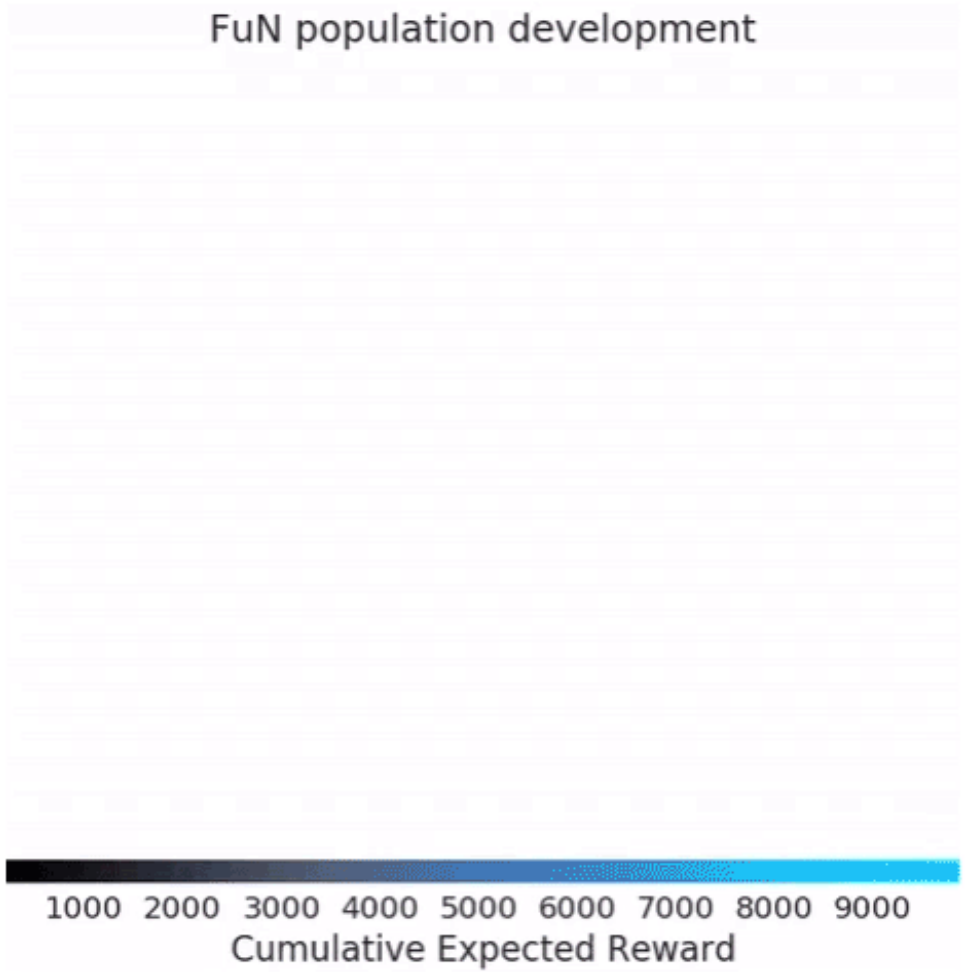


Как подбирать гиперпараметры?



Как подбирать гиперпараметры?





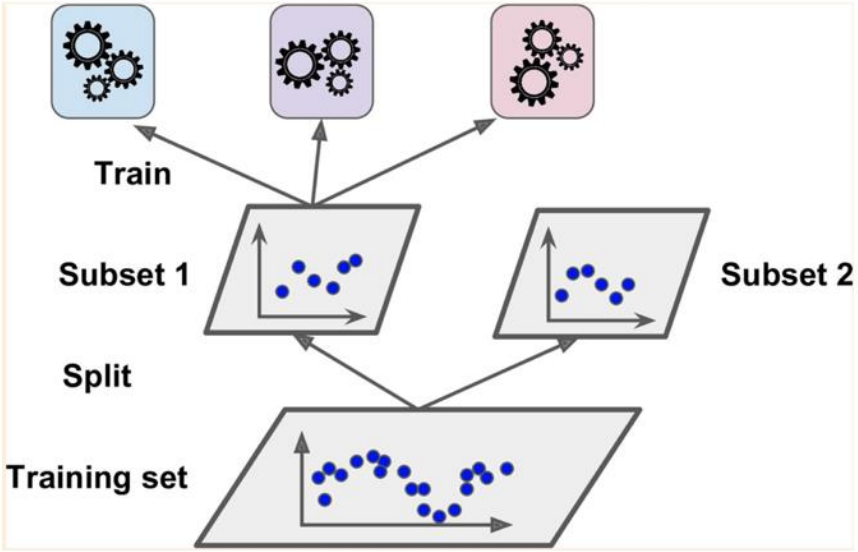
https://education.yandex.ru/handbook/ml/article/podbor-giperparametrov

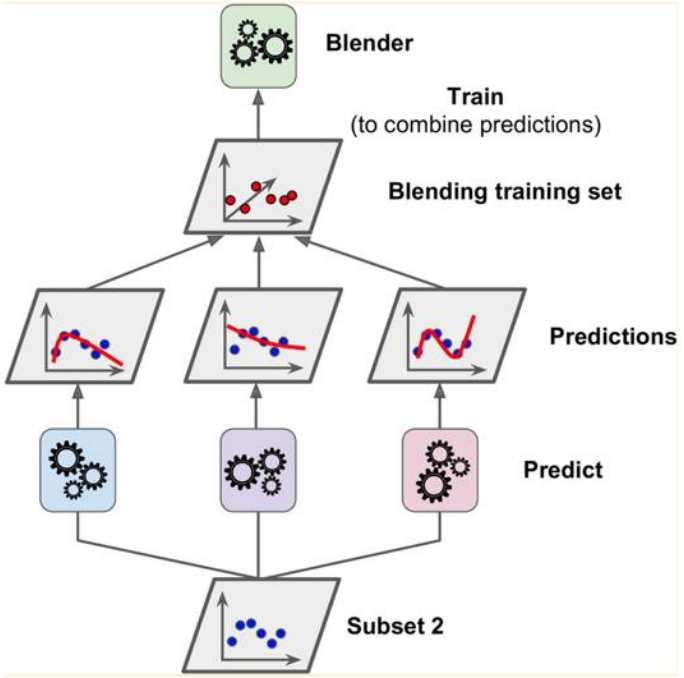
Блендинг

$X = X_1 + X_2$

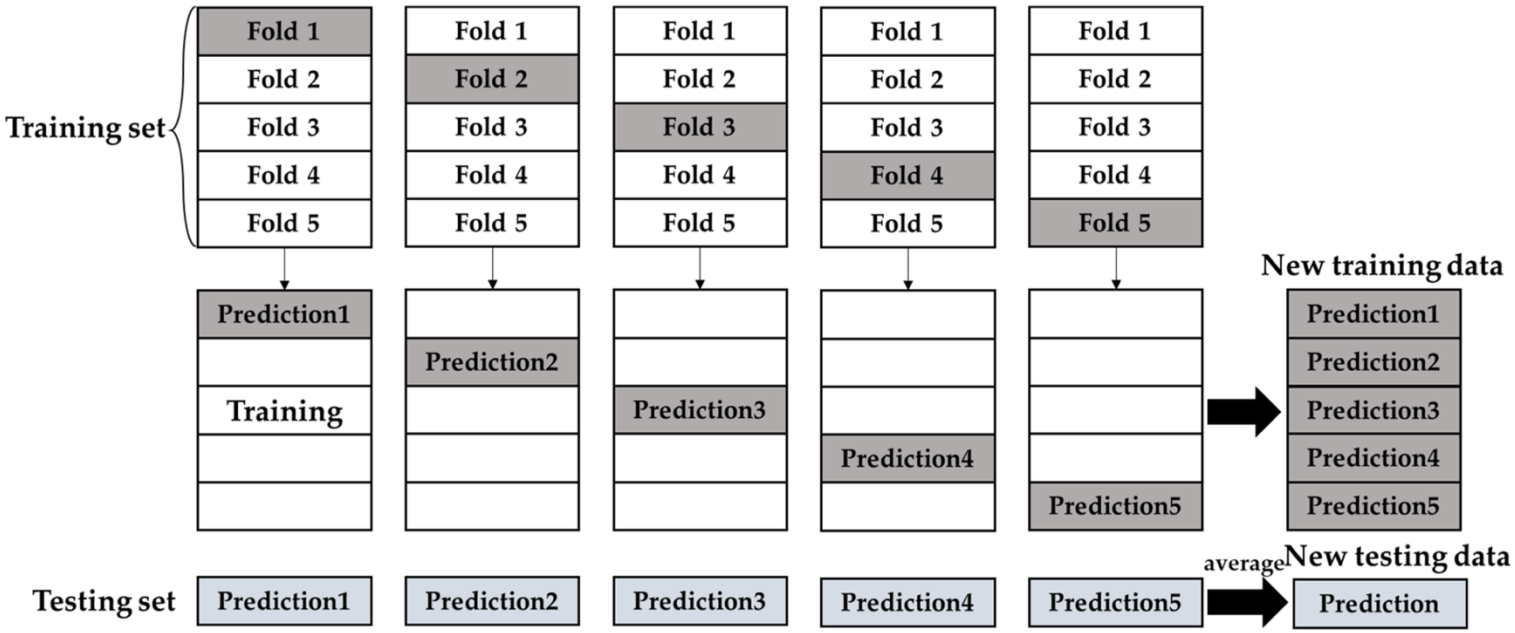
Обучаем базовые модели на X_1

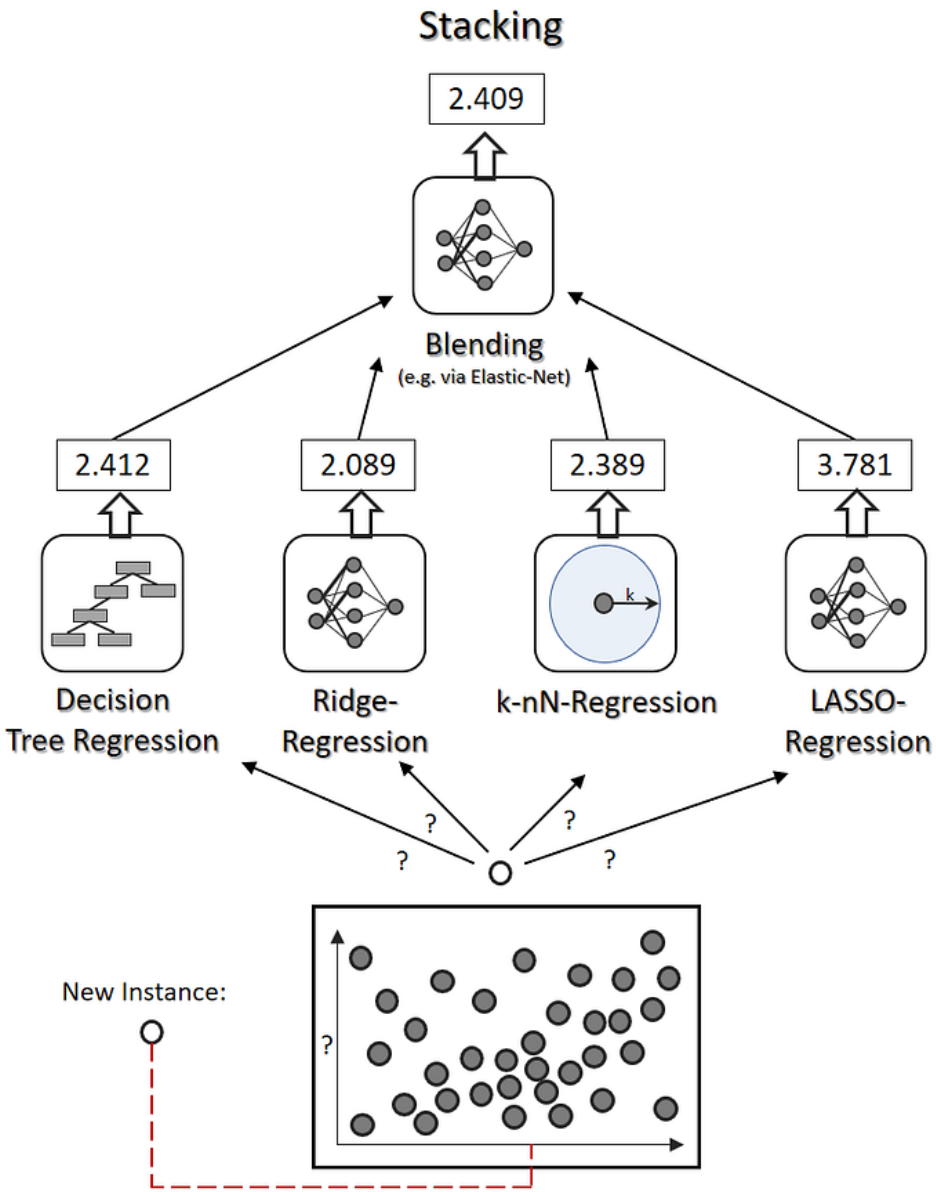
Обучаем метамодель на X_2





Стекинг


Training set
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Prediction1
Training
Prediction2
Prediction3
Prediction4
Prediction5
New training data
Testing set
average
New testing data
Prediction


Stacking
2.409
Blending
(e.g. via Elastic-Net)
2.412
2.089
2.389
3.781
Decision
Tree Regression
Ridge-
Regression
k-nN-Regression
LASSO-
Regression
k
?
New Instance: