Отчет по заданию «Ансамбли алгоритмов. Веб-сервер. Композиции алгоритмов для решения задачи регрессии.» для курса «Практикум на ЭВМ 2021/2022» для бакалавров 3 курса кафедры ММП ВМК МГУ

Фролов А., 317 группа, ВМК МГУ

7 декабря 2021

1 Предобработка данных

Для проведения экспериментов выбрана задача предсказания стоимости квартир. Датасет состоит из 21613 записей. Для предобработки признаки были разделены на три группы:

- численные (и ординальные): bedrooms, bathrooms, sqft_living, sqft_lot, floors, condition, grade, sqft_above, sqft_basement, yr_built, yr_renovated, sqft_living15, sqft_lot15
- бинарные: waterfront
- категориальные: view, zipcode, zone

Был добавлен новый признак zone, показывающий находится ли дом в аномальном участке (цена отклонилась от средней больше чем на 3σ). Для этого местность была поделена на 400 зон. В столбце yr_renovated нули были заменены на значения из столбца yr_built.

Численные признаки были нормализованы, категориальные были преобразованы с помощью One-Hot кодирования, бинарные не изменялись. Колонки id, date, lat, long были удалены. После предобработки размер признакового пространства увеличился до 93 признаков.

2 Случайный лес

Исследуем поведение алгоритма Random Forest в зависимости от следующих параметров:

- количество деревьев в ансамбле
- размерность подвыборки признаков для одного дерева
- максимальная глубина дерева

Для визуализации зависимостей будем строить графики зависимости RMSE на отложенной выборке и времени работы алгоритма от количества деревьев в ансамбле для различных значений параметров.

2.1 Размерность подвыборки признаков

Посмотрим как ведет себя случайный лес при различных размерах подвыборки признаков. Глубину деревьев возьмем неограниченной.

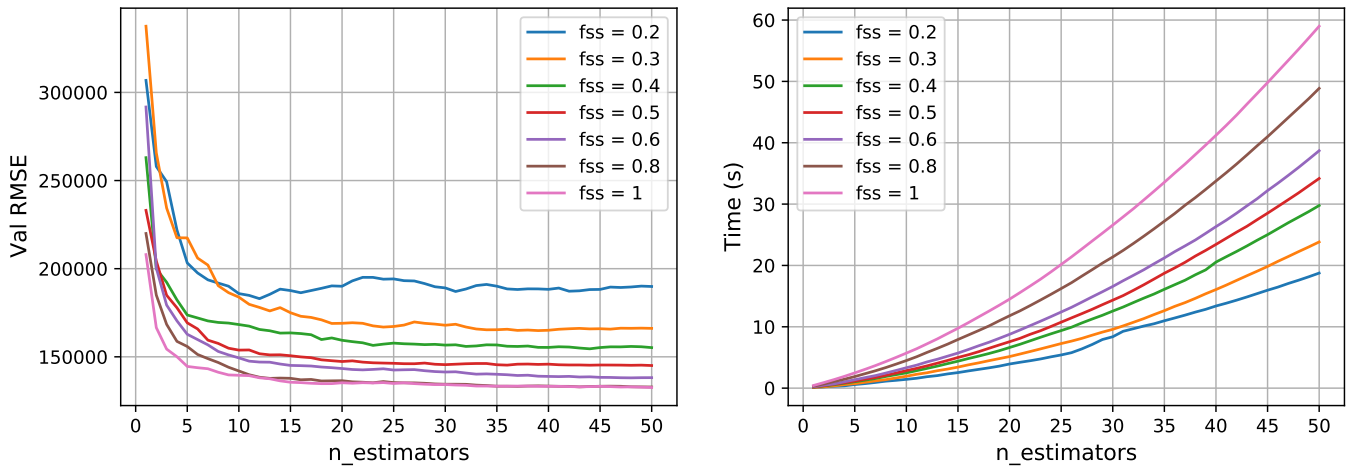

Рис. 1: Random Forest для различных разметростей подвыборки признаков

Можно заметить, что чем больше мы используем признаков, тем выше качество модели и больше время работы.

2.2 Максимальная глубина дерева

Далее попробуем различные значения максимальной глубины дерева. Размер подвыборки признаков возьмем равным 0.5.

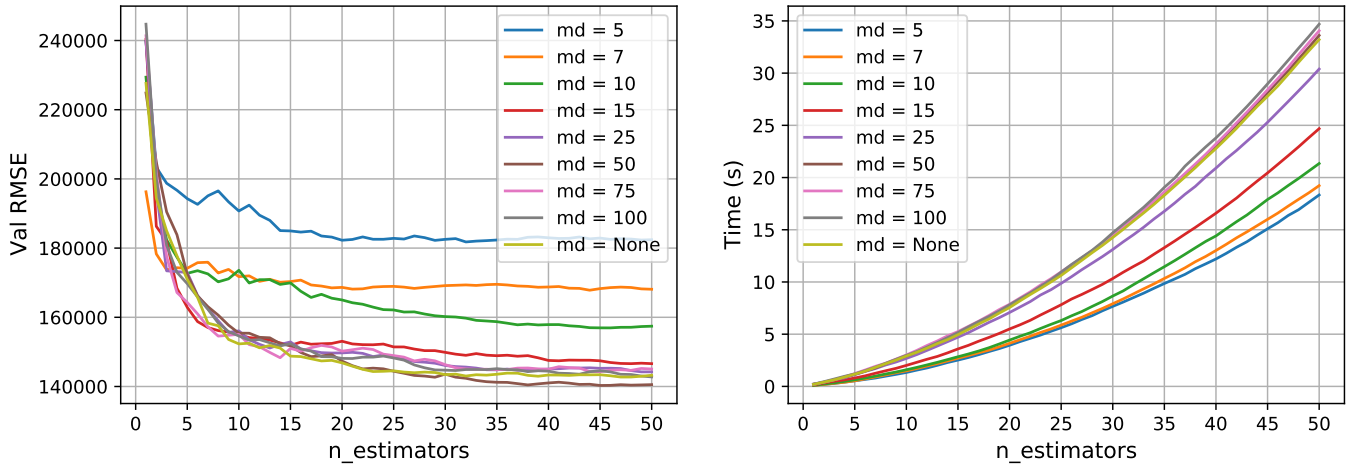

Рис. 2: Random Forest для различной максимальной глубины деревьев

Можно заметить, что при малой глубине алгоритм показывает плохое качество, однако при глубине больше 15 качество меняется незначительно. Время работы при глубине больше 25 также меняется незначительно, что может говорить о том, что модель редко строит деревья большой глубины, даже если ей это разрешить. Лучшим значением глубины оказалось 50 деревьев. При неограниченной глубине качество немного падает, вероятно, из-за переобучения.

2.3 Перебор параметров

Попробуем перебрать параметры близкие к оптимальным и посмотрим на результаты:

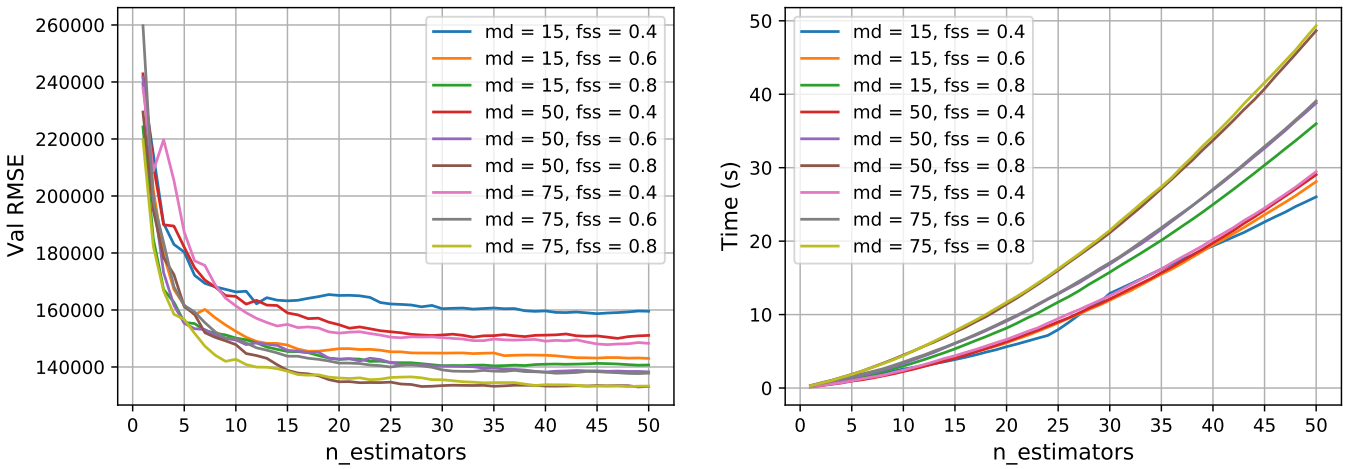

Рис. 3: Gradient Boosting для различных пар параметров

Перебор подтверждает результаты предыдущих эксперименотов, можно сказать что изменение размера подвыборки признаков влияет на качество значительно сильнее, чем глубина деревьев. Лучшей комбинацией параметров можно назвать максимальную глубину деревьев 75 и размер подвыборки признаков 0.8.

3 Градиентный бустинг

Далее исследуем поведение алгоритма Gradient Boosting в зависимости от следующих параметров:

- количество деревьев в ансамбле
- размерность подвыборки признаков для одного дерева
- максимальная глубина дерева
- темп обучения (`learning_rate`)

Для визуализации зависимостей будем строить графики зависимости RMSE на отложенной выборке и времени работы алгоритма от количества деревьев в ансамбле для различных значений параметров.

3.1 Размерность подвыборки признаков

Посмотрим как ведет себя градиентный бустинг при различных размерах подвыборки признаков. Максимальную глубину деревьев возьмем равной 5, `learning_rate` равным 0.1.

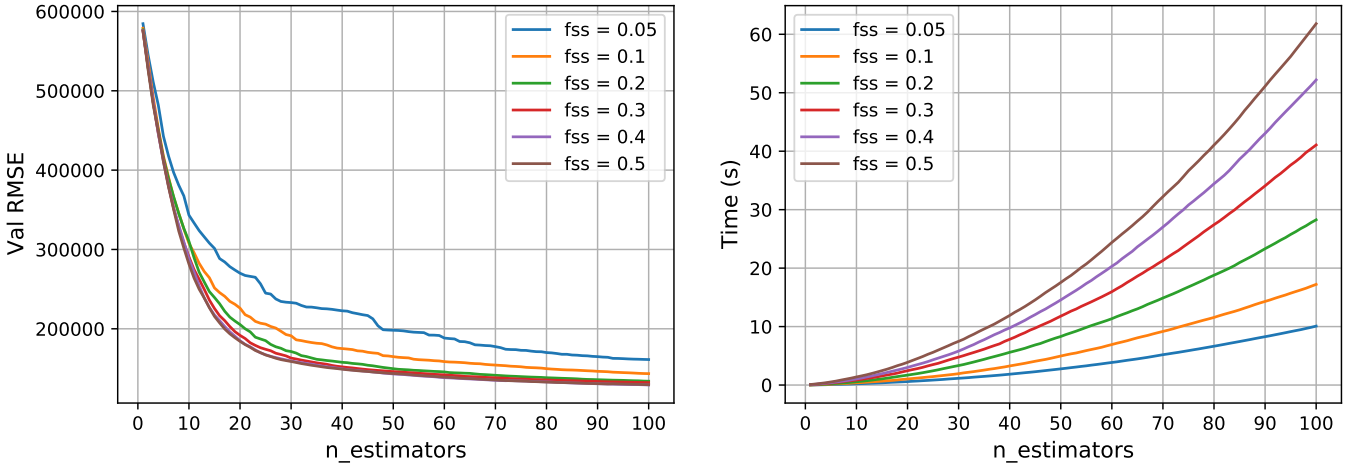

Рис. 4: Gradient Boosting для различных разметростей подвыборки признаков

Можно заметить, что при размере подвыборки больше 0.2, качество почти не меняется, а время работы увеличивается. Поэтому нет смысла брать размер подвыборки признаков больше 0.2-0.3.

3.2 Максимальная глубина дерева

Далее попробуем различные значения глубины деревьев. Размер подвыборки признаков возьмем равным 1/3, `learning_rate` равным 0.1.

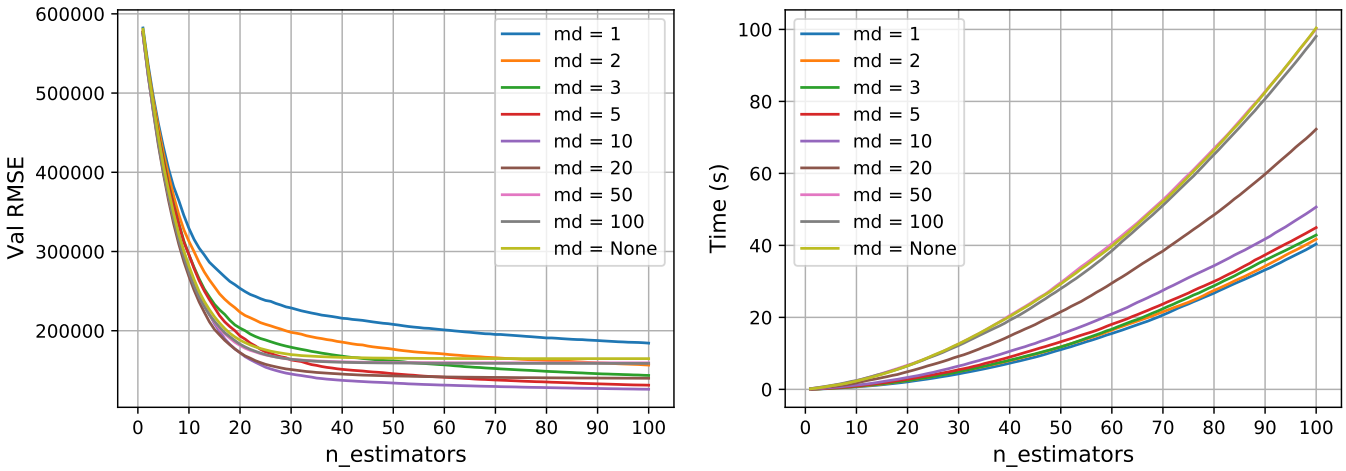

Рис. 5: Gradient Boosting для различной максимальной глубины деревьев

По графику можно сделать вывод, что при малой глубине деревьев алгоритм недообучается и ему требуется больше деревьев. При больших значениях глубины модель переобучается и не достигает лучшего качества. Оптимальной можно назвать глубину 5-10 деревьев. Время работы напрямую зависит от максимальной глубины деревьев, при значениях больше 100 отличия не так заметны, потому что модель не считает построение еще более глубоких деревьев полезным.

3.3 Темп обучения

У градиентного бустинга, в отличие от случайного леса, есть еще один параметр – темп обучения. Посмотрим, как ведет себя модель в зависимости от его значений. Размер подвыборки признаков возьмем равным 1/3, максимальную глубину деревьев равной 5.

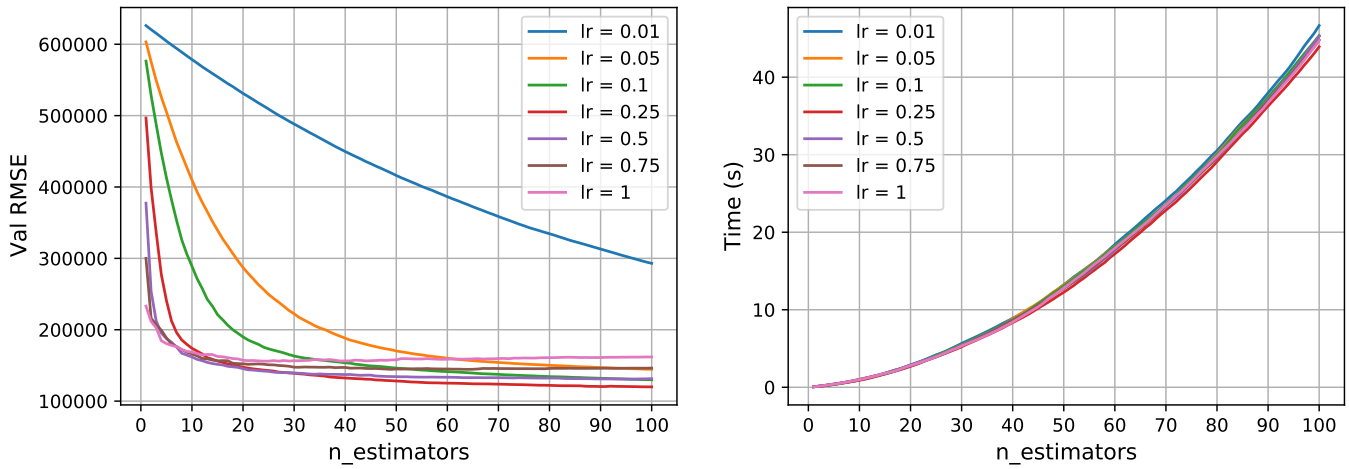

Рис. 6: Gradient Boosting для различного темпа обучения

Можно заметить, что при малом темпе обучения алгоритм сходится медленно, требуется намного больше деревьев. При слишком больших значениях же модель не может сойтись, каждое новое дерево дает слишком большой вклад, поэтому качество страдает. Оптимальным значением темпа обучения можно назвать 0.25. Время обучения одного дерева от значеня темпа обучения не зависит, но при небольших значениях алгоритму требуется больше итераций чтобы сойтись.

3.4 Перебор параметров

Попробуем перебрать параметры близкие к оптимальным при `learning_rate` равном 0.25, и посмотрим на результаты:

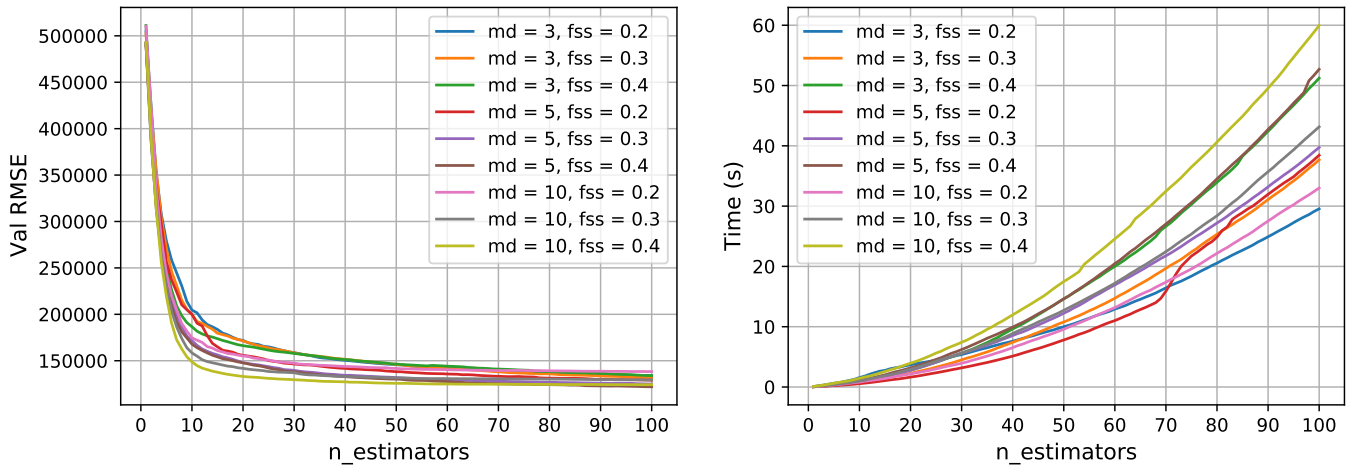

Рис. 7: Gradient Boosting для различных пар параметров

Перебор подтверждает результаты предыдущих экспериметов, лучшей комбинацией параметров можно назвать максимальную глубину деревьев 10 и размер подвыборки признаков 0.4.

4 Вывод

Градиетный бустинг работает лучше при небольшой глубине деревьев и не требует большого размера подвывборки признаков. Случайный лес же лучше работает при большей глубине

и размере подвыборки. Это связано с логикой алгоритмов. В градиентом бустинге каждый следующй алгоритм исправляет ошибки предыдущего, скажем, решает локальную задачу. В случайном лесу отдельное дерево сильнее влияет на ответ, поэтому важно, чтобы оно давало относительно неплохие предсказания для всех объектов.