Отчет по практической работе №3. Ансамблирование.

Кузьмин Александр

14 декабря 2021 г.

1 Постановка задачи

В данной практической работе перед нами стояли следующие задачи:

- Реализовать модели ансамблирования: `RandomForest` и `GradientBoosting`.
- Провести качественный анализ работы моделей на предложенном датасете.

Перед нами стоит задача построения модели регрессии, восстановления численного значения по имеющейся тренировочной выборке. Более формально, мы имеем набор данных $\mathcal{D} = \{(x_i,\ y_i) \mid x_i \in \mathbb{R}^n,\ y_i \in \mathbb{R})\}$ и для каждого примера x_i должны научиться восстанавливать соответствующее ему значение целевой переменной y_i.

Коротко рассмотрим каждый из алгоритмов, предложенных для изучения. `RandomForest` алгоритм ансамблирования решающих деревьев. Основная идея данного алгоритма заключается в том, чтобы усреднить много слабых моделей, и таким образом получить одну сильную. Каждый из базовых алгоритмов обучается на некотором случайном подмножестве признаков и выборке, полученной методом `bootstrap`. Такой подход говорит нам, как выбирать объект из тренировочного множества: на каждом шаге мы извлекаем равновероятно объект из тренировочного датасета, реализуя схему с возвращением, и таким образом набираем выборку необходимого объема. В простейшей своей реализации очередное предсказание алгоритма строится следующим образом:

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

где a - модель ансамблирования, b_i - базовый алгоритм.

Теперь разберёмся с `GradientBoosting`. Главным отличием этого алгоритма является его принципиальная последовательность. В бустинге алгоритмы строятся один за другим, каждый новый алгоритм обучается минимизировать ошибку всех предыдущих ему. Более формально это выглядит так: на каждом шаге алгоритма мы получаем модель b_t и α_t по следующему принципу:

$$b_t = arg\min_{b\in\mathcal{B}} \sum_{i=1}^{l} (\mathcal{L}'(f_i, y_i) + b(x_i))^2$$

$$\alpha_t = arg\min_{\alpha>0} \sum_{i=1}^{l} \mathcal{L}(\alpha * b_t(x_i) + f_i(x_i))$$

где f_i - ансамбль предшествующих алгоритмов, т.е. $\sum_{i=1}^{t-1} \alpha_{t-1} * b_{t-1}$.

Таким образом, бустинг строит ансамбль алгоритмов, каждый из которых «поправляет» предыдущие.

2 Эксперименты

В этой секции мы обсудим вопросы, касающиеся зависимости каждого из алгоритмов от некоторых гиперпараметров. Все эксперименты проведем на валидационной выборке одного датасета, в котором ставится задача предсказания цены дома при продаже. Мы провели минимальную предобработку датасета:

- Добавили бинарный признак, показывающий попал ли дом по программу реновации.
- Специальным образом закодировали почтовые индексы в надежде на то, что можно по ним выделить премиальные и более бедные районы.
- Провели стандартизацию всех численных признаков.
- С помощью OHE закодировали категориальные признаки.

Влияние каждого из гиперпараметров в будующем мы будем изучать с двух сторон: влияние на значение целевого функционала и время работы алгоритма.

В качестве метрики используется `RMSE`:

$$\text{RMSE}(X,\ Y) = \sqrt{\frac{1}{l}\sum_{i=1}^{l}(a(x_i) - y_i)^2}$$

2.1 Random Forest

Здесь рассмотрим влияние на модель следующих гиперпараметров:

- `n_estimators` - количество базовых алгоритмов в ансамбле.
- `max_depth` - максимальная глубина каждого дерева.
- `feature_subsample_size` - доля признаков, на которых обучается алгоритм.

2.1.1 Число базовых алгоритмов

Данный параметр показывает, сколько деревьев нам необходимо построить для ансамбля. Будем рассматривать его по следующей сетке: `n_estimators` $= \{10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000\}$. Ясно, что чем больше базовых алгоритмов в ансамбле, тем больше время его тренировки, тем он может быть точнее, по крайней мере, так говорит теория. Результаты наших экспериментов представлены ниже на Рис.1 и 2.

Мы видим, что лучшее значение в нашей задаче достигается для `n_estimators` $= 200$. Сначал метрика убывает, достигая минимума в области 200, после чего возрастает. выходит на плато, когда добавление новых деревьев не помогает улучшить точность.

Также мы видим, что увеличение числа деревьев ведет к линейному возрастанию времени работы алгоритма.

Из этих графиков можно выяснить, что нет смысла наращивать число деревьев в ансамбле, так как это не только не приводит к улучшению качества модели, но и делает её медленной.

2.1.2 Число признаков

Данный параметр отвечает за то, какая доля от общего числа признаков в датасете используется для обучения очередного базавого алгоритма. Мы рассматриваем следующую сетку данного гиперпараметра: `feature_subsample_size` $= \{0.3,\ 0.5,\ 0.6,\ 0.7,\ 0.8,\ 0.85,\ 0.9,\ 0.95,\ 1\}$. Теория подсказывает: чем менее коррелированы ответы наших алгоритмов, тем лучше должны быть их предсказания, поэтому при меньшем числе общих признаков в каждом из базовых алгоритмов наши предсказания должны стать точнее. Результаты экспериментов приведены на Рис.3 и 4.

Мы видим, что оптимум достигается на 0.6-0.7 объёме от общего числа признаков, при этом рост времени работы алгоритма с увеличением размерности пространства признаков оказывается линейным.

Интересно заметить, что при значениях близких к единице качество падает, разнообразность алгоритмов уменьшается, следовательно они становятся более скореллированными.

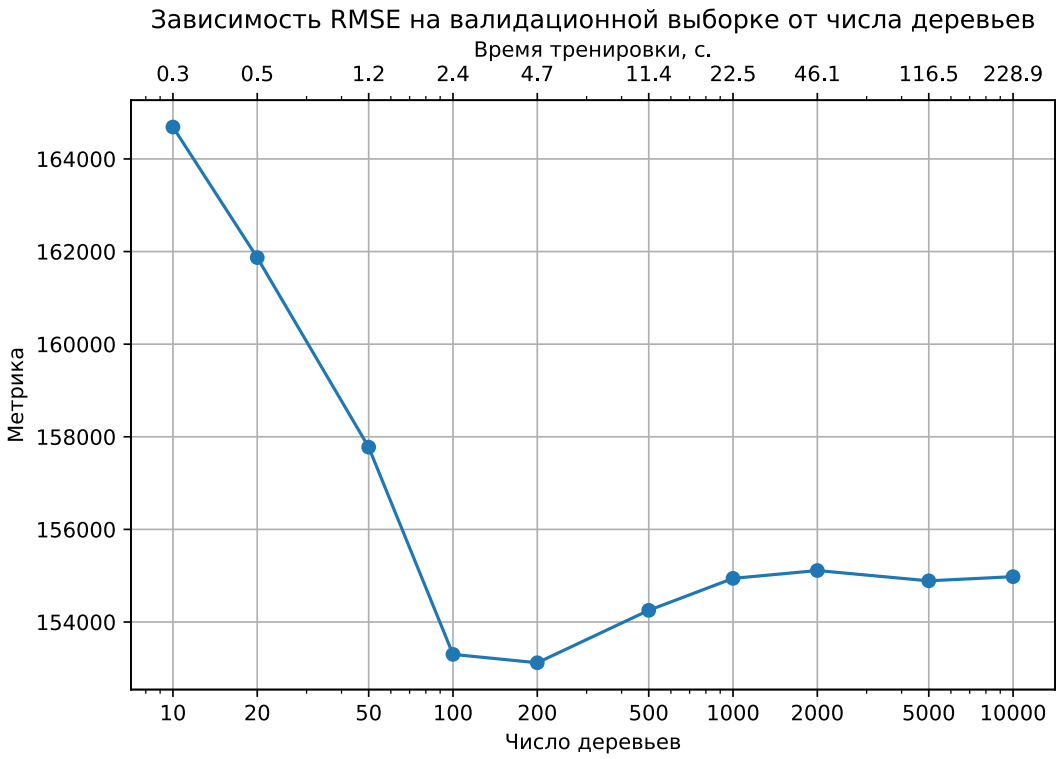

Рис. 1: Результаты экспериментов RandomForest с различным числом базовых алгоритмов. Метрика.

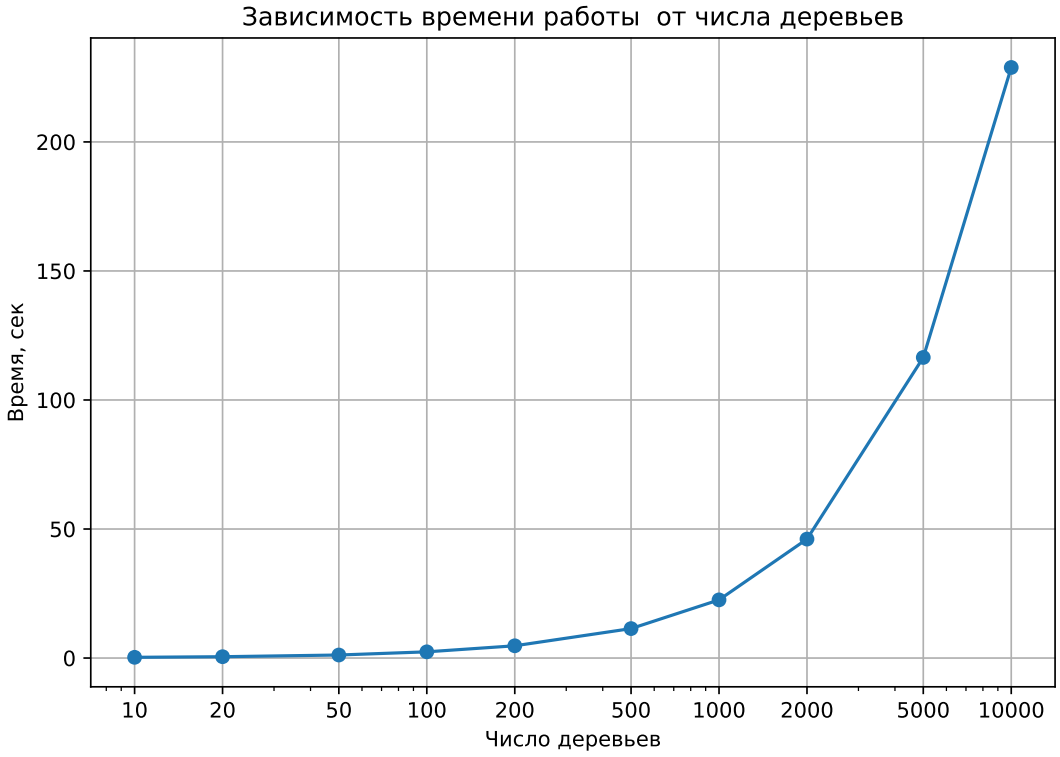

Рис. 2: Результаты экспериментов RandomForest с различным числом базовых алгоритмов. Время работы.

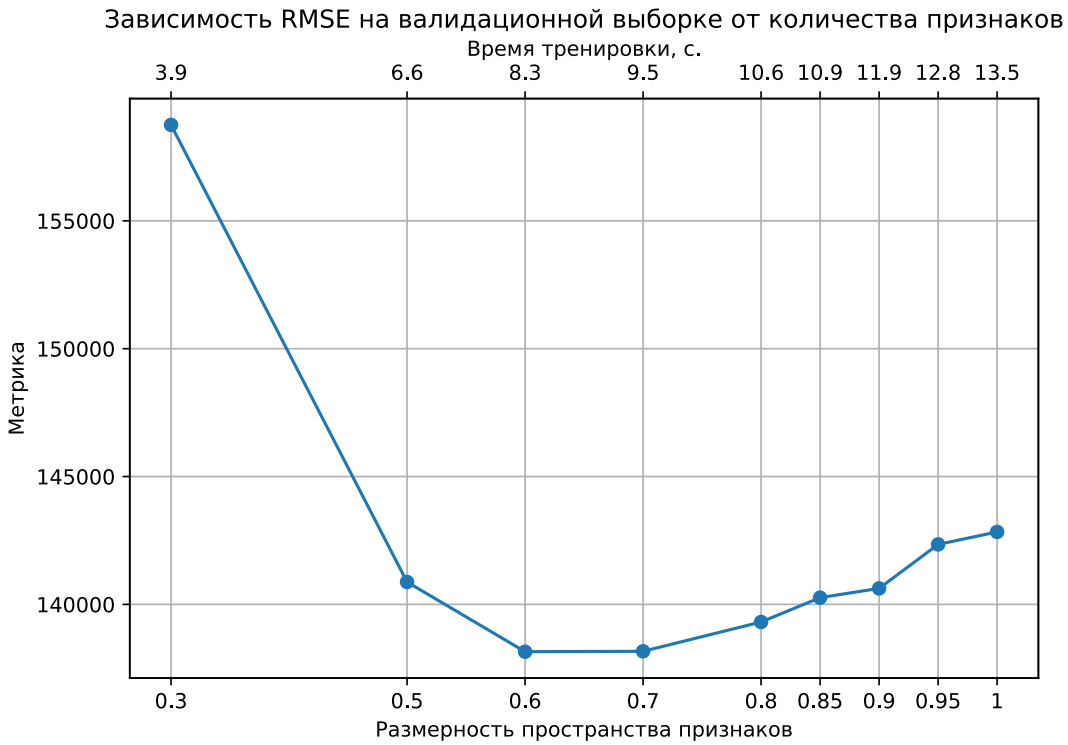

Рис. 3: Результаты экспериментов RandomForest с различным числом признаков. Функция потерь.

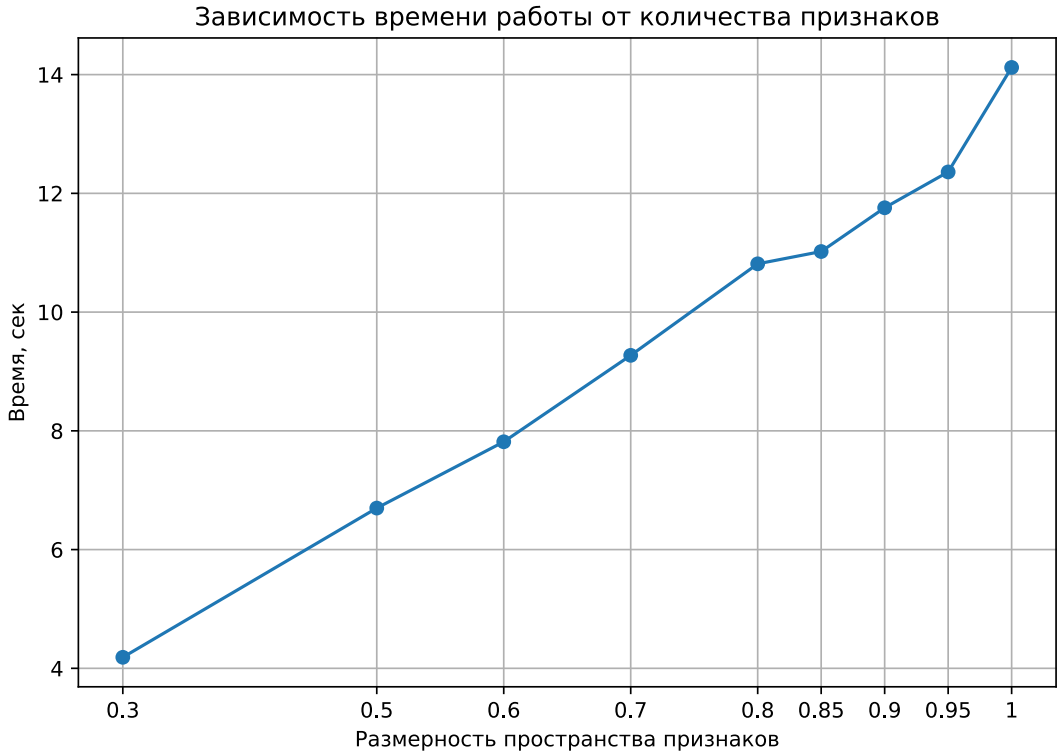

Рис. 4: Результаты экспериментов RandomForest с различным числом признаков. Время работы.

2.1.3 Глубина дерева

Этот гиперпараметр отвечает за сложность базового алгоритма. Более глубокие деревья лучше выучивают зависимости в данных и адаптируются к обучающей выборке. В теории ансамбль из множества глубоких и нейстойчивых деревьев должен становится устойчивым алгоритмом. Мы рассматриваем следующую сетку гиперпараметров: `max_depth` = {None, 3, 5, 6, 7, 8, 10, 12}. Результаты экспериментов представлены на Рис. 5 и 6.

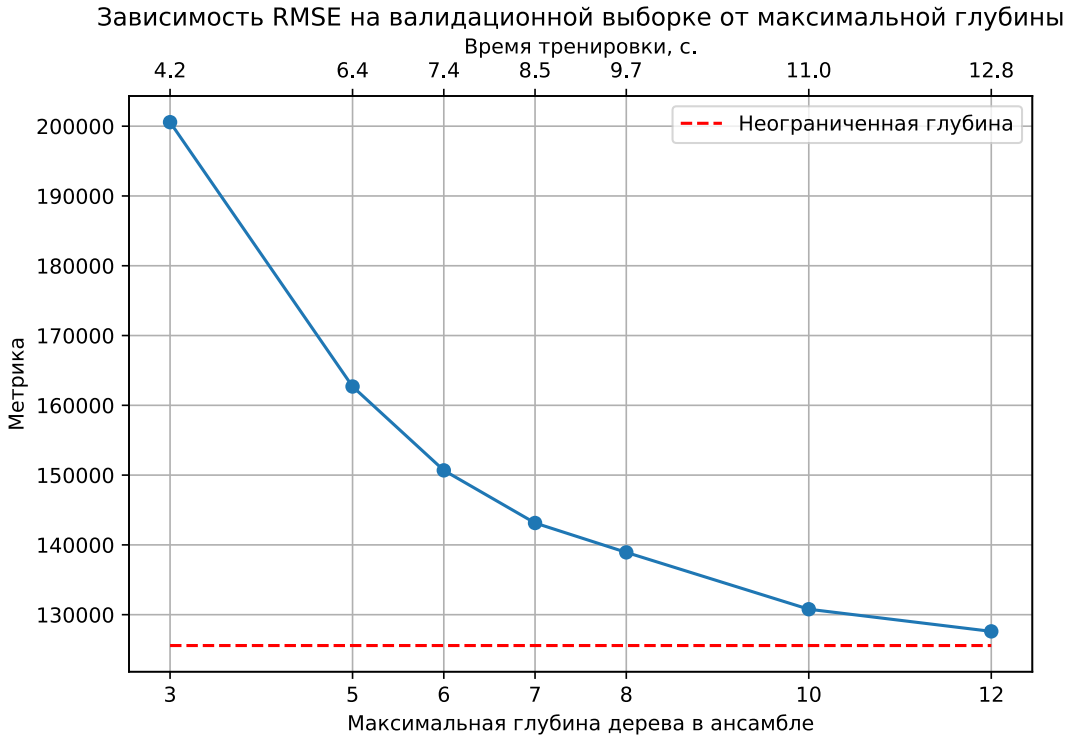

Рис. 5: Результаты экспериментов RandomForest с различной глубиной деревьев. Функция потерь.

Из графиков мы видим, что лучших результатов достигают ансамбли деревьев с бОльшей глубиной, красной линией обозначен уровень качества модели, для которой мы не устанавливали каких-либо ограничений по глубине базовых алгоритмов. Также мы отмечаем, что время работы зависит линейно от глубины базового алгоритма. Из графиков можно заключить, более глубокие деревья - отличный выбор для случайного леса, однако стоит помнить, про то, что и работают они дольше.

2.2 Gradient Boosting

В этой секции мы рассматриваем зависимость бустинга от следующих гиперпараметров:

- `n_estimators` - количество базовых алгоритмов в ансамбле.
- `max_depth` - максимальная глубина каждого дерева.
- `feature_subsample_size` - доля признаков, на которых обучается алгоритм.
- `learning_rate` - темп обучения ансамбля.

2.2.1 Число базовых алгоритмов

Данный параметр показывает, сколько деревьев нам необходимо построить для ансамбля. Будем рассматривать его по следующей сетке: `n_estimators` = {10, 20, 50, 100, 200, 500, 1000,

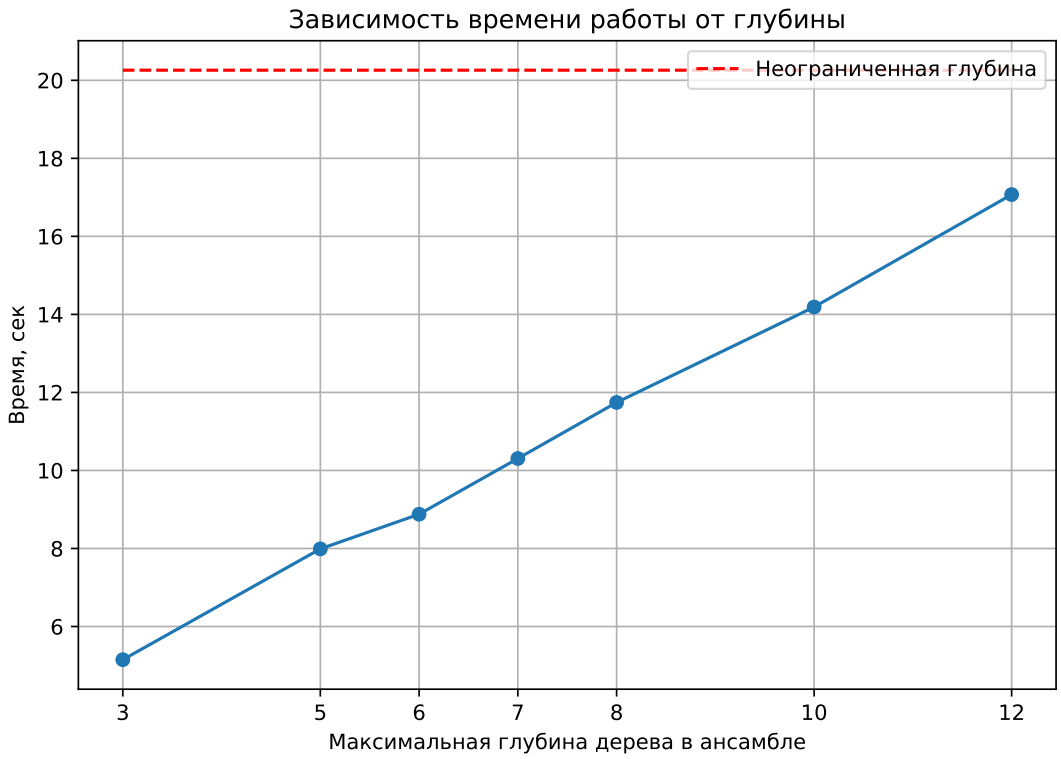

Рис. 6: Результаты экспериментов RandomForest с различной глубиной деревьев. Время работы.

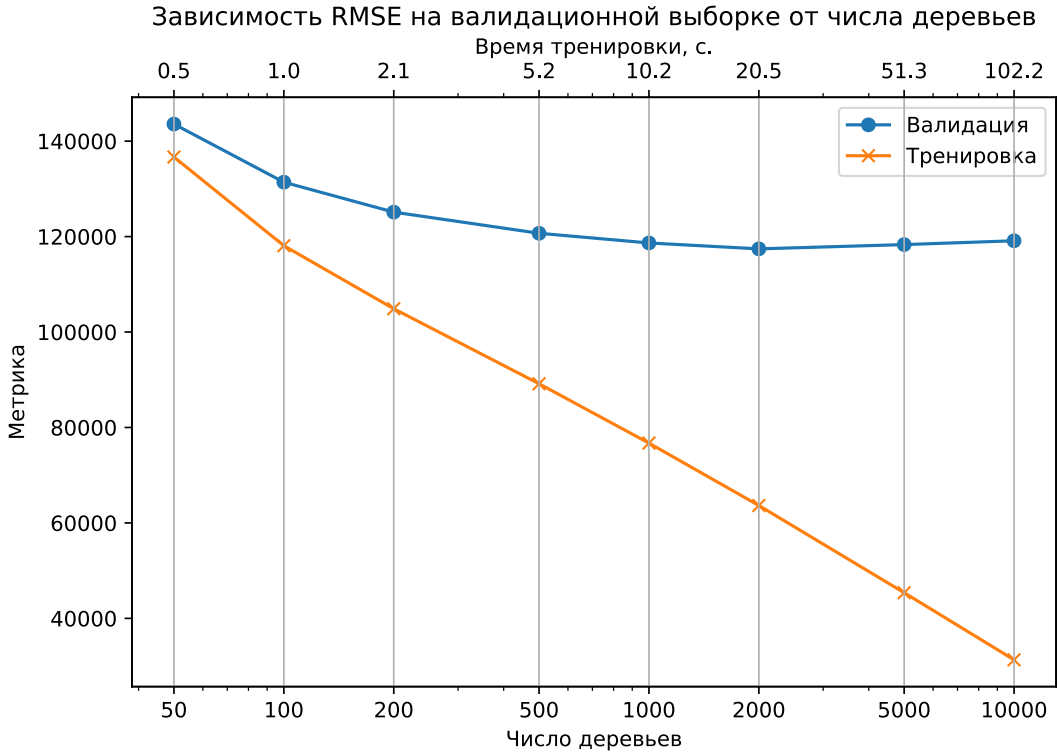

Рис. 7: Результаты экспериментов GradientBoosting с различным числом базовых алгоритмов на валидации и тренировке. Функция потерь.

2000, 5000, 10000}. Кажется, что чем больше мы исправляем себя, тем точнее получается итого-

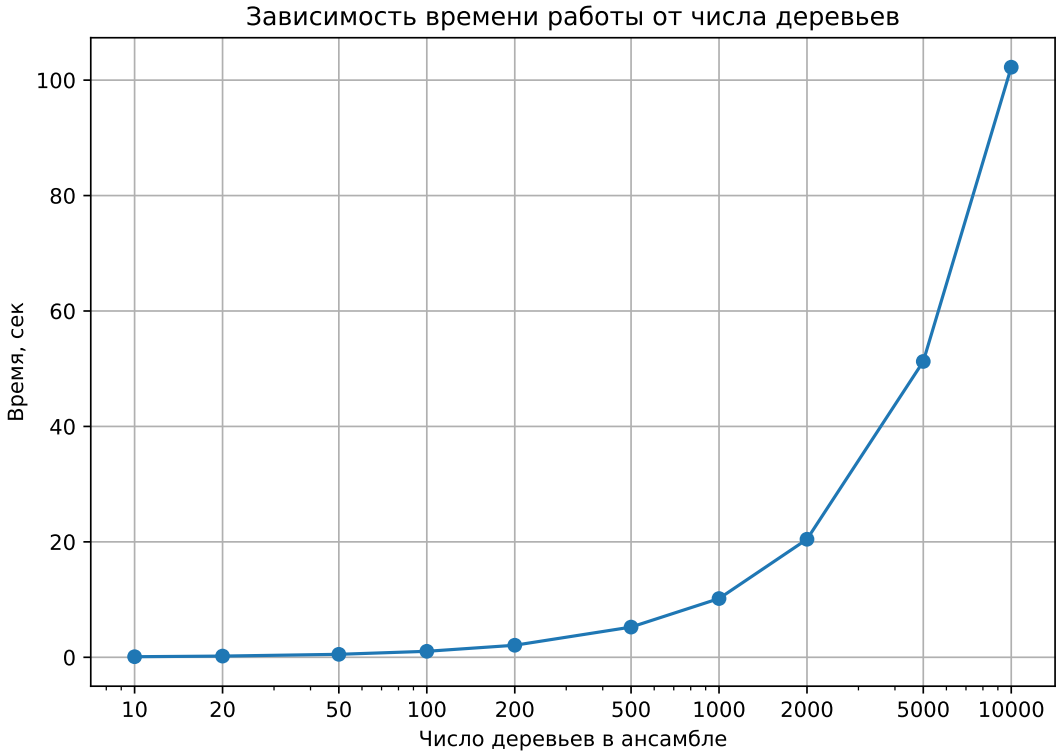

Рис. 8: Результаты экспериментов GradientBoosting с различным числом базовых алгоритмов. Время работы.

вый алгоритм. Результаты наших экспериментов представлены ниже на Рис.7 и 8.

Выводы из наших экспериментов весьма интересные. Мы видим, что минимум на валидации лосс функция достигает при 2000 деревьях. Важно, что мы можем явно видеть, как бустинг переобучается под тренировочный датасет, причем видим при незначительном улучшении на валидации модель значительно переобучается под тренировочный датасет. Также мы снова можем наблюдать линейный рост времени работы алгоритма при росте числа деревьев в ансамбле. Таким образом, большее число деревьев не только ведет к затратам по времени, но и явному переобучению модели. Вероятно, лучшее значе параметра можно выбрать среди значений {500, 1000, 2000}.

2.2.2 Число признаков

Снова рассматриваем гиперпараметр числа признаков для обучения очередного базового алгоритма. Мы рассматриваем следующую сетку данного гиперпараметра: `feature_subsample_size` = {0.3, 0.5, 0.6, 0.7, 0.8, 0.85, 0.9, 0.95, 1}. Результаты экспериментов приведены на Рис.9 и 10.

Интересно, что лосс функция не зависит монотонно от размерности признакового пространства и достигает минимума при значении `feature_subsample_size` = 0.7. Как и для деревьев зависимость времени работы от размерности признаков - близкая к линейной. Снова можно заключить, что для ансамбля алгоритмов важно, чтобы они были не слишком скореллированы и не слишком разными, для того чтобы вместе показывать наилучший результат.

2.2.3 Глубина дерева

Этот гиперпараметр отвечает за сложность базового алгоритма. В бустинге мы хотим поправлять ошибки предыдущих алгоритмов каждым следующим. Исходя из этого, кажется, что, для того чтобы исправлять много не пришлось, каждый из базовых алгоритмов не должен совершать много ошибочных разбиений. Таким образом, вероятно, лучше должны работать ансамбли с менее глубокими деревьями. Проверим наши догадки экспериментально. Мы рассматриваем

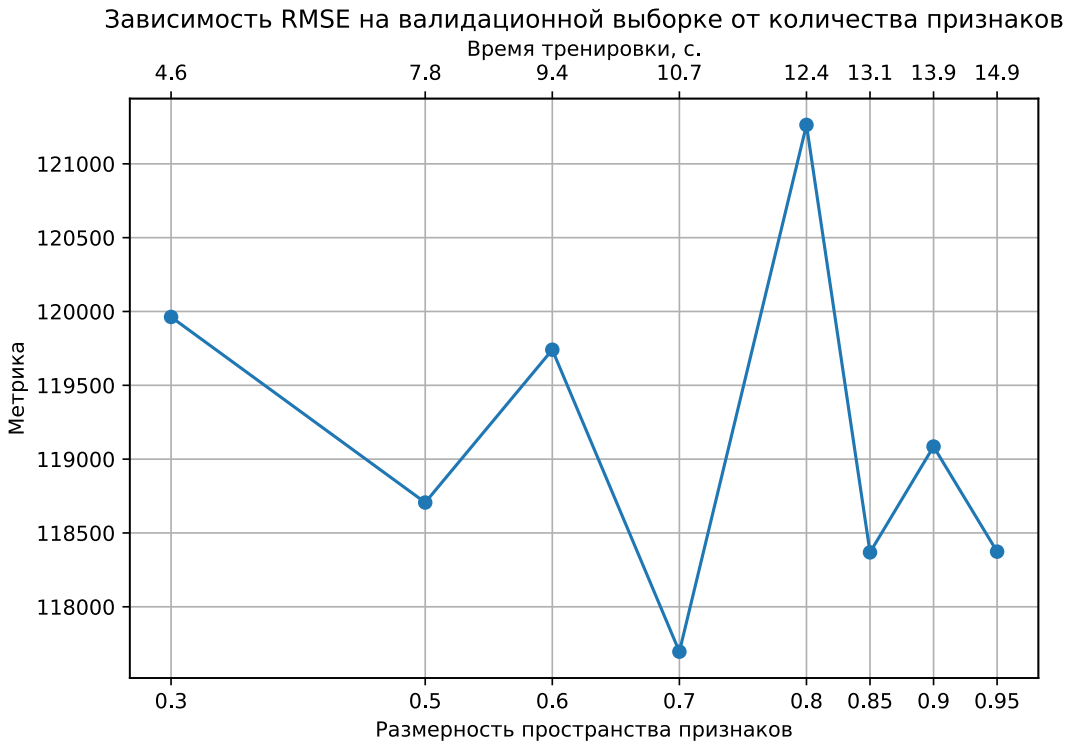

Рис. 9: Результаты экспериментов GradientBoosting с различным числом признаков. Функция потерь.

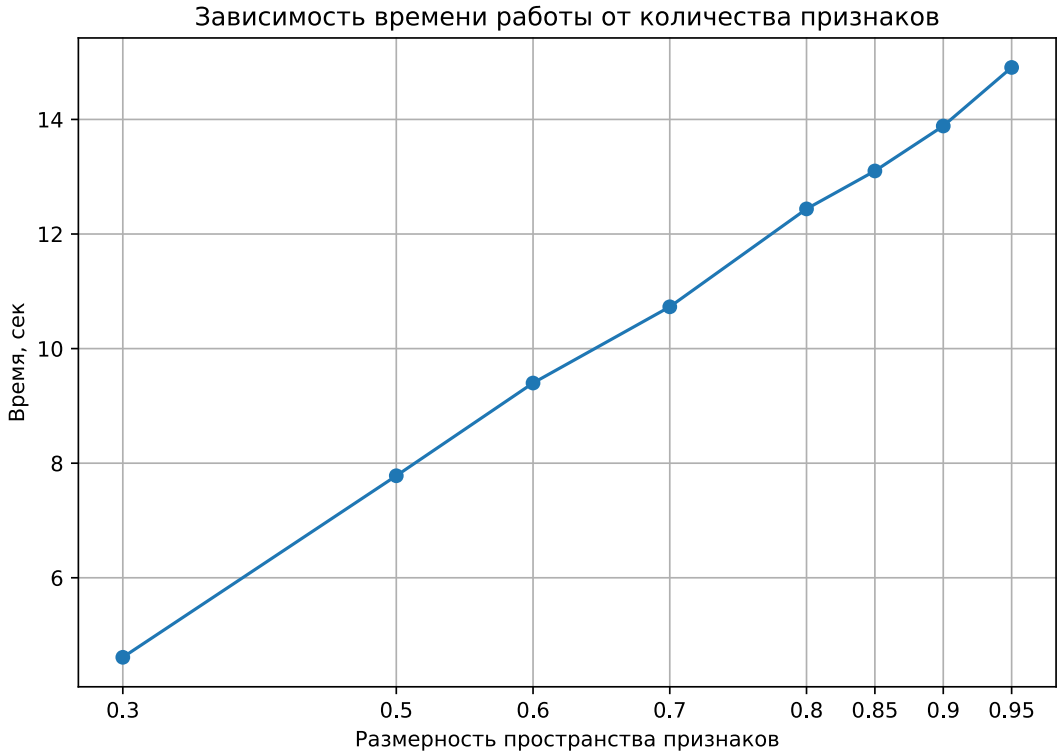

Рис. 10: Результаты экспериментов GradientBoosting с различным числом признаков. Время работы.

следующую сетку гиперпараметров: `max_depth` = {1, 2, 3, 4, 5, 8, 10}. Результаты экспериментов представлены на Рис. 11 и 12.

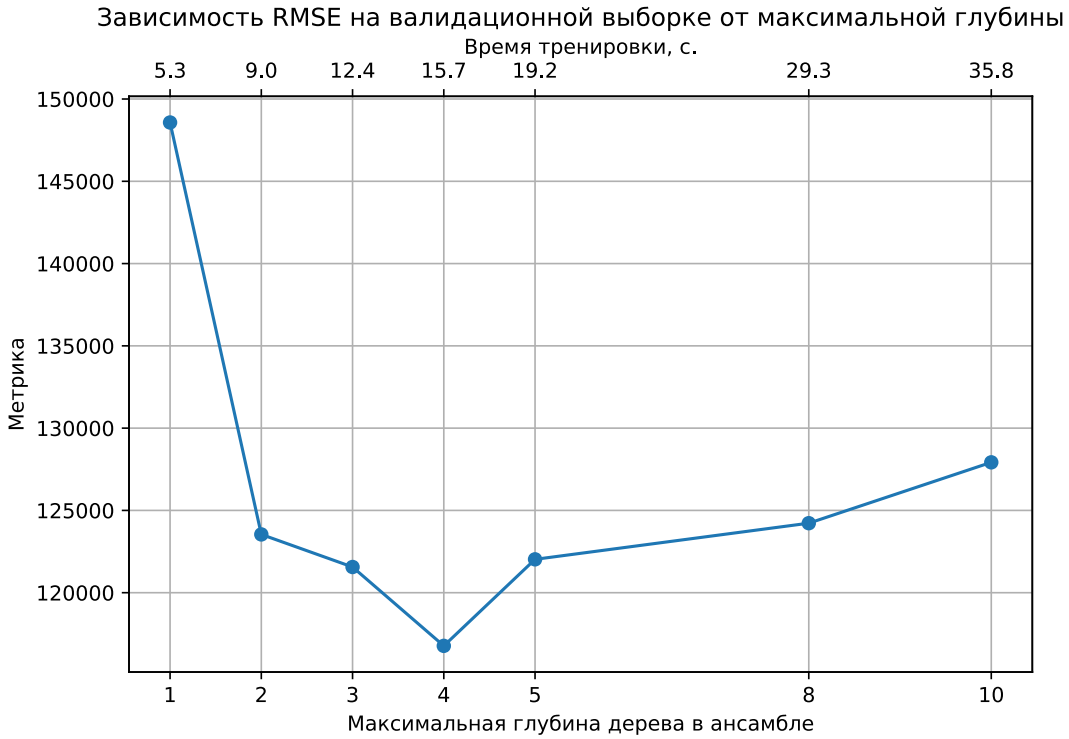

Рис. 11: Результаты экспериментов GradientBoosting с различной глубиной деревьев. Функция потерь.

Результаты оказались весьма предсказуемы. Минимум функции потерь достигается с гиперпараметром `max_depth` = 4. Важно заметить, что ансамбли с меньшей глубиной базового алгоритма оказываются «глупыми», а слишком глубокие, вероятно, сильно переобучаются. Время все также линейно зависит от глубины алгоритма. Мы заключаем, что для бустинга деревья должны быть значительно менее глубокими, нежели для случайного леса, что делает их более быстрыми в обучении.

2.2.4 Темп обучения

Данный параметр отвечает за то, насколько мы «доверяем» каждому новому базовому алгоритму. Отклик каждого из базовых алгоритмов масштабируется этим гиперпараметром, таким образом, делая алгоритм более «осторожным» на каждом шаге обучения. В наших экспериментах идет перебор по следующей сетке `learning_rate` = {1, 0.1, 0.01, 0.001, 0.0001}. Результаты экспериментов представлены на Рис. 13, 14.

По результатам экспериментов мы видим, что слишком малый learning_rate ведет к недообучению, а слишком большой - к более грубому решению. Оптимальным в наших экспериментах стало значение гиперпараметра `learning_rate` = 0.1. Заметим, что время работы возросло, но очень незначительно, сложно объяснить с чем это связано.

2.3 RandomForest vs. GradientBoosting

В этой секции мы сравним между собой два алгоритма ансамблирования. Можно заметить, что оба алгоритма неустойчивы к изменению параметров, разница в перфомансе моделей может быть довольно значительной на соседних значениях гиперпараметра. Оба алгоритма показывают линейный рост времени работы при увеличении значения рассматривавшихся параметров. Отличия в следующем: в нашей задаче бустинг оказался несколько лучше, построение бустин-

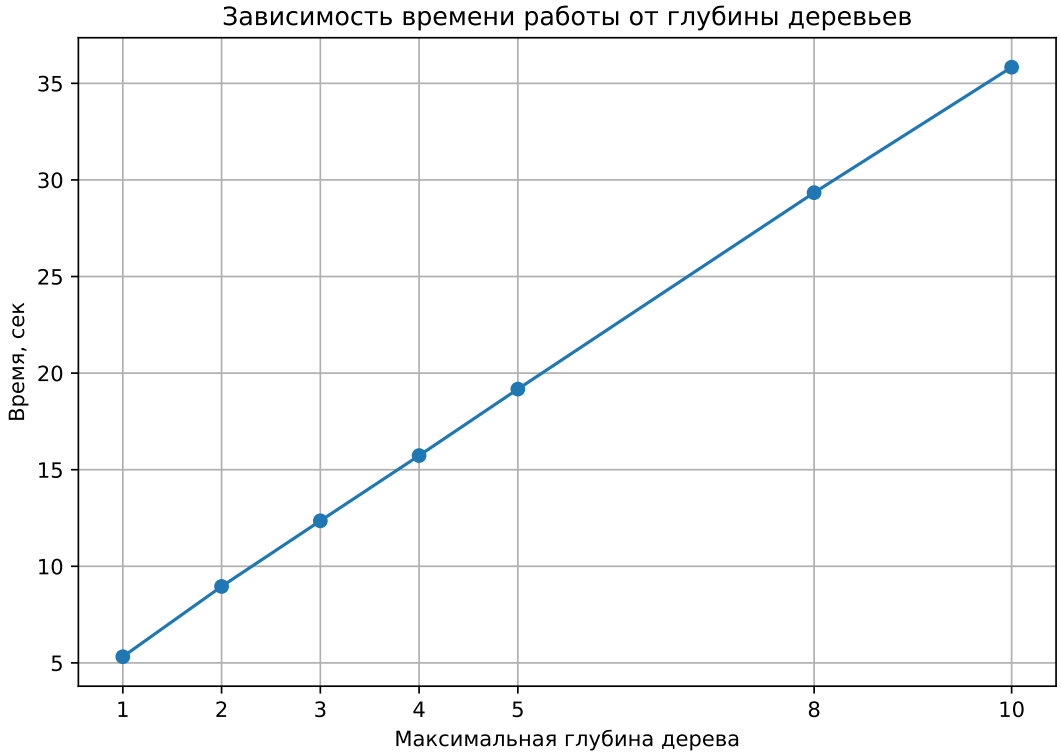

Рис. 12: Результаты экспериментов GradientBoosting с различной глубиной деревьев. Время работы.

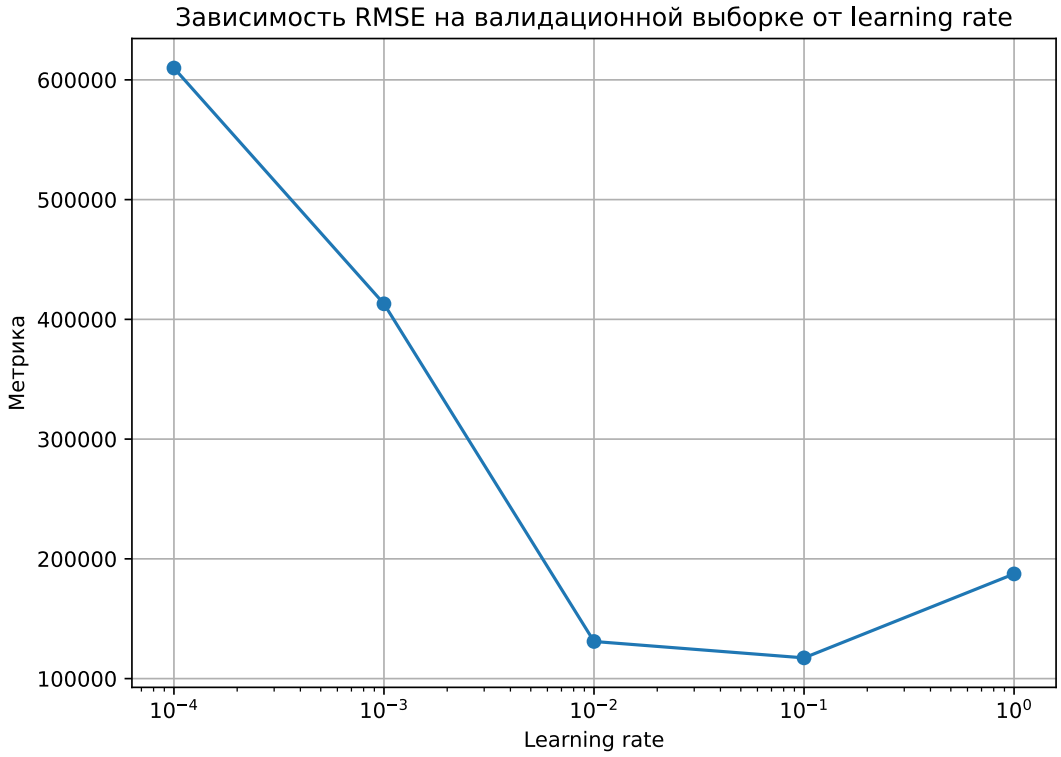

Рис. 13: Результаты экспериментов GradientBoosting с различным темпом обучения. Функция потерь.

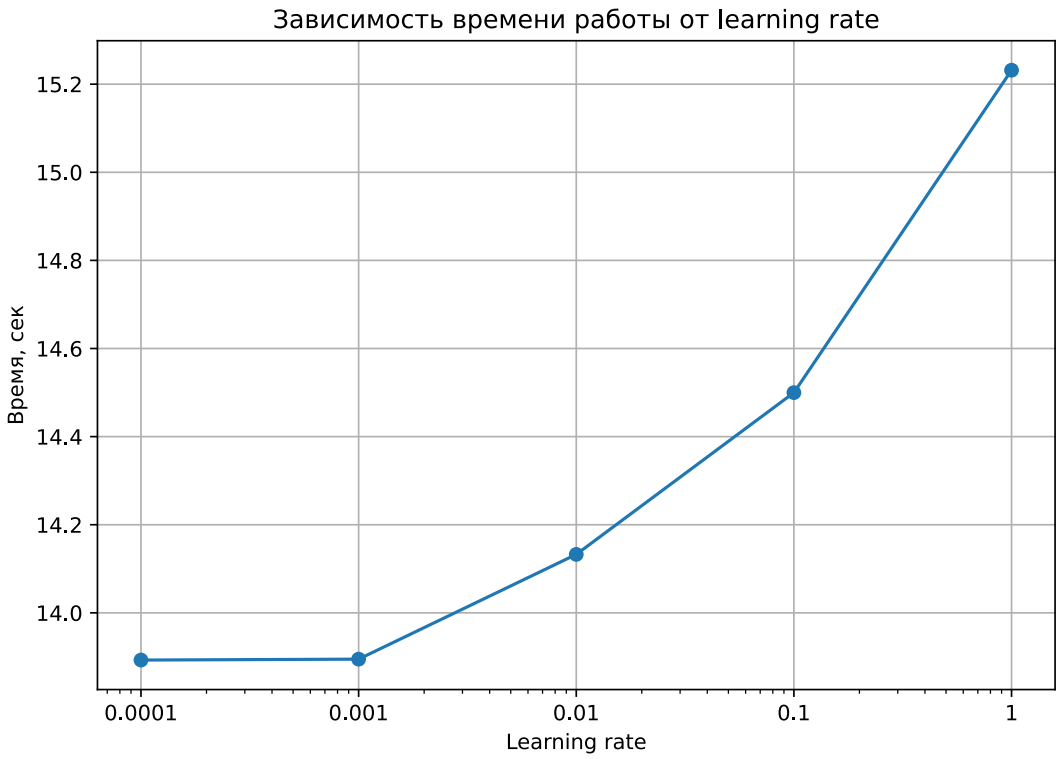

Рис. 14: Результаты экспериментов GradientBoosting с различным темпом обучения. Время работы.

га быстрее, нежели случайного леса, так как мы строим менее глубокие деревья. Однако этого недостатка можно избежать, если строить деревья параллельно.

3 Выводы

Мы реализовали два алгоритма ансамблирования. Внимательно рассмотрели оба, каждый из них имеет свои достоинства и недостатки. Был проведен качественный анализ зависимости каждого из алгоритмов от гиперпараметров на предложенном датасете в задаче регрессии. Дополнительно мы сравнили алгоритмы между собой.