Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Отчёт по практикуму

Сборка при помощи Dockerfile интерактивного веб-приложения, реализующего модели случайного леса и градиентного бустинга

Илюхин Владислав
317 группа

Москва 2021

Содержание

1 Постановка задачи **1**
1.1 Случайный лес и градиентный бустинг . . . 1
1.2 Веб-приложение . . . 1
1.3 *GitHub* . . . 1

2 Датасет **1**
2.1 Описание датасета . . . 1
2.2 Предобработка данных . . . 2

3 Случайный лес **2**
3.1 Описание алгоритма . . . 2
3.2 Реализация . . . 2
3.3 Эксперименты . . . 2

4 Градиентный бустинг **6**
4.1 Описание алгоритма . . . 6
4.2 Реализация . . . 6

1 Постановка задачи

1.1 Случайный лес и градиентный бустинг

Случайный лес и градиентный бустинг - ансамблевые методы машинного обучения. Первая часть задания состоит в их реализации и подборе оптимальных параметров на датасете $HouseSalesinKingCounty, USA$.

1.2 Веб-приложение

Для того, чтобы незнакомые с *Python* пользователи могли использовать обученные модели, необходимо реализовать веб-приложение с простым и удобным интерфейсом на основе $Dockerfile$ и $Flask$, позволяющее клиентам взаимодействовать с построенными ансамблями.

1.3 *GitHub*

Весь проект необходимо вести на платформе *Github* в соответствии с приведенными в задании правилами. В конце необходимо сдать заполненный данными приватный репозиторий вместе с его содержимым.

2 Датасет

2.1 Описание датасета

Датасет описывает продажи домов в округе Кинг штата Вашингтон США в период с мая 2014 года по май 2015 года. Признаки - цена, время продажи, а также различные характеристики дома.

2.2 Предобработка данных

Значения столбца $Date$ (дата продажи) преобразовал в признаки Day (день продажи), $Week$ (неделя продажи), $Month$ (месяц продажи). Во втором домашнем задании по ММРО именно такое преобразование даты дало хорошие результаты на обучении модели.

3 Случайный лес

3.1 Описание алгоритма

Случайный лес - ансамблевый метод машинного обучения, представляющий из себя бэггинг над методом случайных подпространств для решающих деревьев.

Для данных выборки X с ответами y строим $n\_estimators$ решающих деревьев с максимальной глубиной max_depth. Далее для каждого из решающих деревьев:

- Из множества объектов X выбираем случайное подмножество $\overline{X}$ размером 60% от исходного (на лекциях $60 - 80\%$ указывались как оптимальный параметр).
- После этого в подвыборке оставляем значения только $k\_frac\%$ от всех признаков (с округлением числа оставленных признаков вверх), получаем подвыборку $\overline{\overline{X}}$.
- На полученной подвыборке $\overline{\overline{X}}$ обучаем решающее дерево

3.2 Реализация

Алгоритм реализован на языке $Python$ с использованием класса $DecisionTreeRegressor$ из библиотеки $sklearn$. Содержится в файле $ensembles.py$ в директории src $Github$-репозитория.

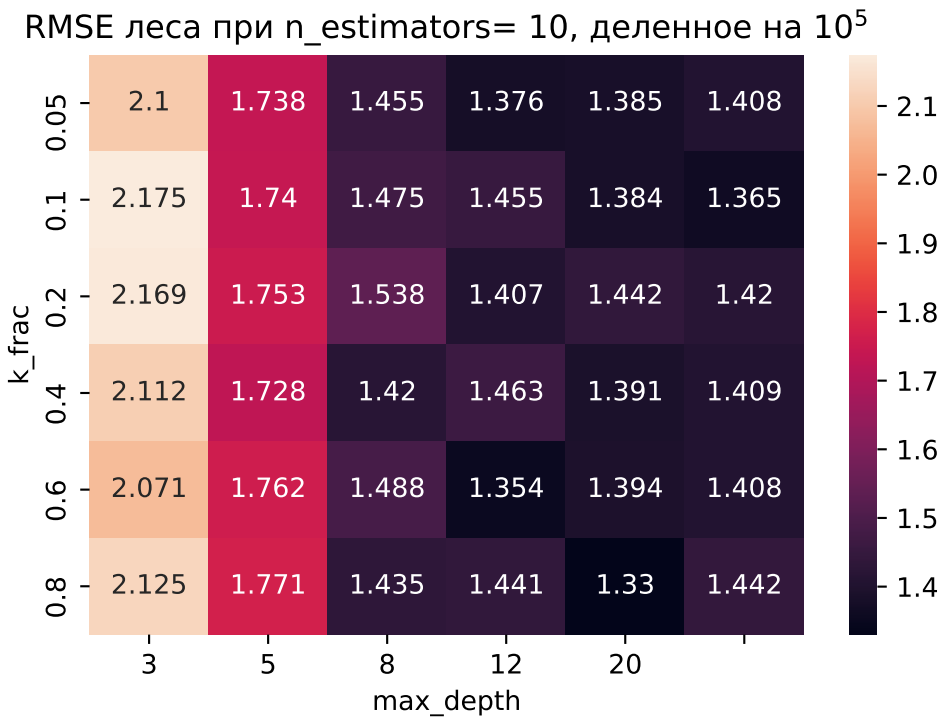
3.3 Эксперименты

На занятиях рассказывалось, что бэггинг показывает наилучшие результаты, когда объединяется много переобученных моделей - каждая из них переобучается "по-своему и в результате усреднение по этим различным переобучениям дает маленький разброс. При этом с ростом $n\_estimators$ результаты бэггинга не ухудшаются - просто происходит усреднение базовых моделей.

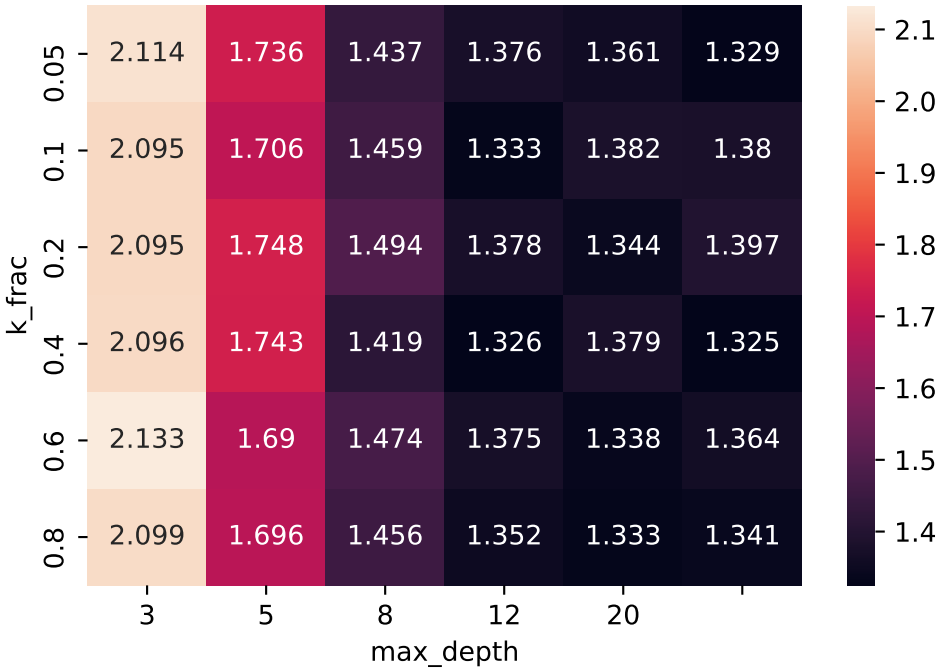
На графиках эти закономерности четко видны - при увеличении $n\_estimators$ все столбцы графика показывают меньший разброс. При этом качество не уменьшается. Отметим также, что качество при $max\_depth = 3, 5$ сильно хуже, чем

при больших значениях параметра. При $max\_depth = 8$ также получаем не очень хорошее качество. А вот при большой (и даже неограниченной глубине) получаем лучшие качетсва, что показывает - в случайном лесу хорошо, когда модели по отдельности переобучаются.

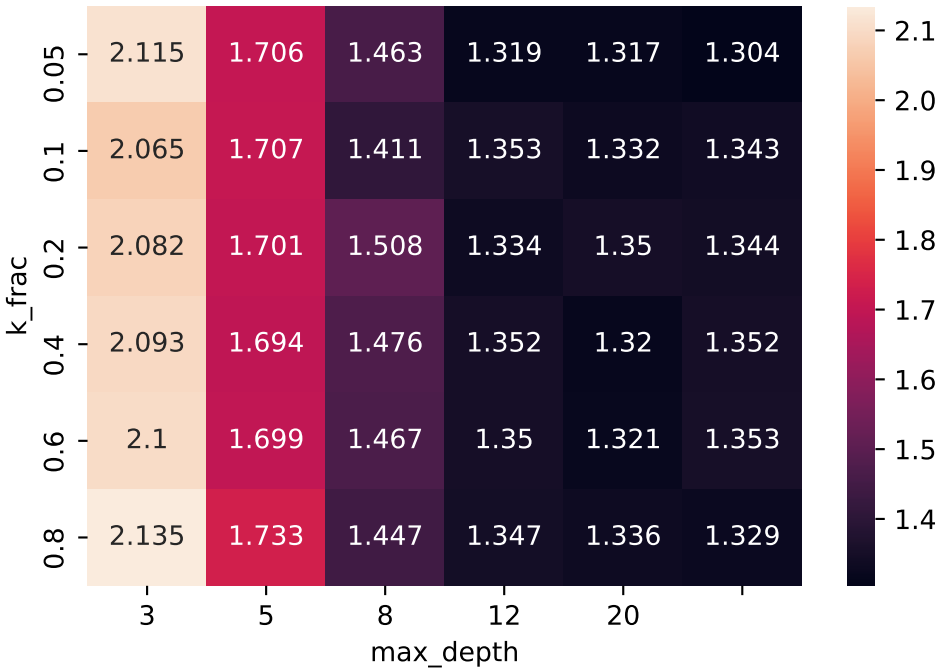
Лучшая модель - $n\_estimators = 500$, $k\_frac = 0.1$, $max\_depth = None$, $RMSE = 129829.43795132353$.



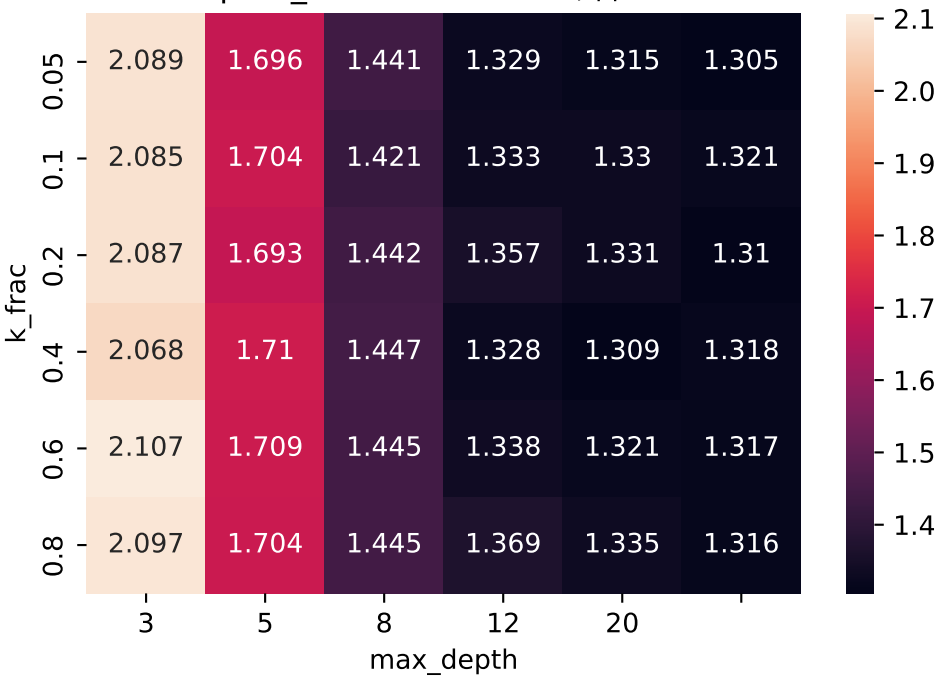
RMSE леса при n_estimators= 25, деленное на 10^5



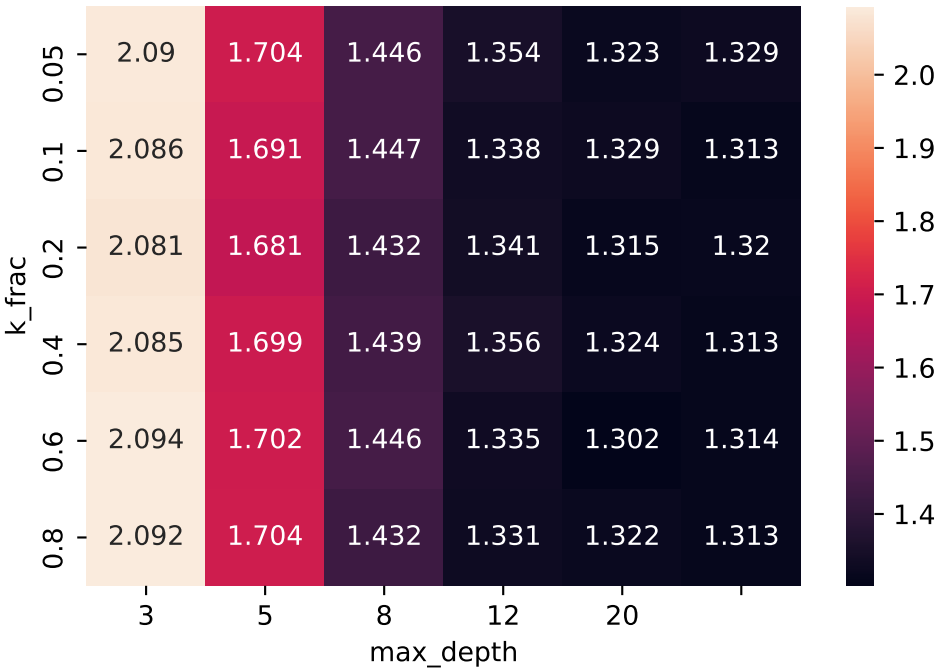
RMSE леса при n_estimators= 50, деленное на 10^5

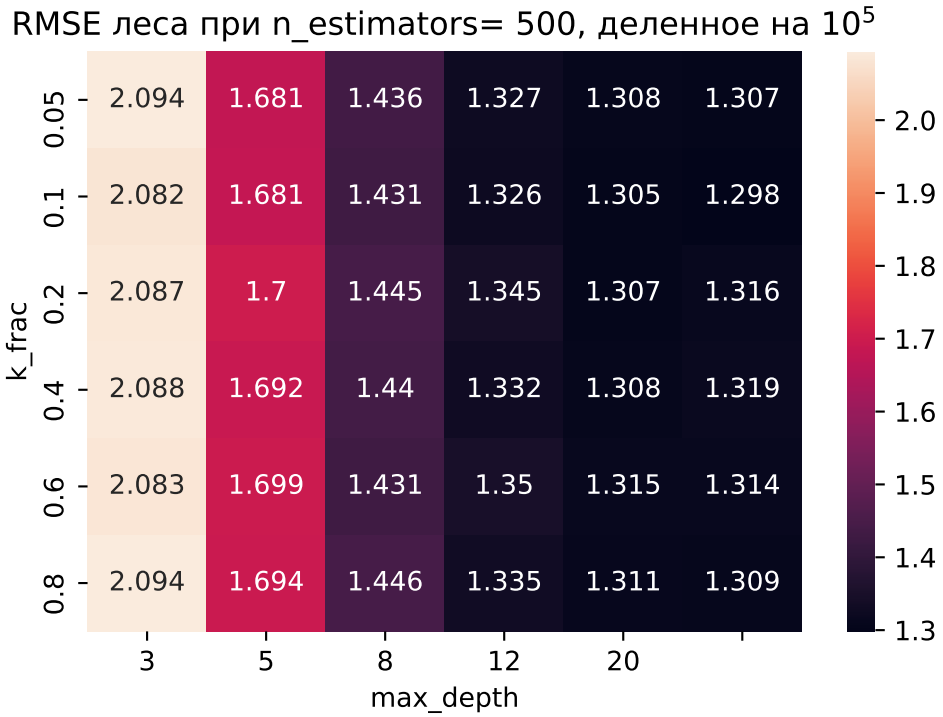


RMSE леса при n_estimators= 100, деленное на 10^5



RMSE леса при n_estimators= 250, деленное на 10^5





4 Градиентный бустинг

4.1 Описание алгоритма

В общем случае градиентный бустинг можно применять над большим количеством разных моделей машинного обучения, однако в рамках данного задании он будет применяться исключительно над решающими деревьями.

blah

4.2 Реализация

Алгоритм реализован на языке *Python* с использованием класса *DecisionTreeRegressor* из библиотеки *sklearn*. Содержится в файле *ensembles.py* в директории *src* *Github*-репозитория.