Ансамбли алгоритмов
Random Forest и Gradient Boosting

Штыков Павел

Московский государственный университет им. М. В. Ломоносова

11 декабря 2021

1 Постановка задачи

В качестве задания предлагается реализовать на языке Python алгоритмы Random Forest и Gradient Boosting для задачи регрессии. Провести эксперименты на предоставленном датасете. Изучить поведение разных методов ансамблирования в зависимости от гиперпараметров моделей.

2 Теоретическая справка

В данной работе мы изучаем следующие два метода ансамблирования:

Random Forest Данный метод является простым голосованием над предсказаниями решающих деревьев, которые обучались на бутстрапированных подвыборках исходного датасета (ссылка).

Gradient Boosting В отличии от Random Forest Gradient Boosting строит композицию последовательно. Так каждое последующее дерево пытается исправить суммарную ошибку предыдущих, приближая антиградиент функции потерь по сумме предыдущих предсказаний: $-\mathcal{L}'_f(f_{prev}, y)$ (ссылка).

При этом в обоих алгоритмах каждому дереву для обучения достается собственное случайно выбранное подмножество признаков размером `feature_subsample_size` от общего размера признакового пространства.

3 Практическая часть

При проведении экспериментов мы будем решать задачу по прогнозированию цены домов в США (датасет). Целевая метрика качества — RMSE.

В качестве базовой предобработки данных колонка `date` с датой продажи была разбита на две колонки: месяц и год продажи и пропуски в данных были заменены на моду по столбцам.

В таблице 1 указаны гиперпараметры, которые использовались в экспериментах по умолчанию.

Все эксперименты производились с разделением выборки на обучающую и валидационную части (в соотношении 3 к 1). На всех графиках представлены обучающие кривые именно для валидационной подвыборки.

Random Forest	
Количество деревьев (`n_estimators`)	100
Максимальная глубина деревьев (`max_depth`)	Неограниченно
Размер подвыборки признаков (`feature_subsample_size`)	0.6
Gradient Boosting	
Количество деревьев (`n_estimators`)	100
Максимальная глубина деревьев (`max_depth`)	5
Размер подвыборки признаков (`feature_subsample_size`)	0.8
Коэффициент скорости обучения (`learning_rate`)	0.1

Таблица 1: Гиперпараметры по умолчанию

3.1 Random Forest

3.1.1 Исследование количества деревьев

На рисунке 1 представлен график зависимости функции потерь от количества деревьев в ансамбле и времени работы. `n_estimators` перебирался в диапазоне от 1 до 10000.

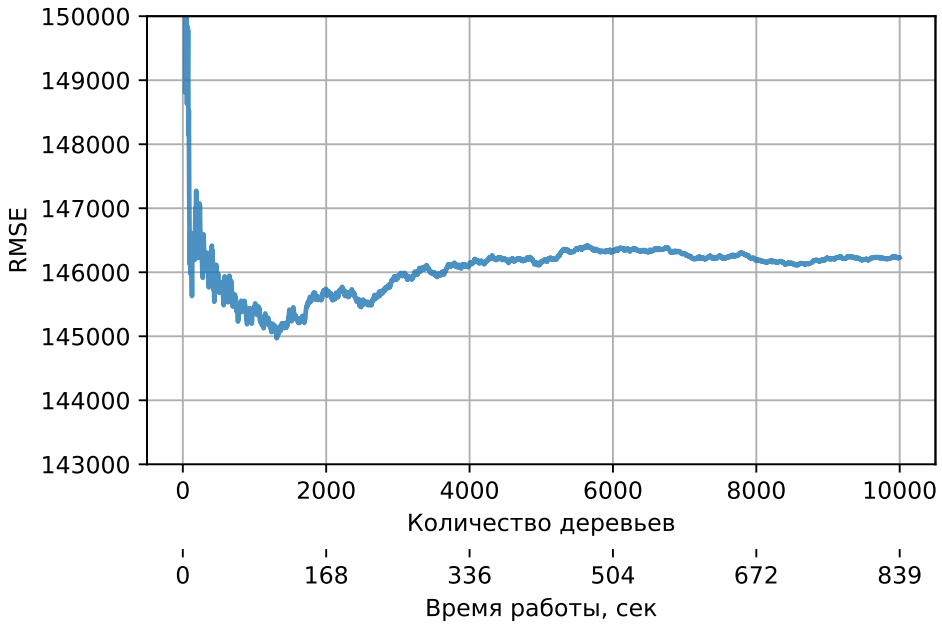

Рис. 1: Зависимость функции потерь от количества деревьев и времени работы для Random Forest

Из графика видно, что наилучшей точности модель достигает при примерно 1200 деревьях в ансамбле. После этого модель начинает переобучаться и выходит на плато примерно при 4000 деревьев. Также заметно, что при относительно небольшом количестве деревьев (меньше 500) кривая неустойчива.

В последующих экспериментах для Random Forest будем считать `n_estimators` = 1200 — значением по умолчанию.

3.1.2 Исследование максимальной глубины деревьев

На рисунке 2 представлен график зависимости функции потерь от времени работы при разных значениях максимальной глубины деревьев в ансамбле. `max_depth` перебирался по сетке: $\{10, 20, 30, 40, None$ (неограниченная глубина)$\}$.

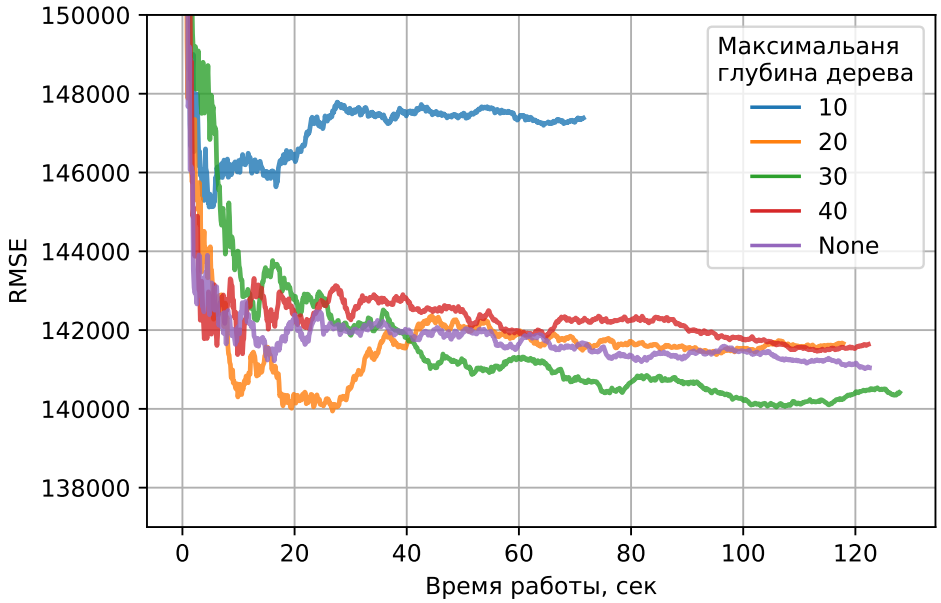

Рис. 2: Кривые обучения при разных значениях максимальной глубины деревьев для Random Forest

Из графика видно, что модель достигает лучших значений при `max_depth` = 20 и `max_depth` = 30, но делает это при разном количестве количестве деревьев (напомним, что `n_estimators` = 1200). Однако, хороший результат для `max_depth` = 20 при малом количестве деревьев может быть случайностью, поэтому выберем значение `max_depth` = 30, как более устойчивое. В целом же "обрубание"деревьев позволяет им меньше переобучаться, поэтому модель с неограниченными деревьями немного им проигрывает.

В последующих экспериментах для Random Forest будем считать `max_depth` = 30 — значением по умолчанию.

3.1.3 Исследование размера подвыборки признаков

На рисунке 3 представлен график зависимости функции потерь от времени работы при разных значениях размера подвыборки признаков для обучения деревьев в ансамбле. `feature_subsample_size` перебирался по сетке: $\{0.2, 0.4, 0.6, 0.8, 1\}$.

Из графика видно, что данный параметр сильно влияет на точность модели. Лучшего же результата алгоритм достигает при `feature_subsample_size` = 0.8, при этом результат устойчив на достаточно большой части обучающей кривой.

В последующих экспериментах для Random Forest будем считать `feature_subsample_size` = 0.8 — значением по умолчанию.

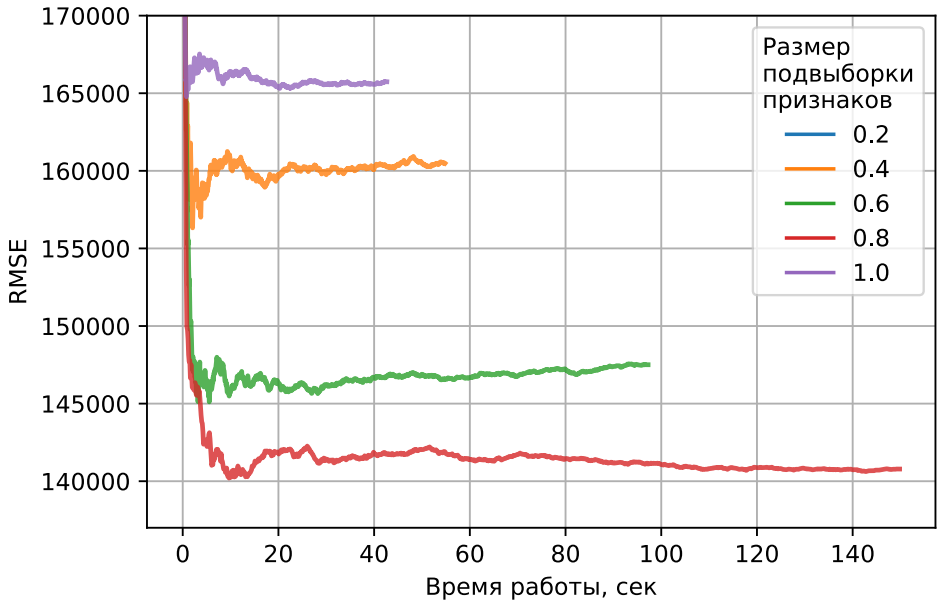

Рис. 3: Кривые обучения при разных значениях размера подвыборки признаков для Random Forest

3.2 Gradient Boosting

3.2.1 Исследование количества деревьев

На рисунке 4 представлен график зависимости функции потерь от количества деревьев в ансамбле и времени работы. `n_estimators` перебирался в диапазоне от 1 до 10000.

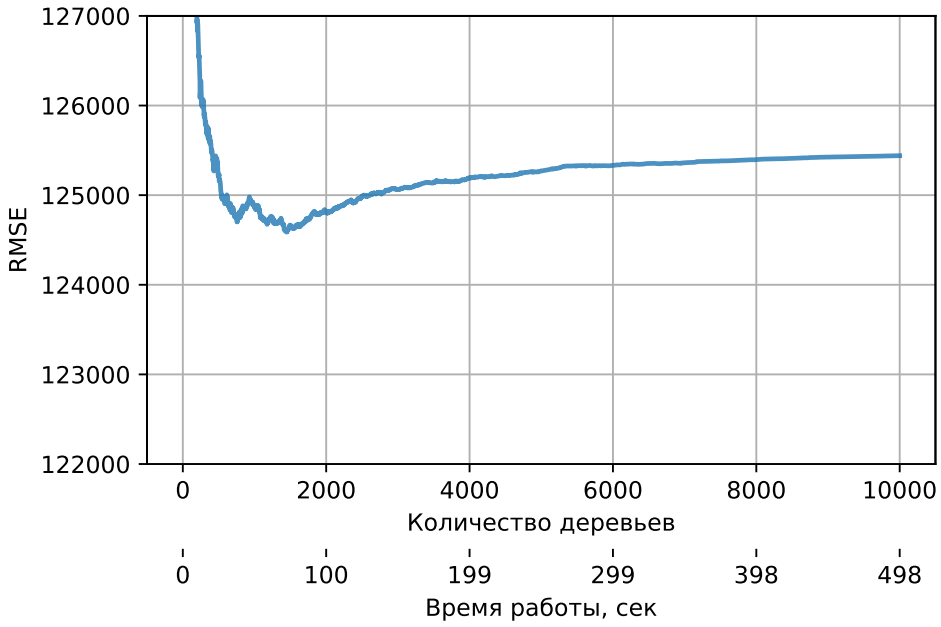

Рис. 4: Зависимость функции потерь от количества деревьев и времени работы для Gradient Boosting

Из графика видно, что наилучшей точности модель достигает при примерно 1000 деревьях в ансамбле. После этого качество немного падает и крива выходит на плато при-

мерно при 6000 деревьев. Стоит также заметить, что график заметно менее шумный чем у Random Forest. Также Gradient Boosting работает примерно в два раза быстрее.

В последующих экспериментах для Gradient Boosting будем считать `n_estimators` = 1000 — значением по умолчанию.

3.2.2 Исследование максимальной глубины деревьев

На рисунке 5 представлен график зависимости функции потерь от времени работы при разных значениях максимальной глубины деревьев в ансамбле. `max_depth` перебирался по сетке: $\{3, 4, 5, 6, 7, None$ (неограниченная глубина)$\}$.

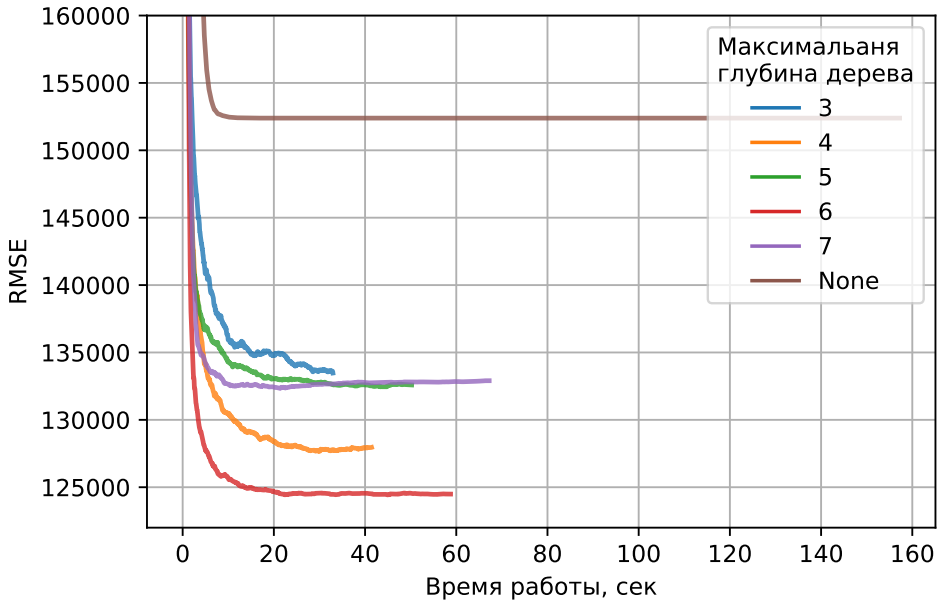

Рис. 5: Кривые обучения при разных значениях максимальной глубины деревьев для Gradient Boosting

Из графика видно, что модель достигает лучших значений при `max_depth` = 6. При этом сильно выделяется случай с неограниченной глубиной дерева — модель работает в 2.5 раза дольше и достигает гораздо более плохих результатов. Так происходит из-за того, что глубокие деревья достаточно быстро устраняют все ошибки предыдущих моделей в ансамбле.

В последующих экспериментах для Gradient Boosting будем считать `max_depth` = 6 — значением по умолчанию.

3.2.3 Исследование размера подвыборки признаков

На рисунке 6 представлен график зависимости функции потерь от времени работы при разных значениях размера подвыборки признаков для обучения деревьев в ансамбле. `feature_subsample_size` перебирался по сетке: $\{0.2, 0.4, 0.6, 0.8, 1\}$.

На графике заметно сильное отличие `feature_subsample_size` = 0.2 и 1 от остальных значений — они достаточно сильно проигрывают в качестве. Также видно, что размер подвыборки значительно влияет на скорость обучения. В целом лучшего качества алгоритм достигает при `feature_subsample_size` = 0.8.

В последующих экспериментах для Gradient Boosting будем считать `feature_subsample_size` = 0.8 — значением по умолчанию.

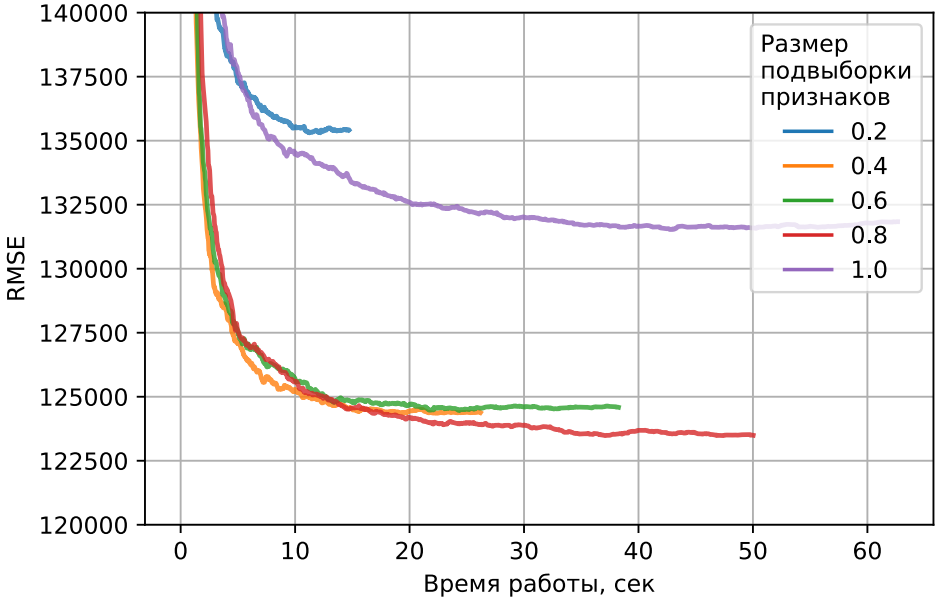

Рис. 6: Кривые обучения при разных значениях размера подвыборки признаков для Gradient Boosting

3.2.4 Исследование коэффициента скорости обучения

На рисунке 7 представлен график зависимости функции потерь от времени работы при разных значениях коэффициента скорости обучения деревьев в ансамбле. `learning_rate` перебирался по сетке: $\{0.1, 0.15, 0.2, 0.3, 0.5\}$.

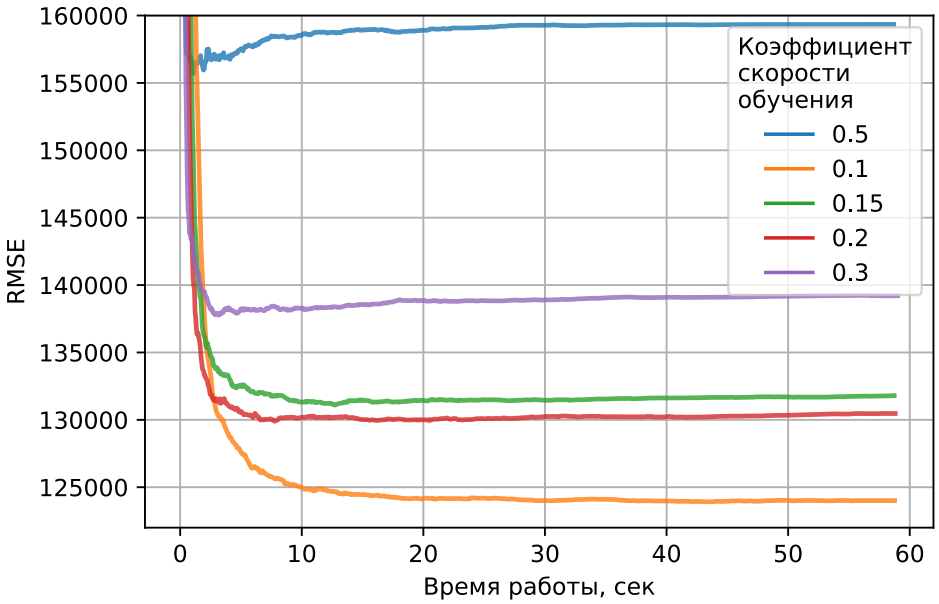

Рис. 7: Кривые обучения при разных значениях коэффициента скорости обучения деревьев для Gradient Booting

Из графика видно, что значение по умолчанию `learning_rate` $= 0.1$ достигает заметно лучшего результата.

В последующих экспериментах для Gradient Boosting будем считать `learning_rate` = 0.1 — значением по умолчанию.

3.3 Результаты экспериментов

В таблице 2 указаны значения лучших гиперпараметров для обоих алгоритмов ансамблирования, подобранные в ходе экспериментов:

Random Forest	
Количество деревьев (`n_estimators`)	1200
Максимальная глубина деревьев (`max_depth`)	30
Размер подвыборки признаков (`feature_subsample_size`)	0.8
Gradient Boosting	
Количество деревьев (`n_estimators`)	1000
Максимальная глубина деревьев (`max_depth`)	6
Размер подвыборки признаков (`feature_subsample_size`)	0.8
Коэффициент скорости обучения (`learning_rate`)	0.1

Таблица 2: Лучшие гиперпараметры

Прогнав модели с лучшими гиперпараметрами, мы получили следующие финальные результаты (таблица 3):

Модель	Время обучения	Результат на обучении	Результат на валидации
Random Forest	124 сек	140420	45603
Gradient Boosting	57 сек	128962	24767

Таблица 3: Результаты экспериментов

4 Заключение

В данной работе нами были реализованы два метода ансамблирования: Random Forest и Gradient Boosting. Мы изучили их поведение в задачи регрессии на реальных данных. Опираясь на кривые обучения мы подобрали оптимальные гиперпараметры и рассмотрели разные особенности алгоритмов.