

Задание [Base][11]. Градиентный бустинг. Часть 2

Курс по методам машинного обучения, 2024-2025, Юлиан Сердюк

1 Характеристики задания

- **Длительность:** 1 неделя
- **Base**
 - **Кросс-проверка:** 4 балла + 3 бонус; в течение 1 недели после дедлайна; нельзя сдавать после жесткого дедлайна
 - **ML-задание:** 6 баллов; можно сдавать после дедлайна со штрафом в 40%; публичная и приватная части; PER8
- **Почта:** ml.cmc@mail.ru
- **Темы для писем на почту:** ВМК.ML[Задание [Base][11]][peer-review], ВМК.ML[Задание [Base][11]][unit-tests]

Кросс-проверка: После окончания срока сдачи, у вас будет еще неделя на проверку решений как минимум **3х других студентов** — это **необходимое** условие для получения оценки за вашу работу. Если вы считаете, что вас оценили неправильно или есть вопросы, можете писать на почту с соответствующей темой письма

2 Описание задания

Внимание! Это описание ML задания про градиентному бустингу. Также на cv-gml.ru имеется ноутбук по соответствующему заданию, в котором вы можете найти дополнительную информацию по методам, которые будут полезны для решения данного задания.

Привет, ребята! Сегодня мы с вами будем заниматься благодарным делом, а именно предсказывать популярность фильмов. Описание присутствует в основном ноутбуке по градиентному бустингу, поэтому просто скопирую сюда самое важное.

3 Кросс-проверка

Внимание! Отправлять задание нужно в систему во вкладку с пометкой (notebook).

Внимание! Отправлять задание нужно только с расширением `ipynb`! После отправки проверьте корректность загруженного задания в систему, просмотрев глазами загруженное решение (оно автоматически конвертируется в `html`). Как это сделать, можно найти в tutoriale по проверяющей системе на сайте курса.

Внимание!: Перед сдачей проверьте, пожалуйста, что не оставили в ноутбуке где-либо свои ФИО, группу и так далее — кросс-рецензирование проводится анонимно.

4 ML-задание

В некотором царстве, некотором государстве была развита кинопромышленность. Новые фильмы в этом государстве показывают по интернету, а пользователи после просмотра могут дать фильму некоторую "награду". Наша цель - предсказать число наград для фильма.

4.1 Описание данных

В нашем распоряжении имеются следующие данные:

awards - количество наград, полученных фильмом от пользователей (целевое значение)

potions - количество магических зелий, потраченных на создание спец-эффектов

genres - жанры созданного фильма

questions - количество вопросов, заданных пользователями на соответствующих форумах об этом фильме до премьеры

directors - режиссеры фильма (если неизвестны, то unknown)

filming_locations - области, в которых снимался фильм

runtime - продолжительность фильма в некоторых единицах, принятых в этом государстве

critics_liked - количество критиков из 100, присудивших награды фильму на предварительных закрытых показах

pre-orders - количество зрителей, заранее купивших билеты на первый показ

keywords - ключевые слова, описывающие содержание фильма

release_year - год, во котором фильм был показан (конечно, в летоисчислении этого государства)

Следующие поля появляются несколько раз с разными значениями i:

actor_i_known_movies - количество известных фильмов актера i (i от 1 до 3)

actor_i_postogramm - количество подписчиков в социальной сети "по сто грамм"актера i (i от 1 до 3)

actor_i_gender - количество пол актера i (i от 1 до 3)

actor_i_age - возраст актера i (i от 1 до 3)

4.2 Формат данных

Данные разбиты на три части: тренировочная часть, публичные тестовые команды (которые вы можете скачать) и приватные тестовые файлы.

Данные хранятся в виде jsonl файлов, то есть файлов, в которых каждая строка - это json. Каждая строка соответствует одному объекту в датасете. Ключи этого json соответствуют названиям переменных. В примере решения Вы можете увидеть как читать такие файлы без проблем.

Важно! В тестовом файле сохраняйте данные в том же порядке, в котором они записаны в файл!

4.3 Решение

Внимание! Подбирать оптимальные параметры стоит только на локальном компьютере/в Google Colab! В решении Вы должны использовать регрессор с оптимальными параметрами, которые вы нашли путём перебора по сетке.

В шаблонном файле 'awards_prediction.py' Вы должны реализовать функцию train_model_and_predict, которая получает на вход папку для обучения и теста. На обучении вы обучаете ваш алгоритм, а затем возвращаете предсказания значений awards для всех фильмов из теста. Предсказания должны быть расположены в том же порядке, в котором они находятся в тесте (то есть, не примените где-то случайно shuffle).

В этом файле вы можете создавать любые дополнительные функции и методы, которые нужны вам для решения. Главное – сохранить интерфейс функции train_model_and_predict.

4.4 Советы по решению

В этом задании Вы можете добиться лучшего качества при помощи:

- Предобработки датасета, выбора категориальных переменных и дополнительной фильтрации.
- Выбора лучшего метода обучения и подбора оптимальных параметров с использованием кросс-валидации.

4.5 Разрешенные методы и библиотеки

В качестве метода обучения предлагается использовать любой регрессор, основанный на градиентном бустинге деревьев. Разрешается пользоваться библиотеками `sklearn`, `xgboost`, `lightgbm`, `catboost`.

Жесткого требования использовать градиентный бустинг нет! Градиентный бустинг является одним из лучших методов обучения на сегодняшний день, поэтому будет даже интересно, удастся ли кому-то получить макс. балл альтернативными методами.

Также разрешается пользоваться библиотекой `hyperopt` для подбора параметров модели (параметры стоит подбирать локально, а в систему загружать решение с подобранными параметрами).

4.6 Ограничения для скрипта на `cv-gml.ru`

В этом задании стоит ограничение по времени: 10 минут. Также Вы не можете использовать памяти больше, чем 1024 мб.

4.7 Используемая метрика

В качестве метрики качества используется значение MAE, которое вычисляется по следующей формуле:

$$MAE = \sum_{i=1}^N \frac{|a(x_i) - y_i|}{N},$$

где N - число объектов в тестовой выборке, x_i - вектор признаков i -го объекта, $a(x_i)$ - предсказание на i -ом объекте, y_i - значение целевой переменной для i -го объекта.

4.8 Оценивание

Баллы выставляются по следующим правилам:

2 баллов : $mae \in [0, 2050]$,

1.6 баллов : $mae \in (2050, 2100]$,

1.2 баллов : $mae \in (2100, 2150]$,

0.8 баллов : $mae \in (2150, 2200]$,

0.2 балла : $mae \in (2200, 2300]$,

0 баллов: $mae \in (2300, +\infty]$

Значение `mae` будет посчитано отдельно на публичной и приватной выборках. Количество полученных баллов на приватной выборке будет дополнительно умножено на 2. Таким образом за это задание Вы можете получить до 6 баллов (2 на публичной и 4 на приватной выборках).

4.9 Возможная проблема с `catboost`

При посылке вашего решения в проверяющую систему обучайте регрессор со следующим параметром `train_dir`:

`"CatBoostRegressor(train_dir='/tmp/catboost_info')"`

5 Тестирование

В `cv-gml` можно скачать все файлы, необходимые для тестирования, одним архивом. Для этого просто скачайте `zip`-архив во вкладке **шаблон решения** соответствующего задания и разархивируйте его. Далее следуйте инструкциям по запуску тестирования.

Тесты запускаются с помощью команды:

```
python run.py
```

Учтите, что после запуска скрипта будет создано несколько дополнительных файлов и директорий (это связано с работой тестирующей системы).

Если всё верно, то Вы увидите что-то вроде 'Mark: 1 OK, mae = [2287.341688095019]', т.е. Вашу оценку и значение MAE на публичном датасете.

6 Стиль программирования

При выполнении задач типа unit-tests, ML-задания вам необходимо будет соблюдать определенный стиль программирования (codestyle). В данном случае мы выбирали PEP8 как один из популярных стилей для языка Python. Зачем мы это вводим? Хорошая читаемость кода – не менее важный параметр, чем работоспособность кода :) Единый стиль позволяет быстрее понимать код сокомандников (в командных проектах, например), упрощает понимание кода (как другим, так и вам). Также, привыкнув к какому-либо стилю программирования, вам будет проще переориентироваться на другой.

Полезные при изучении PEP8 ссылки, если что-то непонятно, дополнительный материал можно найти самостоятельно в интернете:

- [Официальный сайт PEP8, на английском](#)
- [Небольшое руководство по основам на русском](#)

Требования к PEP8 мы вводим только для заданий с авто-тестами, требований к такому же оформлению ноутбуков нет. Но улучшение качества кода в соответствии с PEP8 в них приветствуется!

Внимание!!! В проверяющей системе, при несоответствии прикрепляемого кода PEP8, будет высвечиваться вердикт Preprocessing failed. Более подробно посмотреть на ошибки можно, нажав на них:

12.10.2022 cross_val.py
19:22 scalers.py

Preprocessing failed

Результат

Время
работы в
секундах

Preprocessing failed: Runtime error

```
Traceback (most recent call last):
  File "pre.py", line 39, in <module>
    raise RuntimeError(err_message)
RuntimeError: Found 6 errors or warnings in submission.
Detailed info:
scalers.py:6:65: W291 trailing whitespace
scalers.py:17:73: W291 trailing whitespace
scalers.py:31:13: E128 continuation line under-indented for visual indent
scalers.py:38:56: W291 trailing whitespace
scalers.py:44:43: W291 trailing whitespace
scalers.py:80:33: E131 continuation line unaligned for hanging indent
```

Также послышки, упавшие по code style, считаются за попытку сдачи и идут в счет общего количества посылок за день.

Проверить стиль программирования локально можно при помощи утилиты [pycodestyle](#) (в окружении, которое вы ставили, эта утилита уже есть) с параметром максимальной длины строки (мы используем 160 вместо дефолтных 79):

```
pycodestyle --max-line-length=160 your_file_with_functions.py
```