МГУ им.Ломоносова
Факультет ВМК кафедра ММП

Композиции алгоритмов для решения задач регрессии

Батшева Анастасия 317 группа

Москва
2020

Содержание

1 Формулировка задания

1. Написать на языке Python собственную реализацию методов случайных лес и градиентный бустинг.

2. Написать реализацию веб-сервера с требующейся функциональностью. Обернуть своё решение в докер контейнер.

3. Провести описанные ниже эксперименты с модельными данными и приложенным датасетом

2 Экспериментальная часть

Датасет для данного задания представляет собой данные о стоимости недвижимости на основании параметров жилья.

1. Провести минимальную обработку имеющихся данных. Разделить данные на обучение и контроль, перевести данные в `numpy ndarray`.

2. Исследовать поведение алгоритма случайный лес. Изучить зависимость RMSE и время работы алгоритма на отложенной выборке в зависимости от следующих факторов:

   (a) количество деревьев

   (b) размерность подвыборки признаков для одного дерева

   (c) максимальная глубина дерева (+случай, когда глубина неограничена)

3. Исследовать поведение алгоритма градиентный бустинг. Изучить зависимость RMSE и время работы алгоритма на отложенной выборке в зависимости от следующих факторов:

   (a) количество деревьев

   (b) размерность подвыборки признаков для одного дерева

   (c) максимальная глубина дерева (+случай, когда глубина неограничена)

   (d) выбранный `learning_rate` (каждый новый алгоритм добавляется в композицию с коэффициентом `\gamma \times learning_rate`)

3 Результаты экспериментов

Один из общих подходов в машинном обучении заключается в использовании композиции "слабых"алгоритмов. Итоговый алгоритм строится путем взвешенного голосования ансамбля базовых алгоритмов. Два подхода:

1. Bagging - обучение базовых алгоритмов на различных случайных подвыборках данных или/и на различных случайных частях признакового описания: при этом базовые модели строятся независимо друг от друга

2. Boosting - каждый следующий базовый алгоритм строится с использованием информации об ошибках предыдущего правил: а именно, веса объектов обучающей выборки подстраиваются таким образом, чтобы новый алгоритм точнее работал на тех объектах, на которых предыдущий чаще ошибался.

В теории бустинг работает на больших обучающих выборках, а баггинг – на малых. Проверим это:

(Зависимость снимается по RMSE и времени работы)

3.1 Bagging: Случайный лес

Random forest — это множество решающих деревьев, где в задаче регрессии их ответы усредняются. Все деревья в моей реализации строятся независимо и параллельно по следующей схеме:

1. Выбирается подвыборка (размера `samplesize`): по умолчанию это $\mathrm{n}_{\overline{3(^-)}}$

2. Каждое расщепление в дереве определяется по `max_features` случайных признаков

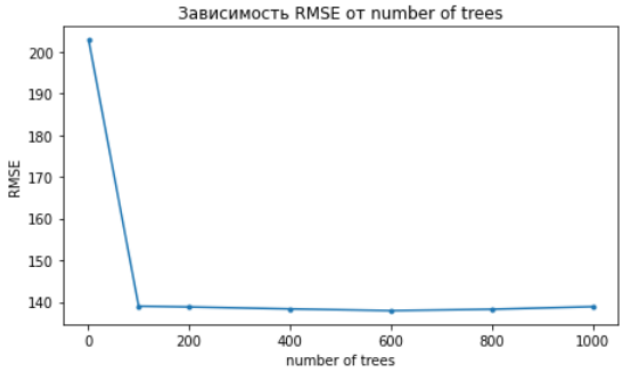
3. Выбираем наилучшие признак и расщепление по нему с ограничением по высоте дерева (или при отсутствии ограничения до исчерпания признаков)

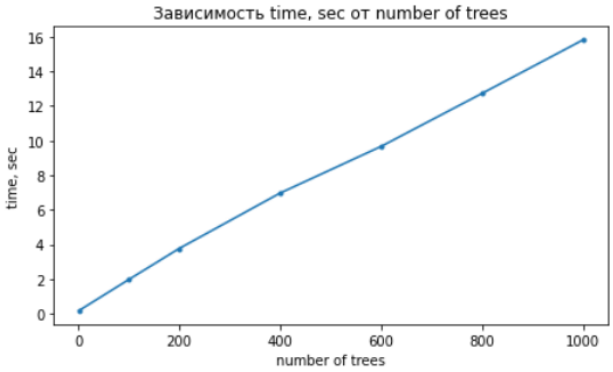
Очевидно, что при таком построении, модель соответствует главному принципу ансамблирования: базовые алгоритмы должны быть хорошими и разнообразными (каждое дерево строится на своей обучающей выборке и при выборе расщеплений есть элемент случайности).

Одно из достоинств случайного леса - отсутствие переобучения и уменьшение разброса при увеличении числа деревьев.

3.1.1 Зависимость от числа деревьев

```
CPU times: user 6.88 s, sys: 569 ms, total: 7.45 s
Wall time: 51.2 s
```



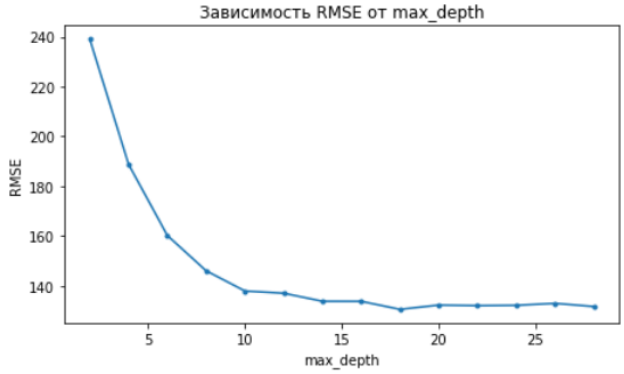


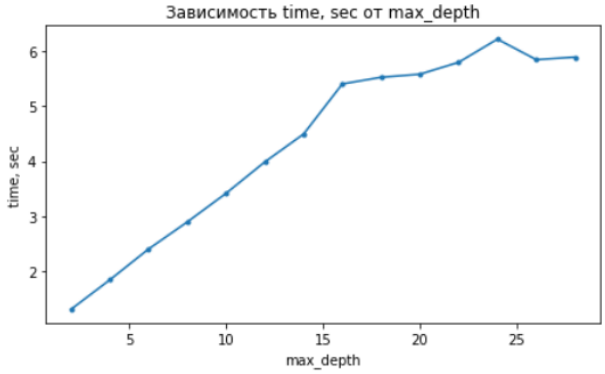
Как видно, 200 деревьев с глубиной 10 оказалось оптимальным числом, ибо впоследствии уменьшение ошибки несоизмеримо меньше прироста времени - качество вышло на асимптоту, что подтверждает теорию об уменьшении разброса модели при росте числа деревьев.

3.1.2 Зависимость от глубины деревьев

Безусловно, увеличение глубины дерева приводит к бОльшей гибкости модели (бОльшей чувствуительности к разного рода зависимостям в данных, даже трудноопределимым). Однако цена за гибкость - переобучение и время работы. График:

```
CPU times: user 10.9 s, sys: 1.69 s, total: 12.6 s
Wall time: 1min
```





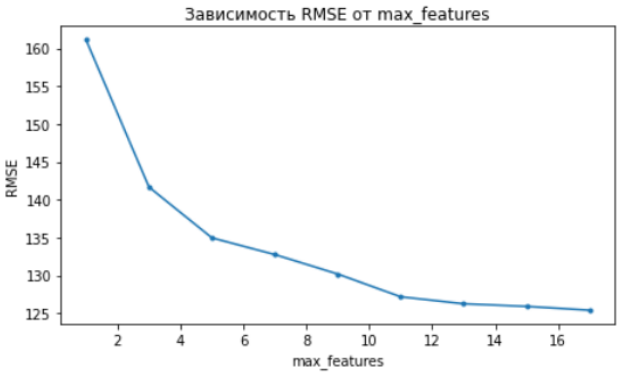
Даже с неограниченной глубиной мы не достигаем абсолютной точности и получаем значения, примерно такие же, как и в крайнем случае (30).

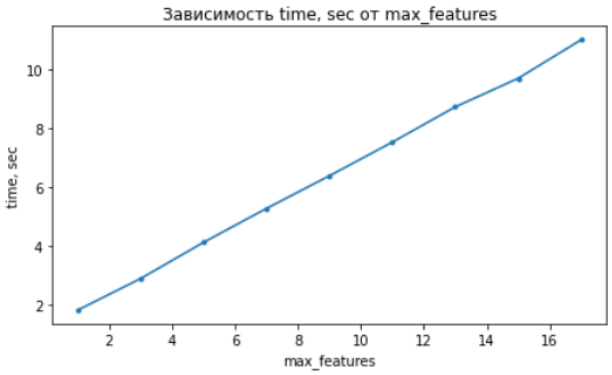
```
RMSE: 133.024  time: 6.265 sec
```

3.1.3 Зависимость от размерности подмножества признаков

Очевидно предположить, что при увеличении размерности признакового (описательного) пространства, качество будет расти за счет выделения закономерностей.

```
CPU times: user 7.61 s, sys: 987 ms, total: 8.6 s
Wall time: 57.6 s
```





Однако зависимость, вопреки ожиданиям не линейная, а почти обратно пропорциональная. Условным плато можно назвать кривую после 11 признаков, что говорит нам о то, что именно такой набор является достаточным для такой же точности, что и на всем пр-ве. К тому же, при большом числе признаков деревья однообразны.

3.2 Boosting: Градиентный бустинг

Градиентный бустинг деревьев решений – другой универсальный алгоритм машинного обучения, основанный на использовании ансамбля деревьев решений. В отличие от случайного леса градиентный бустинг является развитием бустинг-идеи.

Алгоритм минимизирует эмпирический риск жадным пошаговым алгоритмом, аналогичным методу градиентного спуска:

1. на предварительном этапе алгоритм строит оптимальную константную модель `f = g_0`

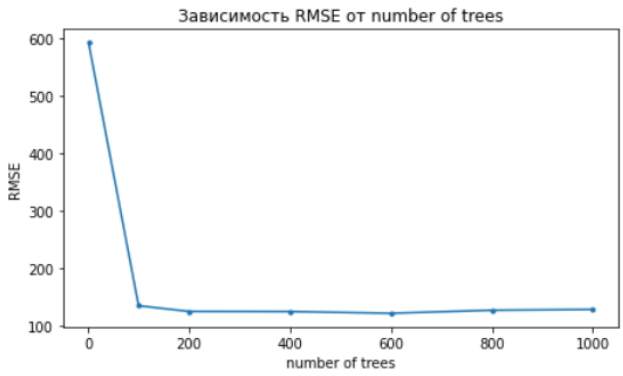
2. на m-й итерации конструируется дерево решений `g_m`, аппроксимирующее компоненты вектора антиградиента, вычисленного для текущей модели `f`

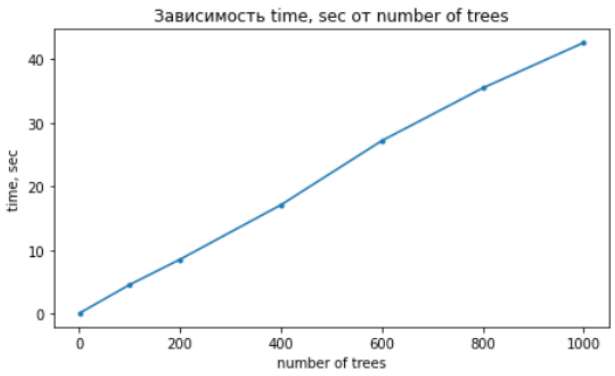
3. значения в узлах построенного дерева `g_m` перевычисляются, так, чтобы минимизировать функционал ошибки (в нашем случае RMSE)

Как и в случае случайного леса мы ставим изначальный параметр `samplesize = \f` размера подвыборки признаков. И в данной реализации деревья, что очевидно, растут последовательно, обучаясь на ошибках предыдущих, что, конечно, приводит к большим затратам времени.

3.2.1 Зависимость от числа деревьев

```
CPU times: user 2min 15s, sys: 6.58 ms, total: 2min 15s
Wall time: 2min 15s
```

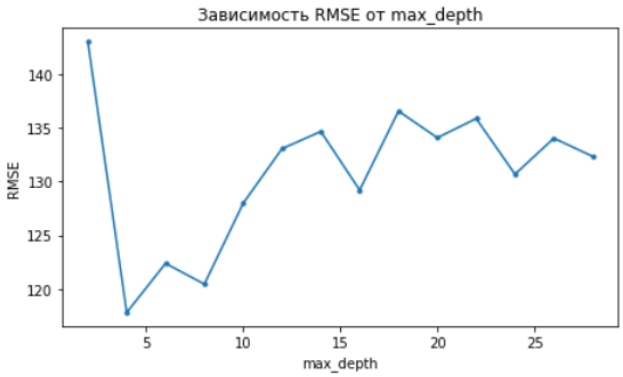


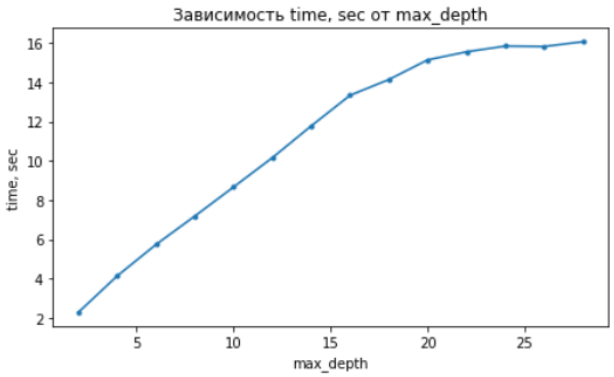


Как видно, зависимость качества градиентного бустинга практичсеки повторяет зависмость для случайного леса. Оценка в 200 деревьев так же оптимальна, после чего RMSE выходит на асимптоту.

3.2.2 Зависимость от глубины деревьев

```
CPU times: user 2min 35s, sys: 563 ms, total: 2min 35s
Wall time: 2min 36s
```





Странный скачущий вид функции объясняется, вероятно, случайным совпадением деревьев по группам единообразия признаков (возможно).

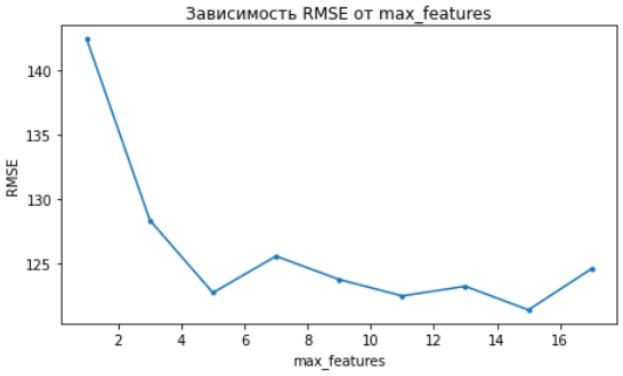
```
RMSE: 131.135  time: 16.351 sec
```

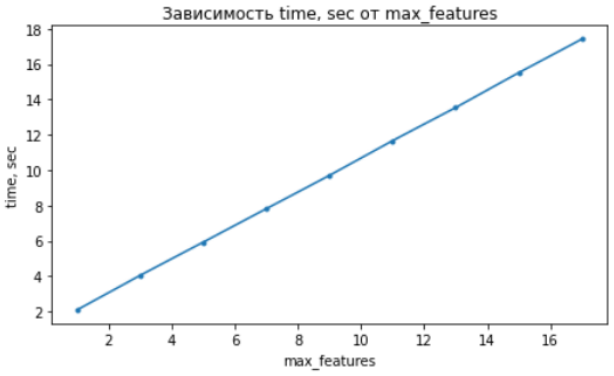
В принципе, при неограниченной глубине качество лучше, чем у леса.

3.2.3 Зависимость от размерности подмножества признаков

Аналогично случайному лесу ожидание - линейная зависимость, реальность: обратная пропорциональность с оптимальным значением в 11 признаков.

```
CPU times: user 1min 27s, sys: 30.7 ms, total: 1min 27s
Wall time: 1min 27s
```

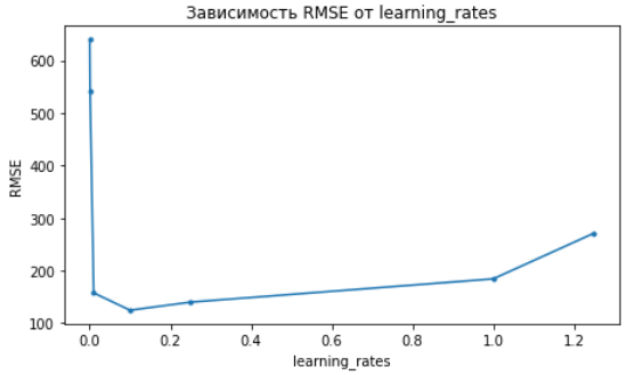


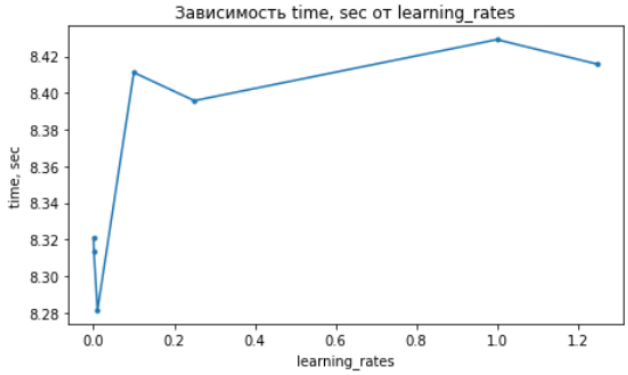


3.2.4 Зависимость от learning-rate

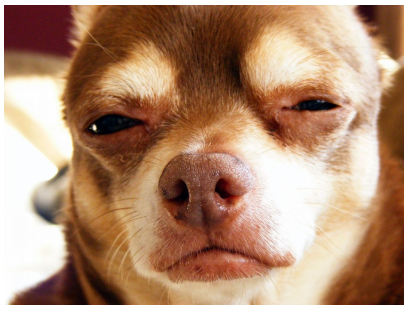
По умолчанию "недоверие"к ответу каждого базового алгоритма learning rate = 0.1 (на этот коэффициент мы умножаем результат). Посмотрим, что происходит при изменении этого параметра:

```
CPU times: user 58.6 s, sys: 1.94 ms, total: 58.6 s
Wall time: 58.6 s
```





Узрим расхождение при learning rate > 1.0. Довольно логично, недоверчивая модель не дает хороший результат:



3.3 Выводы

Были применены методы bagging и boosting в виде случайного леса и градиентного бустинга к данным, построены зависимости качества от разных параметров. Ансамбли лес и бустинг показали себя приблизительно одинаково на входных данных.

Веб-сервер, реализованный под данные методы расположен в репозитории github. Спасибо за внимание, с наступающим Новым Годом!