Московский Государственный Университет

Композиции алгоритмов для решения задачи регрессии

Выполнил: Курцев Д.В.
Группа: 317

Факультет Вычислительной математики и кибернетики
Кафедра Математических методов прогнозирования

Декабрь 2021

Введение

В данном задании были реализованы и разобраны такие модели, как случайный лес (RandomForest) и градиентный бустинг (GradientBoosting). В их основе лежит алгоритм решающего дерева, который мы сами не писали, а взяли готовый из библиотеки scikit-learn (sklearn.tree.DecisionTreeRegressor).

Пусть у нас есть матрица объектов-признаков $X \in \mathbb{R}^{N \times D}$ и вектор значений $y \in \mathbb{R}^N$.

Оновная идея леса заключается в том, что строится l деревьев. Каждое из них обучается на бутстрапированной выборке $X_i \in \mathbb{R}^{N \times d}$ с d признаками. По умолчанию $d = \lfloor \frac{D}{3} \rfloor$, где D - размерность признакого пространства X. Таким образом, выборки получаются менее скорелированными, что улучшает работу алгортима. Предсказания строятся, как взвешенная сумма полученных деревьев: $a(x) = \frac{1}{l} \sum\limits_{i=1}^{l} b_i(X)$, где $b_i(X)$ - те самый деревья.

Градиентный бустинг также для обучения использует деревья. Только его идея состоит в том, чтобы каждый последующий алгоритм исправлял ошибки предыдущего. Таким образом, обучение t-ой модели происходит следующим образом: $\sum\limits_{i=1}^{N} \mathscr{L}(y_i, a_{t-1}(x_i) + \alpha_t \, b_t(x_i)) \to \min\limits_{b_t, \alpha_t}$. Где $\mathscr{L}$ - это функция потерь (мы использовали MSE), а a_{t-1} - наша модель на $t-1$ - ом шаге. То есть после l итераций получим итоговую модель $a_l = \sum\limits_{i=1}^{l} \alpha_i b_i(X)$.

Проанализируем как работают наши модели в зависимости от разных гиперпараметров. Рассмотрим задачу регрессии. Будем предсказывать стоимость дома в Америке. Все эксперименты проводились на датасете, включающим в себя 21613 объектов.

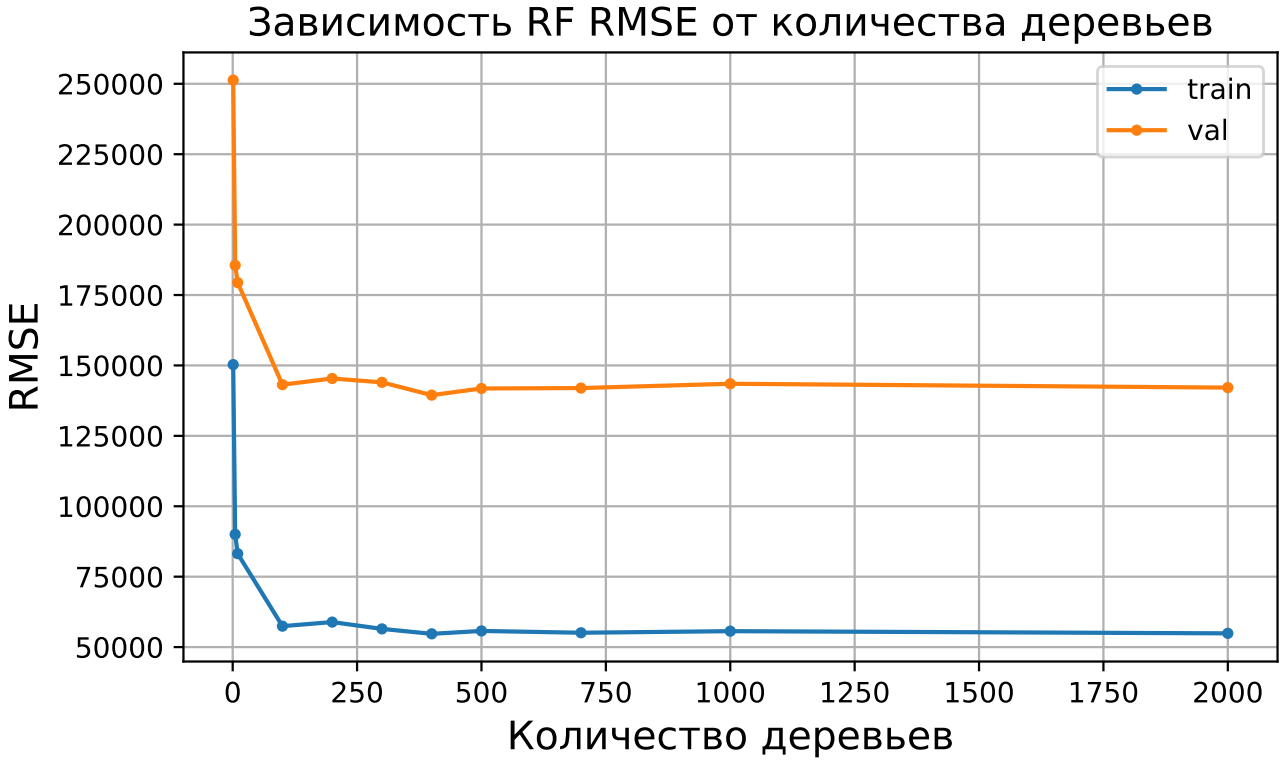
Для начала запишем значение $target$ в отдельную переменную и удалим его из выборки. Далее удалим признаки "id"и "date" , так как они не несут никакой информации о стоимости дома. Проверим, что в нашем датасете нет пропусков. Разделим выборку на

обучающую и тестовую в соотношении 8:2.

Random Forest

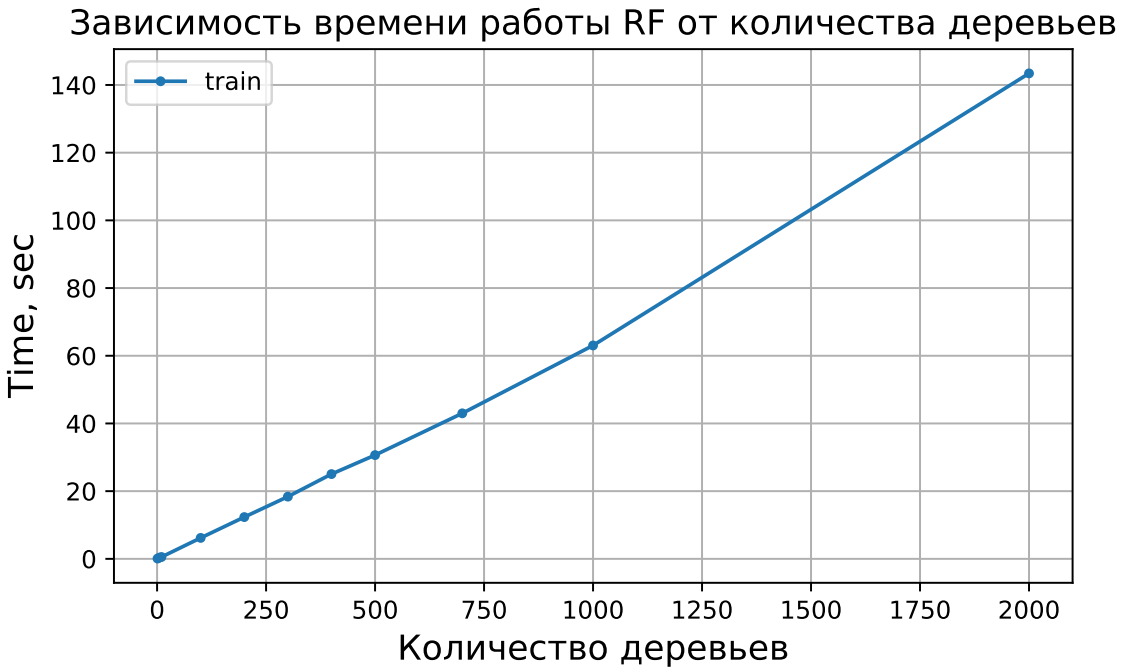
Проанализируем качество работы RandomForest. Будем использовать RMSE, как функцию ошибки. Подберём гиперпараметры для леса. Для этого разобьём случайным образом обучающую выборку на валидационную и обучающую в отношении 2:8.

Сначала выберем подходящее количество деревьев ($n\_estimators$). Оставим остальные параметры по умолчанию: $max\_depth = None$ (глубина), $feature\_subsample\_size = D/3$ (количество признаков используемых для обучения одного дерева), в нашем случае 6. Запустим модель для различных значений количества деревьев и посмотрим на результаты.



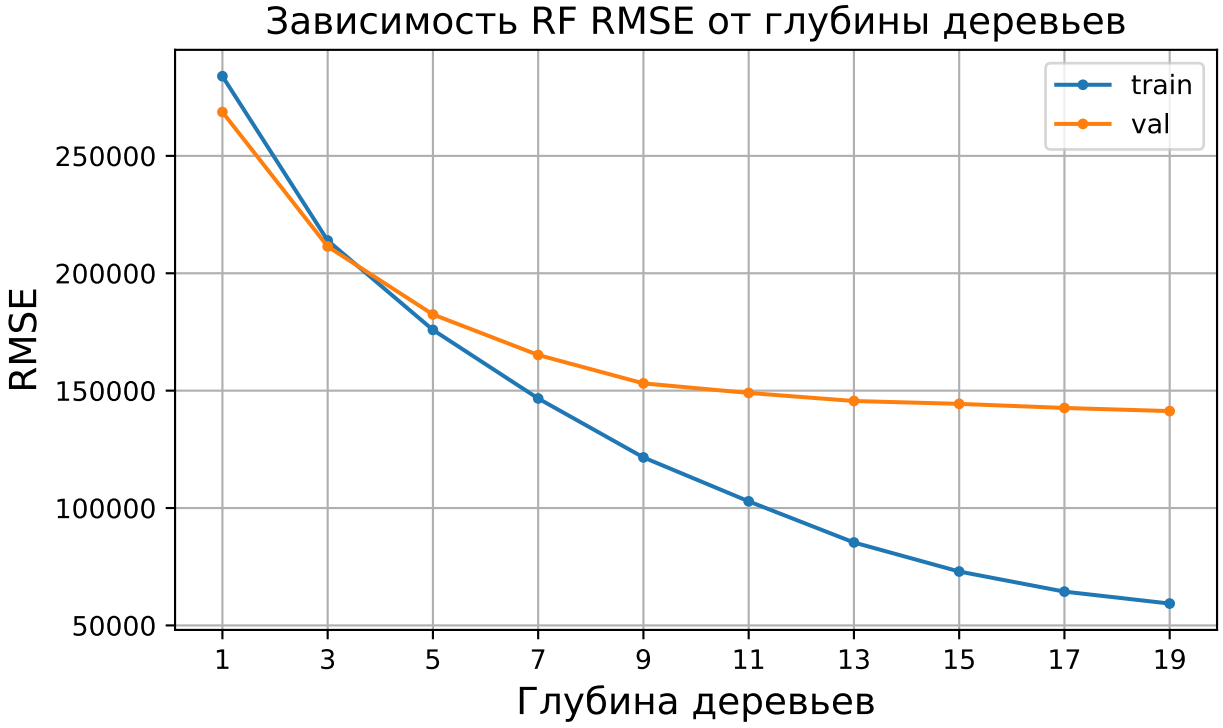
$max\_depth = None, feature\_subsample\_size = 6$

Видим что, при малом количестве деревьев модель плохо обучается. С ростом их числа ошибка уменьшилась больше чем в 2 раза. Но после 200 RMSE выходит на плато и перестаёт уменьшаться. Также заметим, что время работы алгоритма растёт ли-

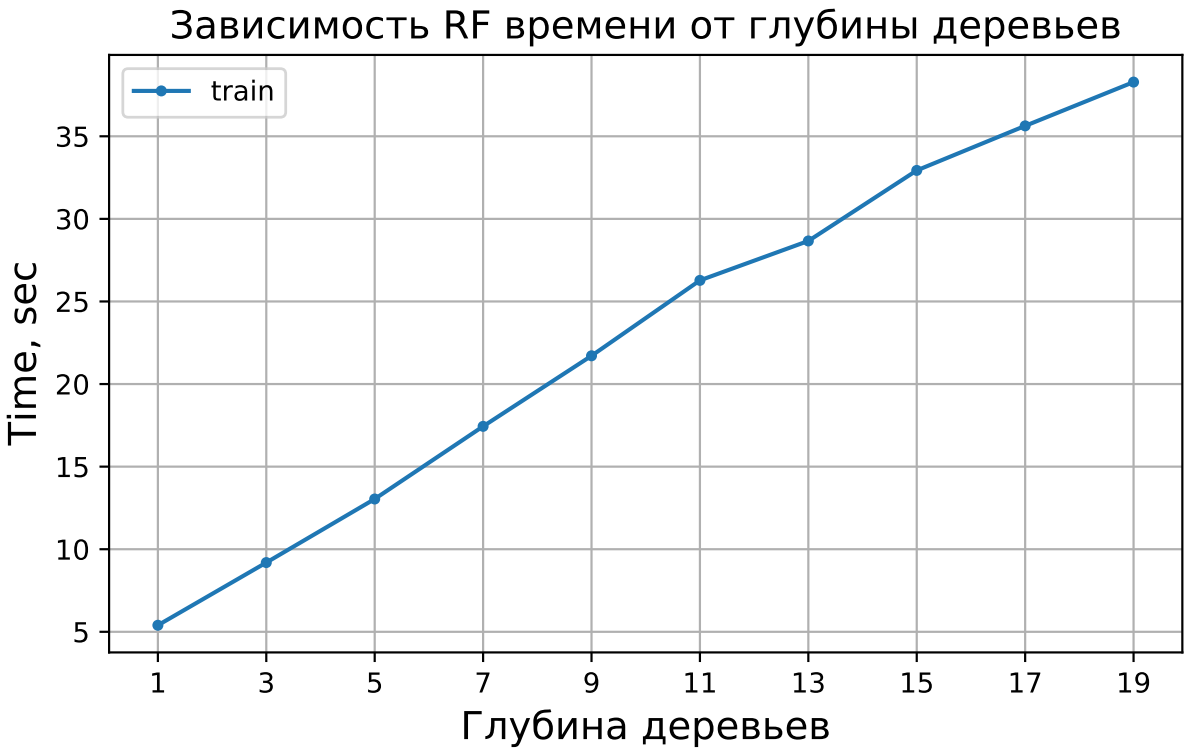


нейно с увеличением количества деревьев в лесе. Так как из теории известно, что большее количество деревьев позволяет уменьшить разброс модели, то получим, что ошибка должна уменьшиться. Поэтому дальше будем брать чуть больше деревьев - 700.

Теперь исследуем качество работы метода в зависимости от глубины.

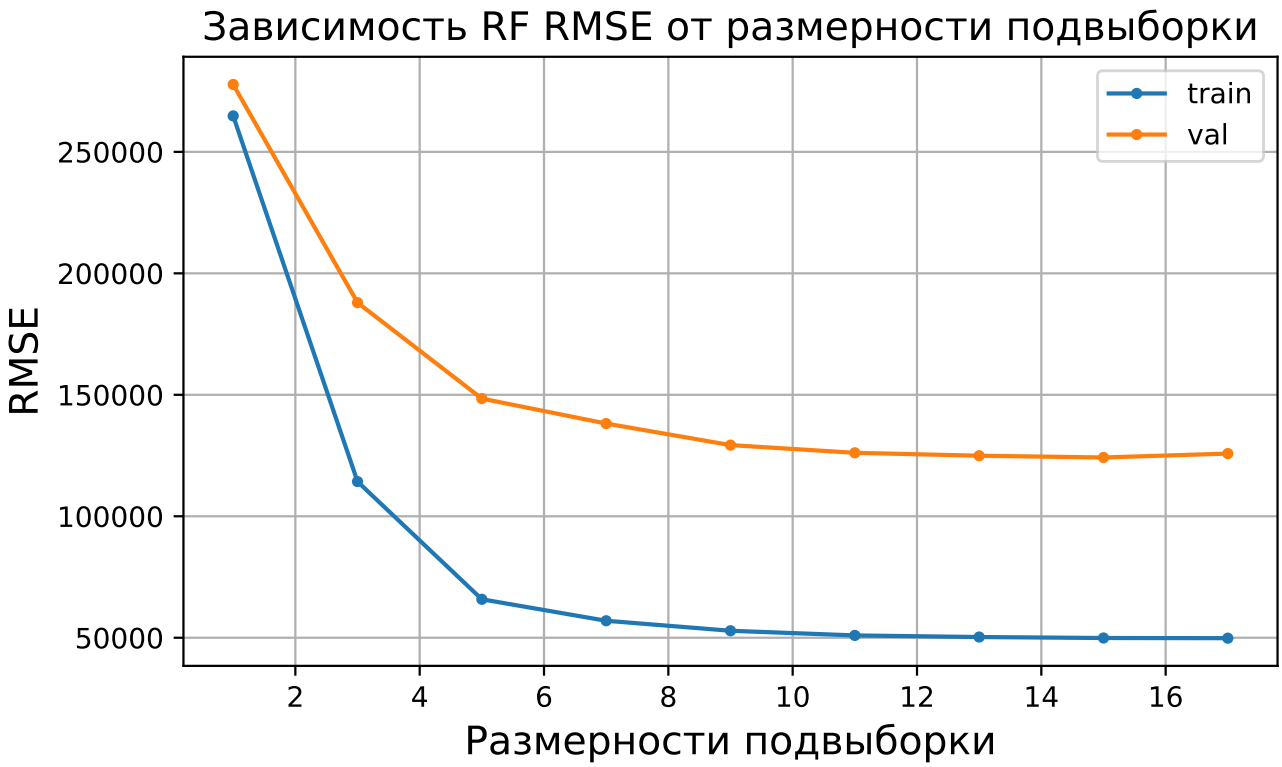


$n\_estimators = 700, feature\_subsample\_size = 6$

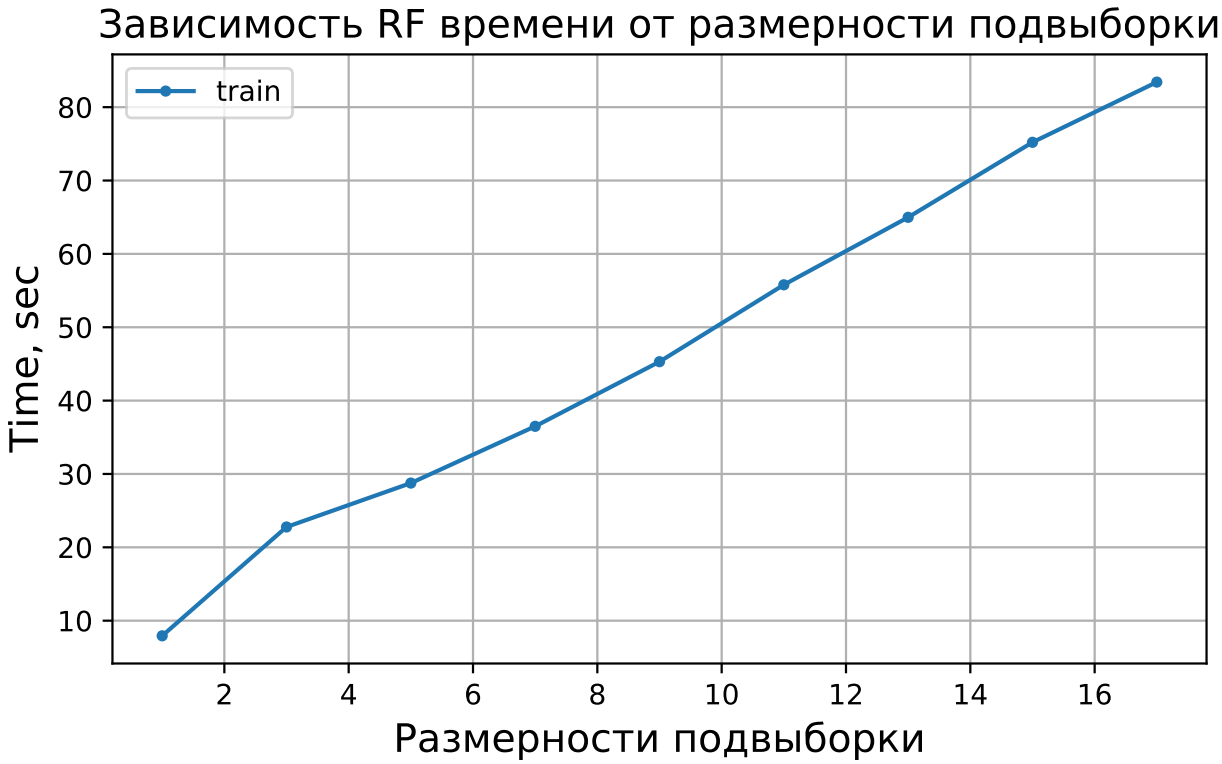


Видим, что при малой глубине модель довольно плохо обучается, так как деревья получаются слишком простые и они не успевают полностью найти все зависимости в данных. С ростом этого гиперпараметра модель получается более гибкой. То есть она может распознать более сложные закономерности. Качество на валидации перестаёт меняться примерно с 15. Такую глубину мы дальше и будем использовать. Заметим, что ошибка примерно равна ошибке, полученной при подборе количества деревьев (там ограничения на глубину мы не накладывали). Но с ростом глубины лесу так же свойственно сильно переобучаться под обучающую выборку. Время зависимости от глубины так же линейно.

Наконец, проверим, как влияет на ошибку размерность признакового пространства для одного дерева.



$n\_estimators = 700, max\_depth = 15$



Здесь можно сделать аналогичные выводы, что и в предыдущих пунктах. При малом количестве признакового пространства, модель плохо улавливает зависимости. А при слишком большом

довольно сильно переобучается. Время зависимости так же практически линейно. Дальше будем использовать 11 признаков.

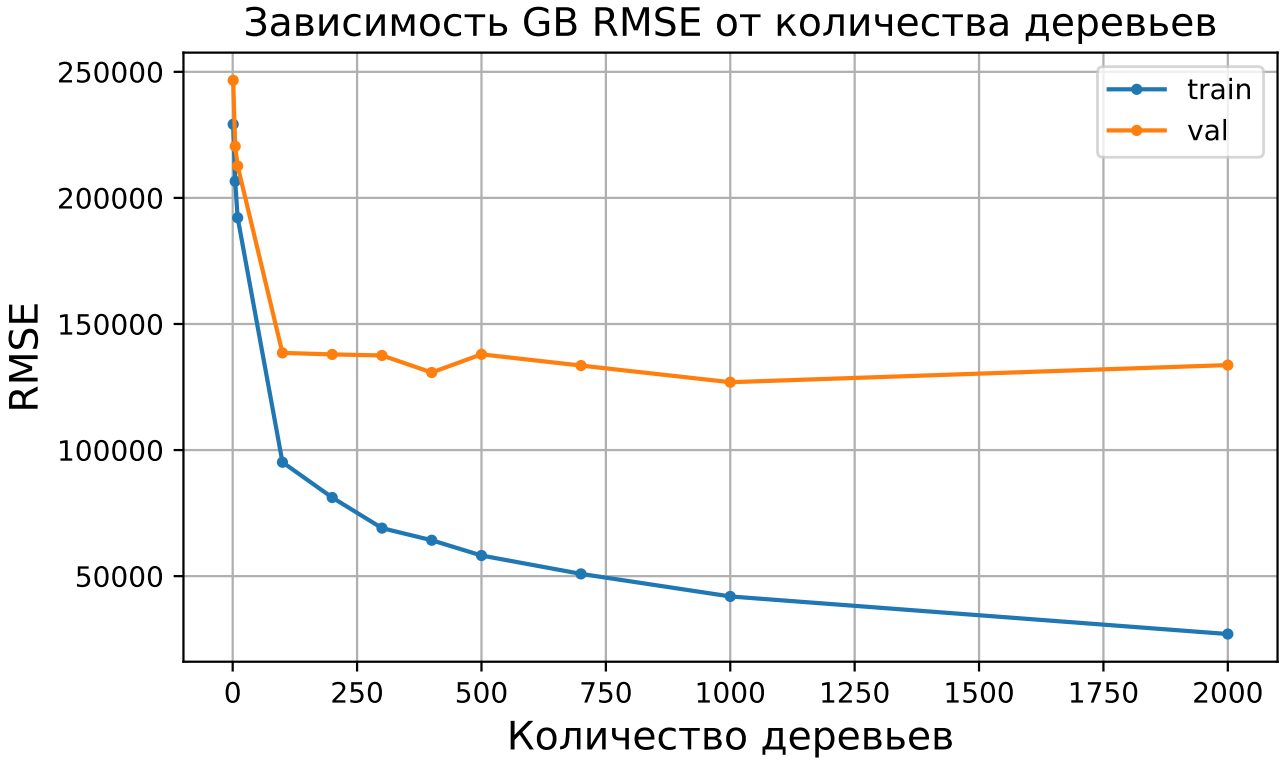
Мы подобрали гиперпараметры по валидации. Теперь обучим модель на всей выборке и посмотрим на качество работы алгоритма на тесте. Возьмём $n\_estimators = 700$ (количество деревьев в лесе), $max\_depth = 15$ (их глубина), $feature\_subsample\_size = 11$ (количество признаков подвыборки для одного дерева).

Ошибка на тесте составила примерно 133644, а время работы 94 секунды. Видим, что качество примерно такое же, как и на валидации, а время чуть увеличилось (из-за того, что мы теперь используем все данные). Заметим, что если запустить модель с 700 деревьями, оставив остальные параметры по умолчанию, то качество будет примерно 151416. Таким образом, подобрав гиперпараметры, можно существенно улучшить качество работы алгоритма.

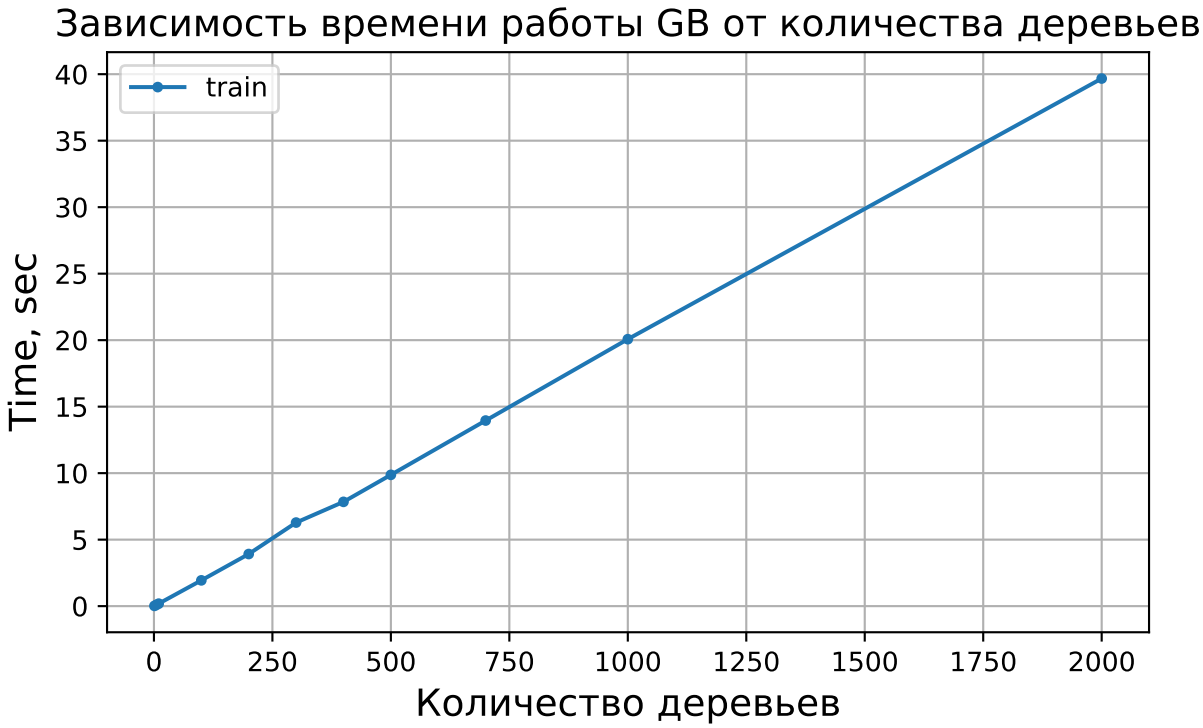
Gradient Boosting

Теперь проанализируем качество работы GradientBoosting. Точно так же подберём гиперпараметры.

Аналогично начнём подбор с $n\_estimators$ (количества деревьев). Оставим остальные параметры по умолчанию: $max\_depth = None$ (глубина), $feature\_subsample\_size = D/3 = 6$ (размер признакового пространства) и $learning\_rate = 0.1$ (темп обучения). Запустим модель для различных значений количества деревьев и посмотрим на результаты на валидации.

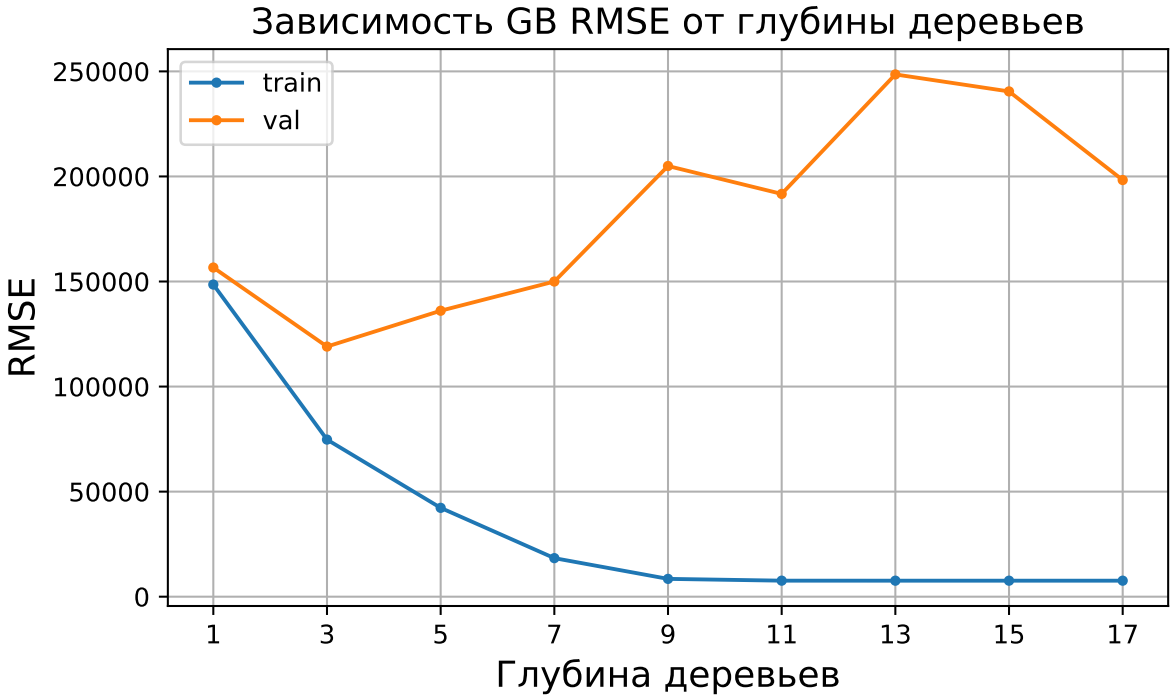


$md = 5, fss = 6, lr = 0.1$

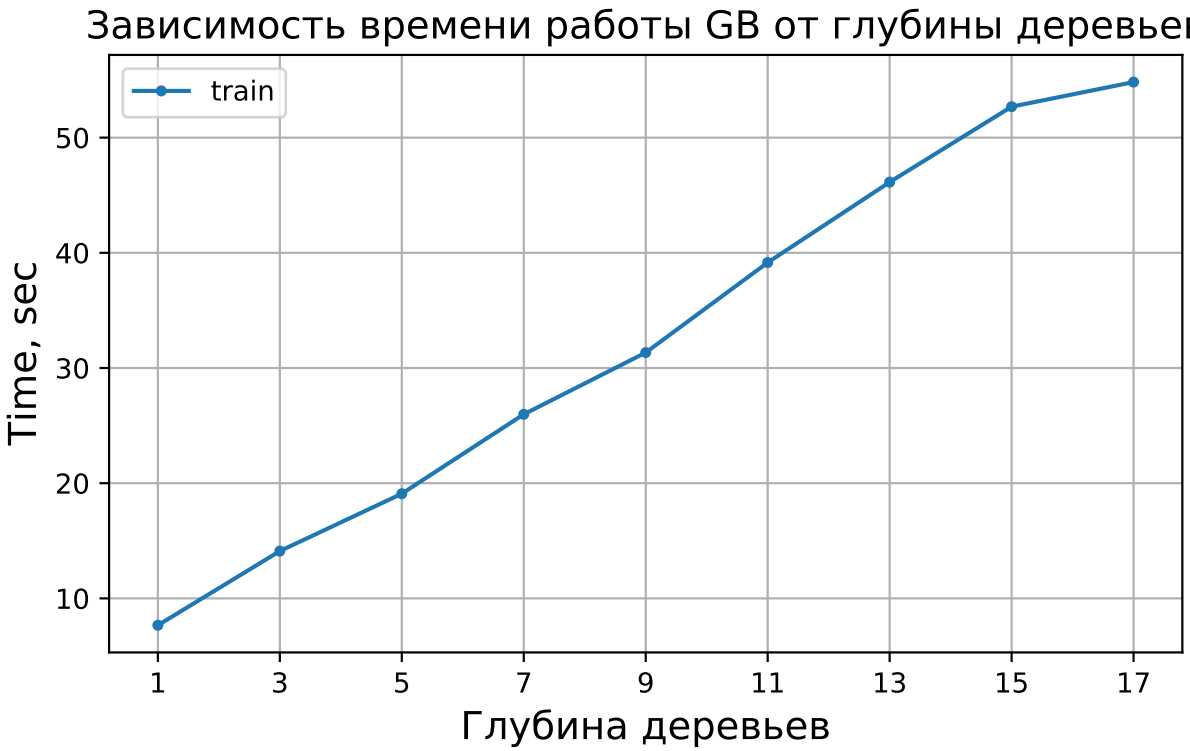


Тут можно сделать аналогичные выводы, что и для алгоритма RandomForest. Наилучшим значением на валидации стало 1000 деревьев. Столько мы дальше и будем использовать.

Теперь исследуем качество работы метода в зависимости от глубины.

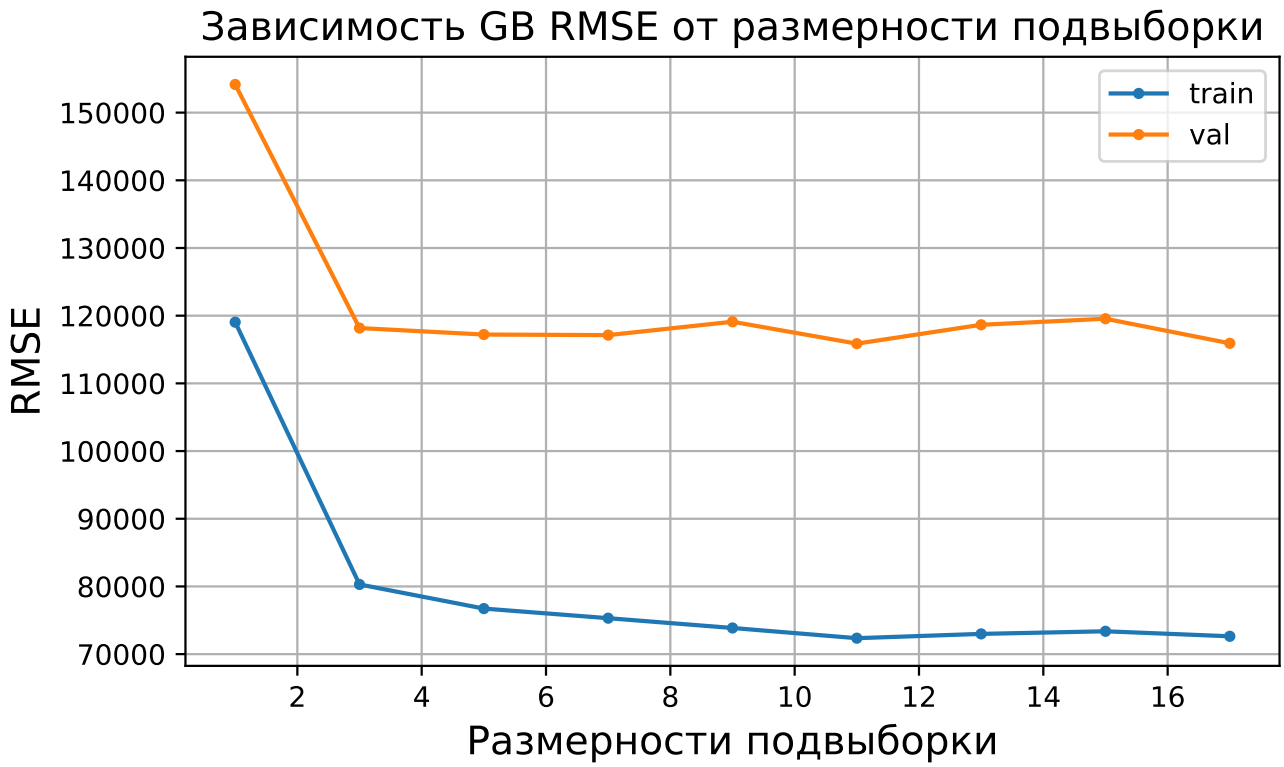


$n = 1000, fss = 6, lr = 0.1$

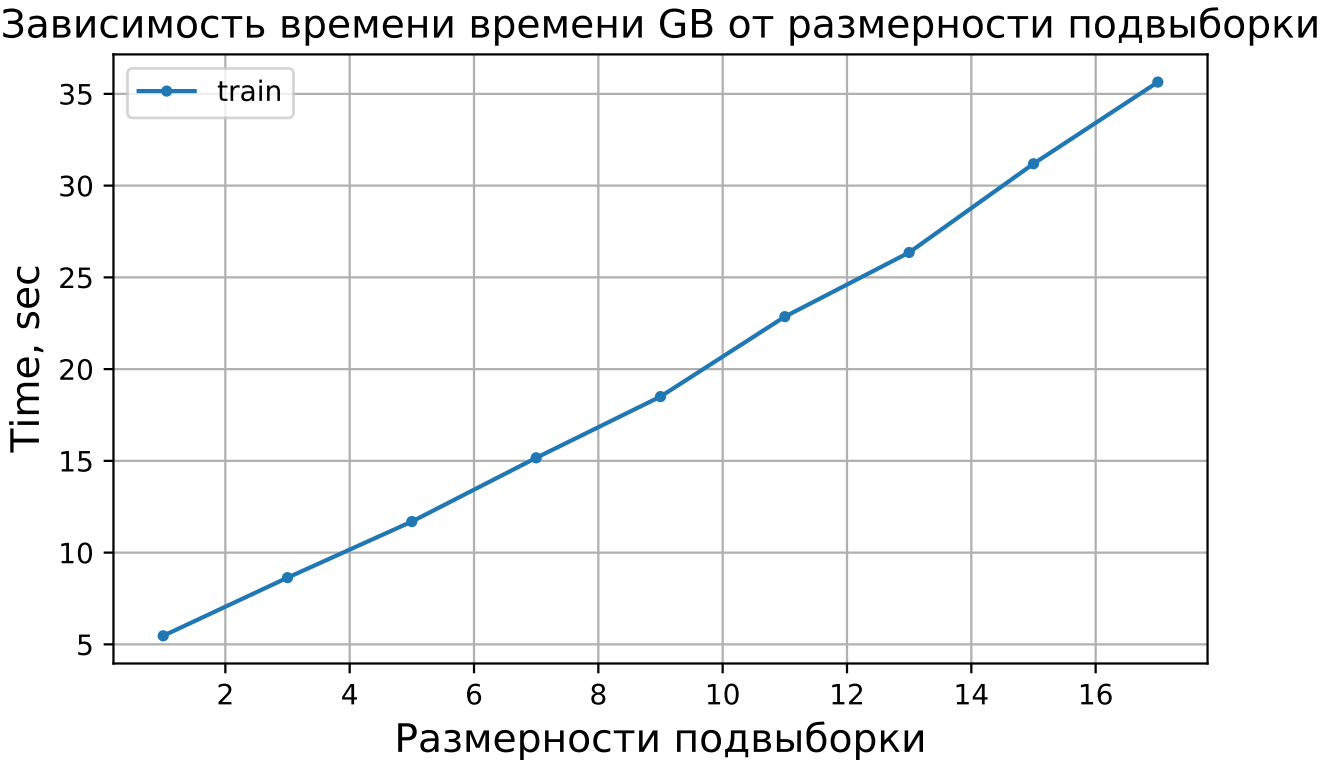


Видим, что в случае с градиентным бустингом ситуация немного другая, чем с лесом. С ростом глубины деревьев ошибка на обучающей выборке падает, но на валидационной возрастает. Вероятно, это связано с тем, что каждое последующее дерево будет довольно сложным. Следовательно оно будет лучше исправлять ошибки предыдущих деревьев. Таким образом, наша модель будет очень сильно подстраиваться под обучающую выборку. То есть будет сильно переобученной. Заметим, что RMSE у градиентного бустинга падает намного быстрее, чем у леса. Отметим, что наилучшим значением этого параметра является 3. Его мы дальше и будем использовать. Видно так же, что зависимость времени работы модели от глубины линейно.

Теперь проверим, как влияет на ошибку размерность признакового пространства для одного дерева.



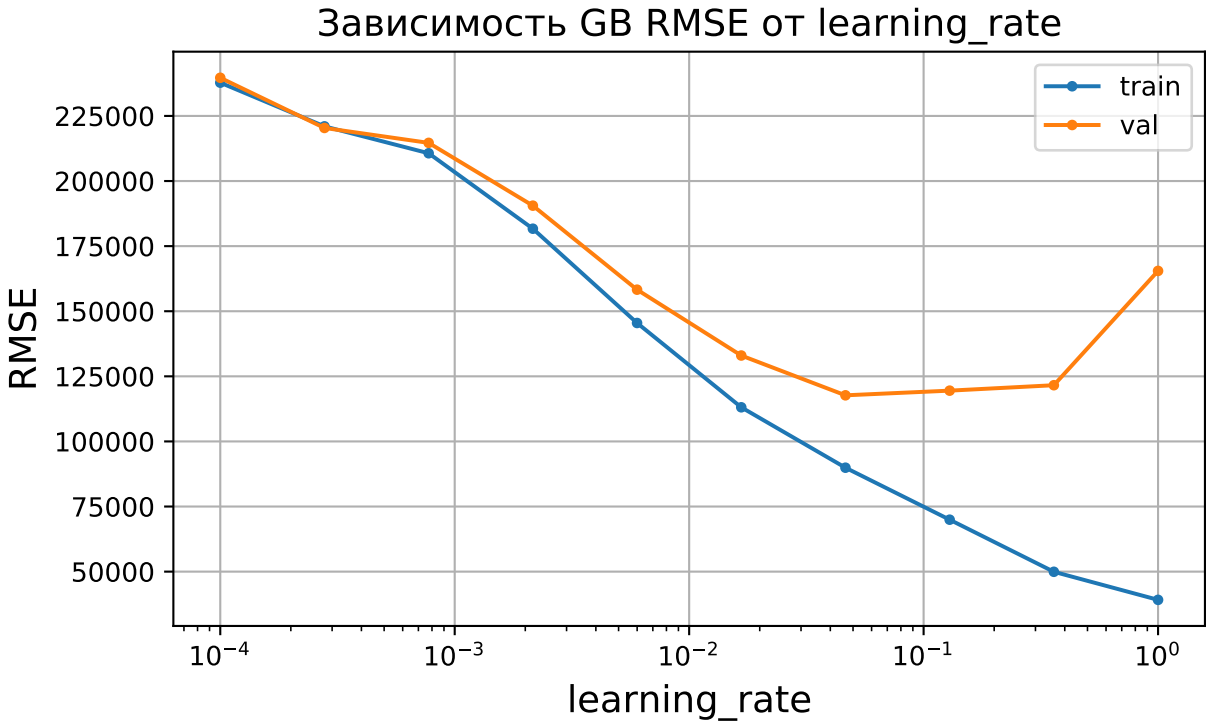
$n = 1000, md = 3, lr = 0.1$



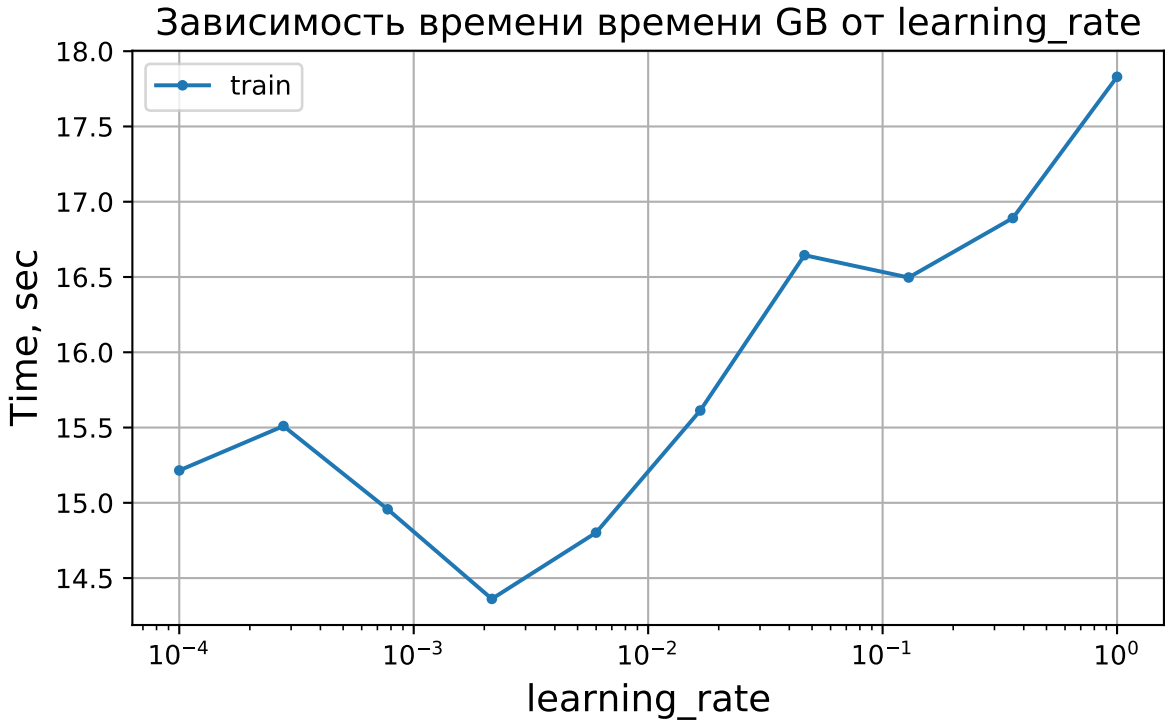
Здесь всё абсолютно аналогично случайному лесу. Однако, отметим, что график выходит гораздо быстрее на плато. Это можно объяснить алгоритмом градиентного бустинга, который исправля-

ет ошибки предыдущих деревьев. Всюду далее будем выбирать 7 признаков для обучения каждого дерева.

Наконец, подберём последний гиперпараметр - темп обучения. Посмотрим на качество работы модели.



$n = 1000, md = 3, fss = 7$



Заметим, что слишком маленький темп обучения приводит к очень большим значениям ошибки. Это можно объяснить тем, что каждое последующее дерево исправляет ошибки предыдущих. Тем самым при очень маленьком $learning\_rate$ этот "вклад"новых деревьев почти не будет учитываться. Т.е. ошибки, произошедшие на прошлой итерации, практически не будут исправляться. Однако при слишком большом значении этого параметра модель будет уделять большее внимание новым деревьям, которые заточены только на ошибки, но не на правильные ответы предыдущих. Следовательно, модель в меньшей степени будет учитывать "правильность"предыдущих методов. Таким образом, темп обучения должен быть не сильно маленьким, но и не большим. На графике видно, что оптимальное значение близко к дефолтному. Его мы и будем использовать.

Мы подобрали гиперпараметры по валидации. Теперь так же бучим модель на всей выборке и посмотрим на качество работы алгоритма на тесте. Возьмём $n\_estimators = 1000$ (количество деревьев в лесе), $max\_depth = 3$ (их глубина), $feature\_subsample\_size = 7$ (количество признаков подвыборки для одного дерева) и $learning\_rate = 0.1$ (темп обучения).

Ошибка на тесте составила примерно 118716, а время работы 21 секунду. Видим, что качество примерно такое же, как и на валидации, и существенно меньше чем у RandomForest. То есть, действительно, градиентный бустинг являетя более устойчивой и качественной моделью. Заметим, что если запустить модель с 1000 деревьями, оставив остальные параметры по умолчанию, то качество будет примерно 144862. Таким образом, и здесь подобрав гиперпараметры, можно довольно сильно снизить ошибку работы алгоритма.

Заключение

В данной работе были подробно рассмотрены и разобраны алгоритмы случайного леса и градиентного бустинга для задачи регрессии предсказания стоимости дома в Америке. Было проанализировано качество работы этих методов в зависимости от их гиперпарметров и сделаны соответствующие выводы. Так же был создан веб-сервис для работы с этими моделями с интерактивными графиками (библиотека *plotly*).