ММП ВМК

Задание 3. Ансамбли алгоритмов для решения задачи регрессии. Веб-сервер.

Практикум 317 группы, 2024

Начало выполнения задания: 21 ноября 2024 года.
Мягкий Дедлайн: **18 декабря 2024 года, 23:30.**
Жёсткий Дедлайн: **22 декабря 2024 года, 23:30.**

Формулировка задания

В задании необходимо:

1. Написать на языке `Python` собственную реализацию методов **случайный лес** и **градиентный бустинг**. Прототипы функций предоставлены (рис. 1). При написании необходимо пользоваться стандартными средствами языка `Python`, библиотеками `numpy`, `scipy` и `matplotlib`. Из библиотеки `scikit-learn` разрешено импортировать только `DecisionTreeRegressor`.
2. Провести описанные ниже эксперименты с выданными данными. Написать отчёт о проделанной работе (формат PDF). Отчёт должен быть подготовлен в системе LaTeX.
3. Написать реализацию веб-сервера с требуемой функциональностью.
4. Обернуть своё решение в `docker`.
5. Весь код, написанный во время выполнения задания, должен быть размещён в приватном репозитории. Требования к ведению репозитория также описаны ниже.

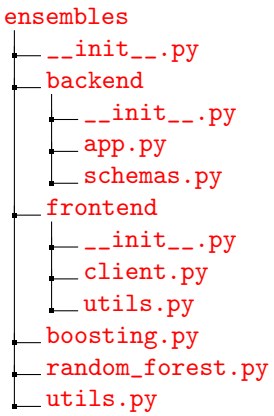

Рис. 1: Структура шаблона.

Экспериментальная часть

Эксперименты для этого задания необходимо проводить на датасете данных о продажах недвижимости **House Sales in King County, USA**. Данные можно скачать по ссылке.

Реализация алгоритмов (10 баллов)

Прототипы питоновских файлов прилагаются (рис. 1).

Подмодули `boosting.py` и `random_forest.py` содержат шаблоны реализации случайного леса и градиентного бустинга. Заполненный результат должен соответствовать классическим реализациям, разобранным на лекциях ММРО [1, 2, 3].

Эксперименты (15 баллов)

1. Проведите предобработку имеющихся данных. Разделите данные на обучение и контроль. Опишите выполненную предобработку данных в отчёте.
2. Исследуйте поведение алгоритма **случайный лес**. Изучите зависимость **RMSE** на отложенной выборке и **время работы алгоритма** в зависимости от следующих факторов:
   - количество деревьев в ансамбле
   - размерность подвыборки признаков для одной вершины дерева
   - максимальная глубина дерева (дополнительно разберите случай, когда глубина не ограничена)
3. Исследуйте поведение алгоритма **градиентный бустинг**. Изучите зависимость **RMSE** на отложенной выборке и **время работы алгоритма** в зависимости от следующих факторов:
   - количество деревьев в ансамбле
   - размерность подвыборки признаков для одной вершины дерева

- максимальная глубина дерева (дополнительно разберите случай, когда глубина не ограничена)
- выбранный `learning_rate` (каждый новый алгоритм добавляется в композицию с коэффициентом `learning_rate`)

Замечание: Для исследования зависимости от количества деревьев не обязательно с нуля переобучать модель.

Инфраструктурная часть

Реализация веб-сервера (15 баллов)

В этой части задания вам предлагается спроектировать HTTP API и веб-интерфейс для взаимодействия с вашей моделью. Считайте, что назначение вашего интерфейса — обучение моделей человеком, который не знает языка `Python`.

Вам предоставлен веб-интерфейс со следующими возможностями (файл `ui.py`):

1. Создание новой модели и выбор гиперапараметров. Загрузка датасета, совпадающего по формату с датасетом из условия (то есть `.csv` файл в котором один из столбцов задаёт целевую переменную, а подмножество остальных столбцов задает признаки объектов выборки).
2. Просмотр информации о модели и полученных кривых обучения.
3. Инференс с использованием ранее обученной модели.

Файл со `streamlit` приложением запрещается модифицировать в иных целях кроме как для выполнения бонусного задания. Ваша задача написать HTTP API, соответствующий данному веб-интерфейсу, с помощью фреймворка `FastAPI`.

1. Подпакет `backend` содержит пример одной «ручки» `FastAPI` приложения. Разрешается создавать какие удобно «ручки», но они должны находиться в подмодуле `app.py`. Единственное требование: у всех «ручек» должны быть аннотированы все аргументы (с помощью `Annotated[]`, к примеру) и возвращаемые значения (с помощью `->` или аргумента декоратора `response_model`).
2. Подпакет `frontend` содержит шаблон клиента для общения с бэкендом. Шаблон полностью совместим с предоставленным веб-интерфейсом.

Решение должен быть обёрнуто в `docker`. Образ должен быть загружен на `dockerhub.com`. Поскольку в задании требуется запустить два сервиса (фронт и бэк), необходимо использовать `docker-compose.yml`.

Ведение проекта (10 баллов)

Весь код вашего решения должен быть выложен в приватный `github` репозиторий. Ваш проект должен быть организован в соответствии с рис. 2. По необходимости вы можете создавать другие дополнительные файлы и директории. Полный балл за данный пункт может быть выставлен только при условии качественного ведения репозитория. Качественное ведение включает в себя следующие требования:

1. Основная разработка ведётся не в `master`, а в отдельных ветках. Ветка соответствует решению одной глобальной задачи.
2. Одно важное изменение в коде — один коммит в системе.
3. Обновление `master` ветки происходит посредством `pull request` и `merge`.
4. Сообщения коммитов и описание `pull request` написаны понятно и содержательно.

Рис. 2: Структура репозитория

Качество кода влияет на итоговую оценку, код должен быть структурированным и понятным. Ваш код должен удовлетворять кодстайлу. В частности, проходить проверку линтерами:

```
# Linter for Dockerfile
cat Dockerfile | docker run --rm -i hadolint/hadolint
# Linter for shell scripts
docker run --rm -v "/path/to/script/folder:/mnt" koalaman/shellcheck:stable script.sh
# Linter for Python scripts
flake8 script.py --max-line-length=120 ; pylint script.py --max-line-length=120 --disable="C0103,C0114,C0115"
```

В репозитории должен быть указан `README.md` файл, объясняющий как необходимо пользоваться вашей системой, как билдить докер-образ и запускать контейнер и проч. В `README.md` необходимо подробно описать не только процесс сборки и запуска контейнера, но и инструкцию по использованию всех реализованных функций в приложении. Использование иллюстраций и скриншотов в инструкции крайне желательно.

Не забудьте закоммитить отчет в виде файла `report.pdf`.

Бонусная часть (до 10 баллов)

Добавьте функционал для сравнения сразу нескольких экспериментов, чтобы можно было чекбоксами выбрать существующие эксперименты и отобразить на одном графике их кривые обучения.

Список литературы

[1] *Воронцов К. В.* Линейные ансамбли. — http://www.machinelearning.ru/wiki/images/3/3a/Voron-ML-Compositions1-slides.pdf. — 2021.

[2] *Воронцов К. В.* Продвинутые методы ансамблирования. — http://www.machinelearning.ru/wiki/images/2/21/Voron-ML-Compositions-slides2.pdf. — 2021.

[3] *ММРО Лекция.* Введение в ансамбли алгоритмов. — https://github.com/mmp-mmro-team/mmp_mmro_fall_2024/blob/main/seminars/Seminar_11_intro_to_ensembles/lect.pdf. — 2024.