Московский государственный университет имени М. В. Ломоносова.

Композиции алгоритмов для решения задачи регрессии.

Чванкина Дарья
Декабрь 2021
Отчет по курсу "Практикум на ЭВМ 2021/2022"

1 Введение

В данной работе исследуются композиции алгоритмов. В экспериментах используются собственные реализации случайного леса и градиентного бустинга. Были поставлены эксперименты для выявления того, как значения гиперпараметров влияют на алгоритмы. Во время экспериментов использовался набор данных о продажах недвижимости.

2 Эксперименты

2.1 Предобработка данных

Перед экспериментами была проведена предобработка данных, в ходе которой выяснилось, что пропущенных данных в наборе данных нет. Есть один категориальный признак - дата. Так как этот признак может быть существенным, то из него были извлечены год, день месяца, день недели и месяц. Новые признаки были помещены в набор данных, а признак "date"был удален. Далее набор данных был переведен в numpy.array.

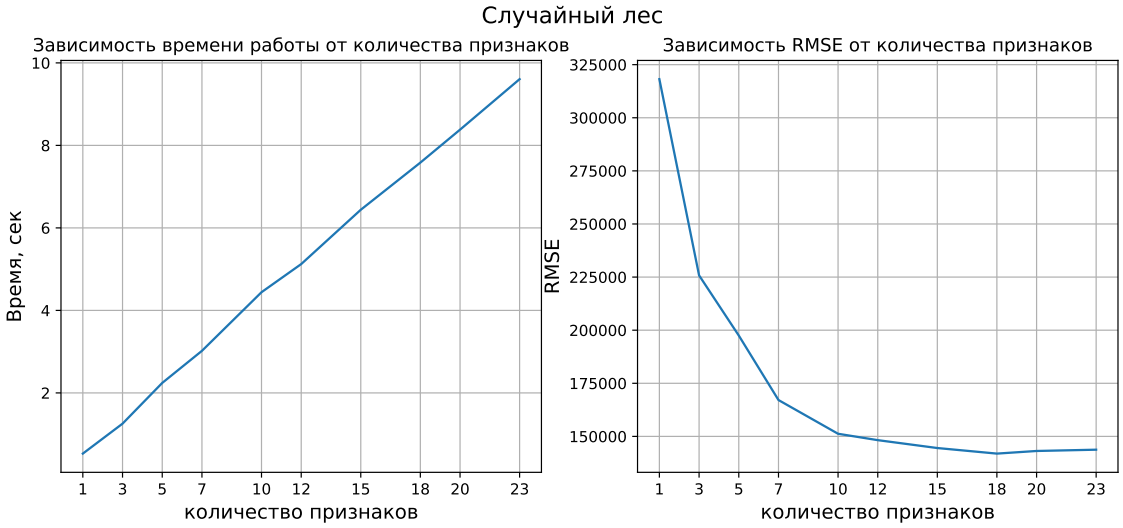
2.2 Случайный лес

Для алгоритма случайного леса параметр feature_subsample_size (размер признакого пространства) в дефолтных настройках был равен одной трети от признакого пространства набора данных.

Исследуем как влияет количество деревьев на алгоритм случайного леса, используя функцию потерь RMSE. Время работы алгоритма зависит линейно от количества деревьев. Функция потерь сначала резко падает, затем чуть-чуть растёт и выходит на асимптоту, то есть алгоритм во время роста функции потерь немного переобучается. Оптимальный параметр примерно равен 60.

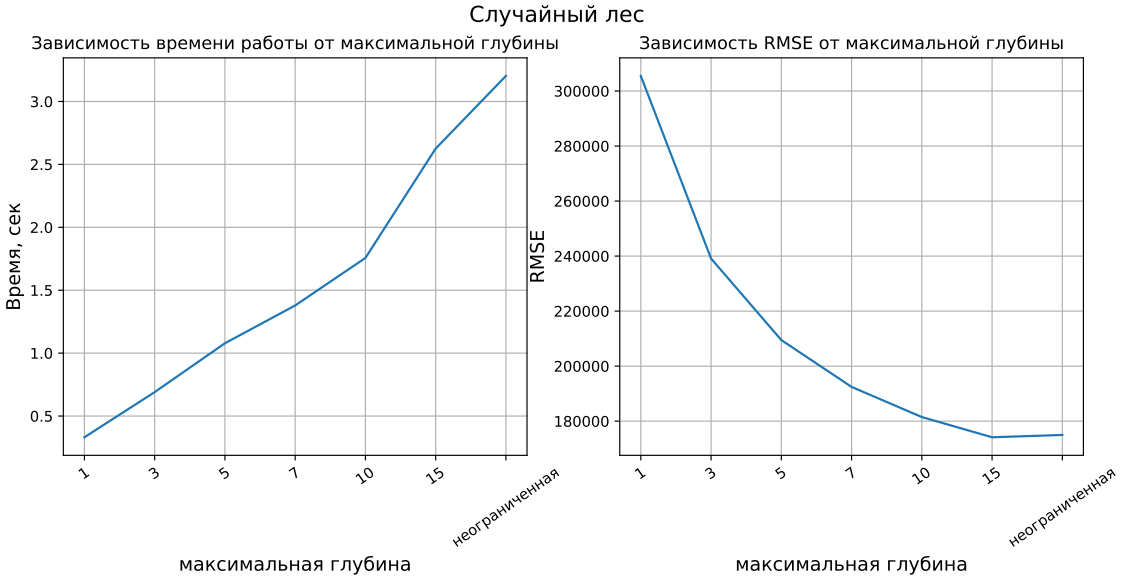
Пусть теперь количество деревьев равно 60, будем изменять размер признакого пространства. Время работы алгоритма от количества признаков зависит также линейно. Функция потерь уменьшается, но при использовании больше, чем 18 признаков функция потерь немного увеличивается. Это значит, что в данных есть признаки, которые не очень важны и при

удалении которых качество алгоритма может повыситься. В данном случае оптимальное количество признаков равно 18.



Рассмотрим влияние глубины одного дерева в ансамбле на модель. Для этого также зафиксируем количество деревьев 60, остальные параметры оставим дефолтными и будем изменять глубину.

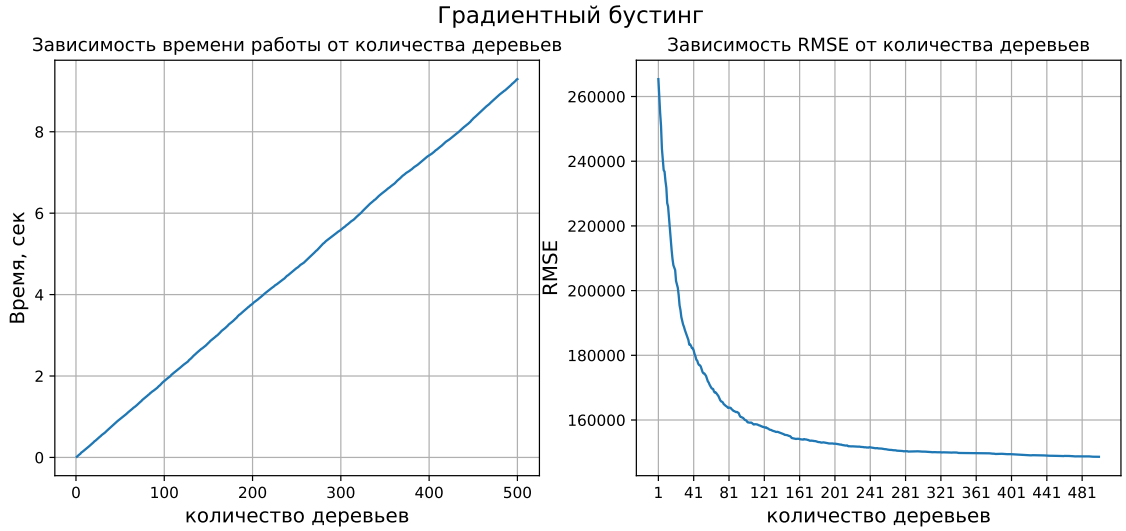
Время работы алгоритма от глубины также зависит линейно. Функция потерь уменьшается, но выходит на некоторую константу с ростом глубины. Это связано с тем, что в какой-то момент глубина дерева достигает своего максимума. В данном случае чем больше глубина, тем лучше результат показывает алгоритм.



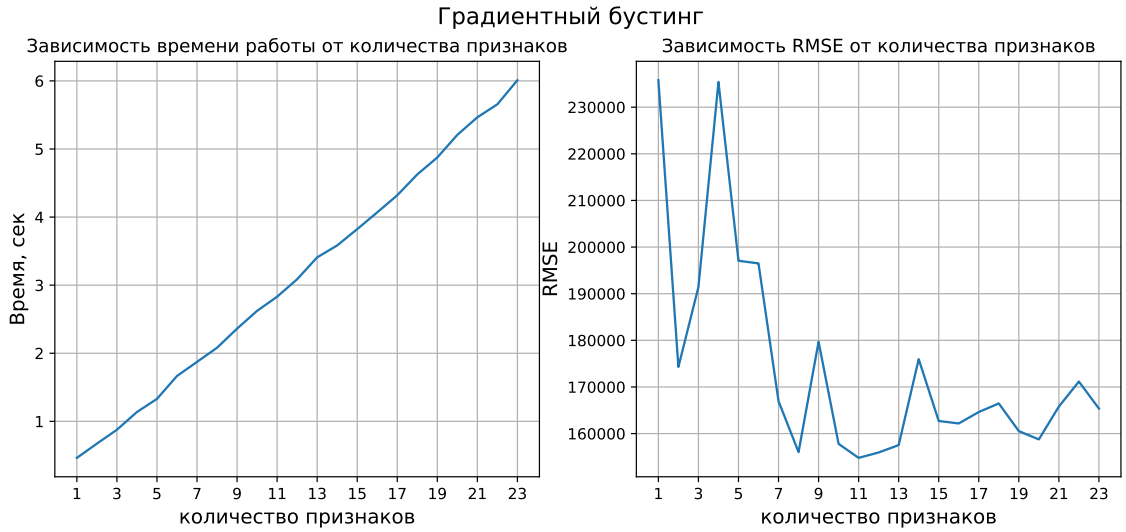
2.3 Градиентный бустинг

В данном эксперименте исследуются параметры градиентного бустинга. Во время эксперимента использовались дефолтные параметры(темп обучения = 0.1, максимальная глубина

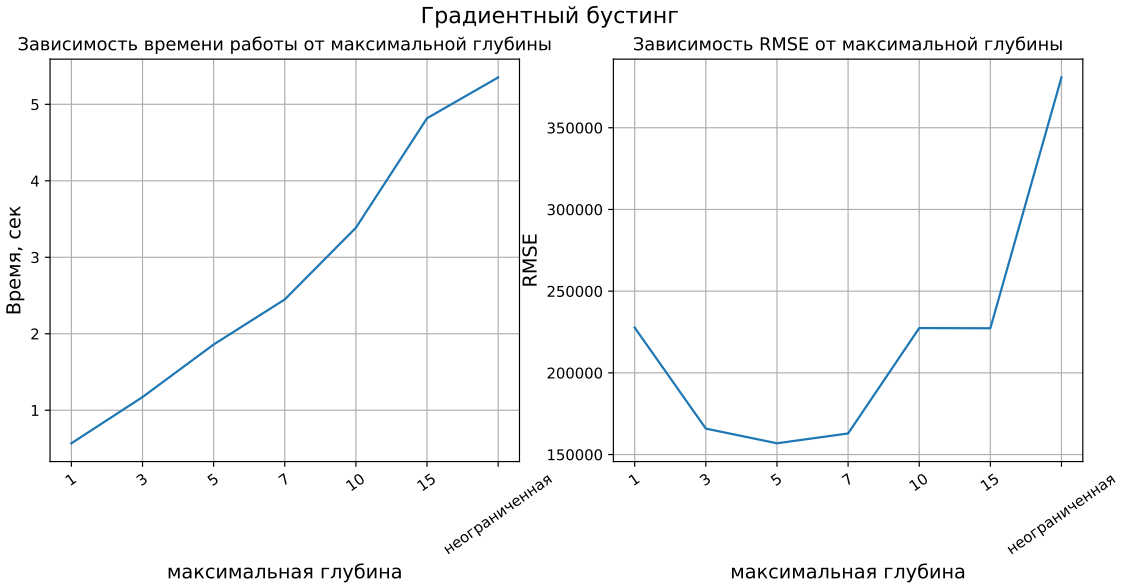
дерева = 5, размер признакого пространства = одна треть признаков набора данных). На графике зависимости функционала от количества деревьев нет колебаний графика, он более гладкий, чем у случайного леса. Это объясняется тем, что каждое следующее дерево исправляет ошибки предыдущих. Функционал падает к некоторой асимптоте. Оптимальное количество деревьев в данном случае примерно 280. Время работы алгоритма линейно зависит от количества деревьев.



Время работы алгоритма линейно зависит от количества сэмплируемых признаков. На графике можно увидеть большие колебания функционала, чего не было на графике случайного леса. Но можно заметить, что функционал при количестве признаков меньше, чем 7 больше, чем при количестве признаков больше 7. В данном случае оптимальное количество признаков равно 7, что меньше, чем для случайного леса. Это говорит о том, что для градиентного бустинга требуются более простые модели, чем для случайного леса.



Время работы алгоритма линейно зависит от глубины деревьев. Функционал с ростом глубины сначала понижается, а потом возрастает, то есть при большой глубине алгоритм переобучается. Оптимальная глубина деревьев равна 5. Данный алгоритм подтверждает то, что для градиентного бустинга требуются более простые модели, чем для случайного леса.



Исследуем как влияет темп обучения на алгоритм градиентного бустинга. Время алгоритма уже не имеет линейную зависимость. Но при темпе обучении меньше 0.07 время работы алгоритма стабильно возрастает, а функционал уменьшается. Причем при темпе обучении больше 0.07 график функционала начинает колебаться и значение функционала больше, чем при темпе меньше 0.07. Слишком маленькие значения темпа дают большой функционал. Оптимально выбирать темп роста из отрезка $[0.01, 0.1]$. Для этой задачи оптимальный темп обучения равен 0.07.