ВМК

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

Практикум 317 группы, 2021

Начало выполнения задания: 24 ноября 2021 года, 02:00.
Мягкий Дедлайн: **15 декабря 2021 года, 08:00.**
Жёсткий Дедлайн: **22 ноября 2021 года, 08:00.**

Формулировка задания

Данное задание направлено на ознакомление с алгоритмами композиций.
В задании необходимо:

1. Написать на языке `Python` собственную реализацию методов **случайный лес** и **градиентный бустинг**. Прототипы функций должны строго соответствовать прототипам, описанным в спецификации. При написании необходимо пользоваться стандартными средствами языка `Python`, библиотеками `numpy`, `scipy` и `matplotlib`. Библиотекой `scikit-learn` пользоваться запрещается, если это не обговорено отдельно в пункте задания.
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. Обернуть своё решение в docker контейнер.
4. Весь код, написанный во время выполнения задания, должен быть размещён в приватном репозитории. Требования к ведению репозитория также описаны ниже.

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 всех необходимых функций описаны в файлах, прилагающихся к заданию. Среди предоставленных файлов должны быть следующие модули и функции в них:

1. Модуль `ensembles.py` с реализациями случайного леса и градиентного бустинга. Алгоритмы должны соответствовать классическим реализациям, разобранным на лекциях ММРО [1], [2].

Замечание: Для одномерной оптимизации используйте функцию `minimize_scalar`. Разрешается использовать класс `DecisionTreeRegressor` из библиотеки `scikit-learn`.

Эксперименты (15 баллов)

1. Проведите предобработку имеющихся данных. Разделите данные на обучение и контроль, переведите данные в `numpy.ndarray`. 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 одного дерева
   - максимальная глубина дерева (дополнительно разберите случай, когда глубина неограничена)
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 одного дерева
- максимальная глубина дерева (дополнительно разберите случай, когда глубина неограничена)
- выбранный `learning_rate` (каждый новый алгоритм добавляется в композицию с коэффициентом $\alpha \cdot$ `learning_rate`)

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

Инфраструктурная часть

Реализация веб-сервера (15 баллов)

В этом задании вам предлагается спроектировать веб-интерфейс для взаимодействия с вашей моделью. Считайте, что назначение вашего интерфейса — обучение моделей человеком, который не знает языка `Python`. Это творческое задание, вы можете использовать при реализации всё, что считаете нужным.

В интерфейсе обязательно реализовать следующие функции:

1. (7 баллов) Функция создания новой модели. Необходимо предусмотреть возможность указывать тип (случайный лес или градиентный бустинг) и гиперпараметры модели. В интерфейсе должна быть предусмотрена функция обучения модели на произвольном датасете, совпадающем 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). Также, предусмотрите возможность передавать данные для валидации.

2. (4 балла) Функция просмотра информации о модели. Пользователь должен иметь возможность получать информацию о гиперпараметрах модели, датасете, на котором она обучалась, а также о значении функции потерь на обучающей и валидационной (при наличии) выборках после каждой итерации.

3. (4 балла) Функция для выполнения предсказаний с использованием ранее обученной модели. Считайте, что данные для предсказания представленны в том же формате, что и данные для обучения модели (за исключением отсутствия столбца с целевой переменной).

Замечание: При реализации интерфейса необходимо было предусмотреть все возможные проверки на входные данные и действия пользователя. Например, сайт не должен падать при передаче невалидного файла с обучающими данными.

Ведение проекта (10 баллов)

Весь код вашего решения должен быть выложен в приватный `github` репозиторий. Ваш проект должен быть организован в соответствии с Рисунком 1. По необходимости вы можете создавать другие дополнительные файлы и директории. За качественное ведения репозитория могут назначаться бонусные баллы. Под качественным ведением подразумевается:

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`.

Решение должен быть обёрнуто в docker контейнер. В репозитории должен содержаться `Dockerfile`, а также инструкция по его сборке. Образ вашего контейнера должен быть загружен на `dockerhub.com`.

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. Подробнее про Markdown разметку можно прочитать в документации. 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.

```
github.com/<YourNickname>/<RepoName>
├── src ..........................Исходники веб-приложения и реализация алгоритмов
│   ├── ensembles.py
│   └── ...
├── scripts ......................Скрипты для сборки и запуска docker контейнера
│   ├── build.sh
│   ├── run.sh
│   └── ...
├── Dockerfile ...........................Описание контейнера с веб-приложением
├── report.pdf ...................................................Отчёт по заданию
├── ...
└── README.md ....................................Описание проекта. Документация
```

Рис. 1: Требуемая структура github репозитория

Бонусная часть (до 10 баллов)

В этом задании нет чётких условий для бонусной части. Дополнительные баллы могут быть поставлены за любой хорошо реализованный дополнительный функционал, не описанный в задании. Не забудьте в отчёте и в посылке в anytask написать, что за дополнительный функционал вы реализовали.

Несколько идей для получения дополнительных баллов:

1. (до 3 баллов) Использование интерактивных графиков, например, с помощью библиотеки `plotly`.
2. (до 3 баллов) Реализация веб-интерфейса без перезагрузки страниц, например, с помощью `AJAX`.

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.