ВМК МГУ

Задание 8. Деревья решений

Курс по методам машинного обучения, 2025-2026, Юлиан Сердюк

1 Характеристики задания

Уровень Base

- **Длительность:** 1 неделя
- **Юнит-тестирование:** 3 балла; Можно сдавать после дедлайна со штрафом в 40%; Публичная и приватная часть.
- **Кросс-проверка:** 7 баллов; в течение 1 недели после дедлайна; Нельзя сдавать после дедлайна.

Уровень Research

- **Длительность:** 2 недели
- **Юнит-тестирование:** 4.5 балла; Нельзя сдавать после дедлайна; Публичная и приватная часть.
- **Кросс-проверка:** 5.5 баллов + 0.5 бонус; в течение 1 недели после дедлайна; Нельзя сдавать после дедлайна.

Почта

- **Почта:** ml.cmc@mail.ru
- **Темы для писем на почту:** ВМК.ML[Задание 8][unit-tests], ВМК.ML[Задание 8][peer-review]

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 какие-то вопросы, можете писать на почту с соответствующей темой письма.

2 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 сконвертируется в html). Как это сделать, можно найти в туториале 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.

3 Base

3.1 Юнит-тестирование (Оценка разбиений)

В этом задании Вам нужно реализовать функцию, которая оценивает качество разбиения множества объектов. На лекции вам рассказали о трех метриках: gini, entropy и classification_error. В этом задании вам нужно реализовать их все. Для вычислений разрешается пользоваться библиотекой numpy.

Замечание: При вычислении этих метрик используйте **натуральный** логарифм!

3.1.1 Формат отправки

В шаблонном файле split_measures.py, вам необходимо реализовать функцию evaluate_measures. На вход эта функция получает список меток классов объектов, которые попали в одно из получившихся разбиений. Эта функция возвращает словарь, в котором содержатся значения всех трёх метрик. Вовращаемый словарь должен содержать три ключа: “gini”, “entropy” и “error”, которым должны быть сопоставлены значения метрик gini, entropy и classification error в типе float ().

Ниже показан схематический пример такого скрипта.

```
def evaluate_measures(sample):
    return {"gini": float(np.sum(sample)),
            "entropy": float(np.min(sample)),
            "error": float(np.max(sample))}

print(evaluate_measures([1, 2, 3, 2, 3, 1, 2, 0]))
```

3.1.2 Используемая метрика

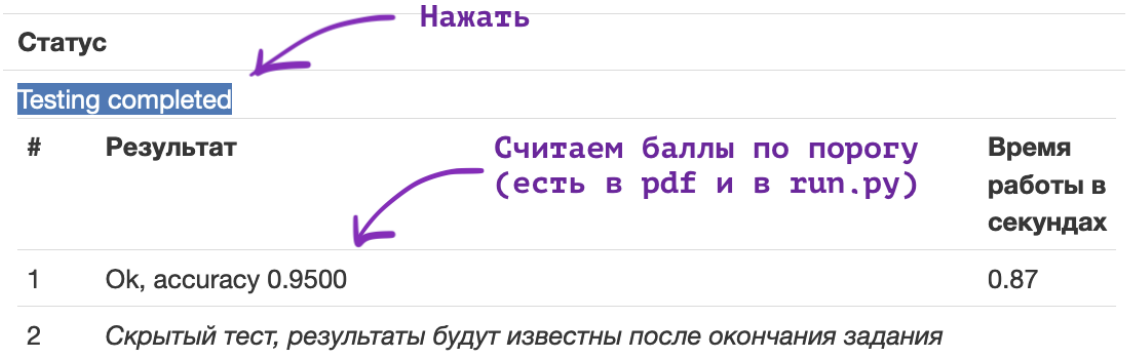
Для успешного выполнения задания необходимо, чтобы Ваше полученное значение отличалось от истинного не более, чем на 0.001 (абсолютное значение разности).

3.1.3 Оценивание

Задание разбито на две части: публичную (один пример) и приватную (содержащую несколько выборок). Для получения оценки в каждой из частей вам необходимо пройти все тесты без ошибок вычислить ответ и получить значение, удовлетворяющее метрике.

В данном задании два теста: публичный, который оценивается из **1 балла**, и приватный (недоступен вам), оцениваемый из **2 баллов**. За публичный тест Вы можете получить гарантированные баллы уже до жёсткого дедлайна, а баллы за приватный тест вам откроются после дедлайна.

В системе, из-за наличия закрытых тестов, вы не сможете увидеть свои баллы до дедлайна даже за открытый тест. Но вы сможете посчитать баллы за открытый тест способом, приведенным на картинке ниже:



3.1.4 Дополнительные материалы

Если вы хотите почитать что это за метрики и как их вычислять, Вы можете обратиться к лекциям Китова или Дьяконова. Воронцов, к сожалению, слишком поверхностно упоминает эти метрики :(

Замечание: Запрещается пользоваться библиотеками, импорт которых не объявлен в файле с шаблонами функций.

Замечание: Задания, в которых есть решения, содержащие в каком-либо виде взлом тестов, дополнительные импорты и прочие нечестные приемы, будут автоматически оценены в 0 баллов без права пересдачи задания.

4 Research

4.1 ML-задание (Предсказание потенциалов)

4.1.1 Описание задания

Внимание! АХТУНГ! АЛЯРМ! Дедлайн у задания жесткий! После оглашения результатов посылки будут запрещены.

Имеется некоторая размером 256 x 256, характеризующая некоторый потенциал. Для каждого потенциала нам необходимо предсказать величину его энергии.

Нельзя использовать: любые решения, являющиеся или содержащие градиентный бустинг, а также любые нейросетевые решения. Вы можете попробовать как и простые решения, так и пройденные в этом семинаре методы ансамблирования. Со списком ансамблей, релизованных в sklearn, можно ознакомиться по ссылке https://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble. Убедитесь, что выбранный Вами ансамбль не использует бустинг. Поскольку этот датасет описан в соответствующем ноутбуке, повторять его описание не буду.

Добиться хорошего качества в этом задании необходимо при помощи трех вещей:

- нахождения лучшего алгоритма обучения
- нахождения оптимальных параметров выбранных алгоритмов (например, в случае леса: размер леса, максимальная глубина, размер признаков или прочего). Рекомендуется ознакомиться с параметрами, используемыми в в каждом алгоритме, чтобы знать, какие параметры Вы можете подбирать (для случайных лесов СОВЕТ: Вы можете изменять не только свойства деревьев в лесу, но и оптимизируемый функционал)
- нахождения лучшего способа преобразования 2d матрицы в одномерный вектор признаков (с возможным перемещением/изменением потенциала).

Внимание! В качестве обучаемого регрессора (который делает предсказания) можно выбрать любой метод, не содержащий градиентный бустинг или нейросети!

Для тестирования решения Вы должны скачать публичный датасет, поместить его в папку public_tests и, используя скрипт run.py, протестировать решение так, как это будет сделано проверяющей системой.

4.1.2 Вид решения

В Вашем решении в шаблонном файле potential_prediction.py необходимо реализовать(дополнить) функцию
train_model_and_predict(train_dir, test_dir), которая тренируется на потенциалах из тренировочной папки, а потом возвращает словарь вида {файл:предсказание} для каждого файла из тестовой директории. Датасеты и пример скрипта, реализующий необходимый функционал, Вы можете найти на страницах соответствующего задания на cv-gml.ru.

4.1.3 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.

4.1.4 Время обучения

Внимание! В проверяющей системе установлено ограничение на время работы Вашего скрипта, поэтому настройку методов и подбор параметров Вы должны провести локально (например, в ноутбуке), а в систему загружать уже метод с установленными параметрами.

Время решения ограничено 10 минутами (суммарно время трансформации, обучения и предсказания). Поэтому, если вы будете перебирать параметры по сетке, то такой перебор может вполне занять несколько часов. Вы можете перебрать параметры в прилагаемом ноутбуке, а в качестве решения уже отправить финальный вариант с оптимальными найденными параметрами.

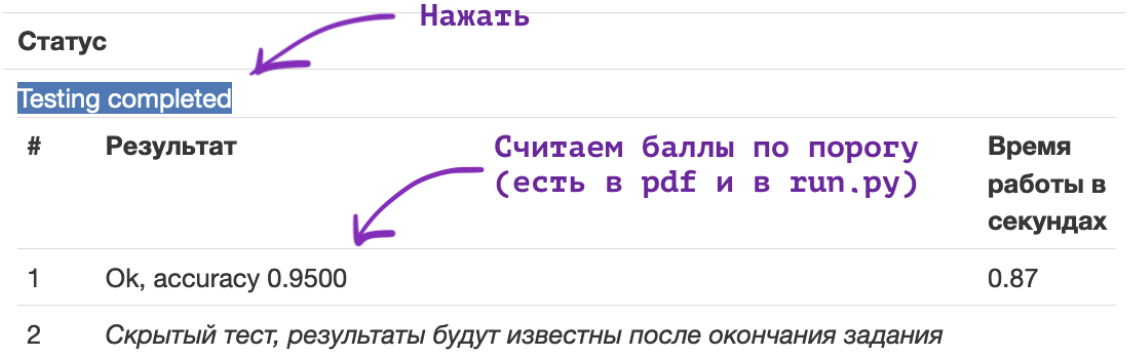
При формировании GridSearch учтите, что время обучения напрямую зависит от количества узлов в сетке. В то же время, слишком большие значения параметров тестировать не стоит, если время обучения на них превышает лимит в 10 минут.

Pro tip! Как это часто бывает в машинном обучении, время обучения напрямую зависит от сложности оптимизационной задачи. Если зависимости, которые Вы хотите найти, достаточно "очевидны", то леса будут строиться относительно быстро. Если Ваши зависимости очень сложны (или, еще хуже, они отсутствуют), то леса будут тратить много времени на поиск оптимального разбиения. Поэтому для ускорения решения рекомендуется не только подбирать правильные параметры, но и найти то преобразование, которое упростит решение задачи.

4.1.5 Видимость результатов

Тестирование проводится на двух тестовых выборках: открытой и закрытой. Открытую вы можете загрузить себе и тестировать свой метод на ней. После загрузки Вашего решения в тестирующую систему оно изначально будет протестировано на открытой выборке. Когда наступит дедлайн задания, Вы увидите результаты своего метода на закрытой тестовой выборке, и по значению целевой метрики вы получите оценку за это задание.

В системе, из-за наличия закрытых тестов, вы не сможете увидеть свои баллы до дедлайна даже за открытый тест. Но вы сможете посчитать баллы за открытый тест способом, приведенным на картинке ниже:



4.1.6 Оценивание

Баллы выставляются по следующим правилам:
1.5 баллов : mae $\in$ [0, 0.007],
1 балл: mae $\in$ (0.007, 0.01],
0.6 баллов: mae $\in$ (0.01, 0.02],
0.3 балла: mae $\in$ (0.02, 0.05],
0 баллов: mae $\in$ (0.05, $+\infty$]

Значение mae будет посчитано раздельно 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 4.5 баллов.

4.1.7 Советы по решению

Внимание! В функции train_model_and_predict рекомендуется менять лишь строку, в которой определяется регрессор, а именно

```
regressor = Pipeline([('vectorizer', PotentialTransformer()), \
('decision_tree', DecisionTreeRegressor())])
```

Здесь Вы можете переопределить Pipeline, реализовать свой собственный трансформер и поставить предпочтительный алгоритм обучения.

Важно: *перед сдачей проверьте, пожалуйста, что не оставили в ноутбуке где-либо свои ФИО, группу и т.д. — кросс-рецензирование проводится анонимно.*

Важно: *В этом задании в файле с шаблоном* potential_prediction.py *можно использовать дополнительные импорты (например, для обработки признаков), однако модель, которую вы обучаете не должна содержать* ***бустинги и нейросети***

Важно: *задания, в которых есть решения, содержащие в каком-либо виде взлом тестов и прочие нечестные приемы, будут автоматически оценены в 0 баллов без права пересдачи задания.*

5 Стиль программирования

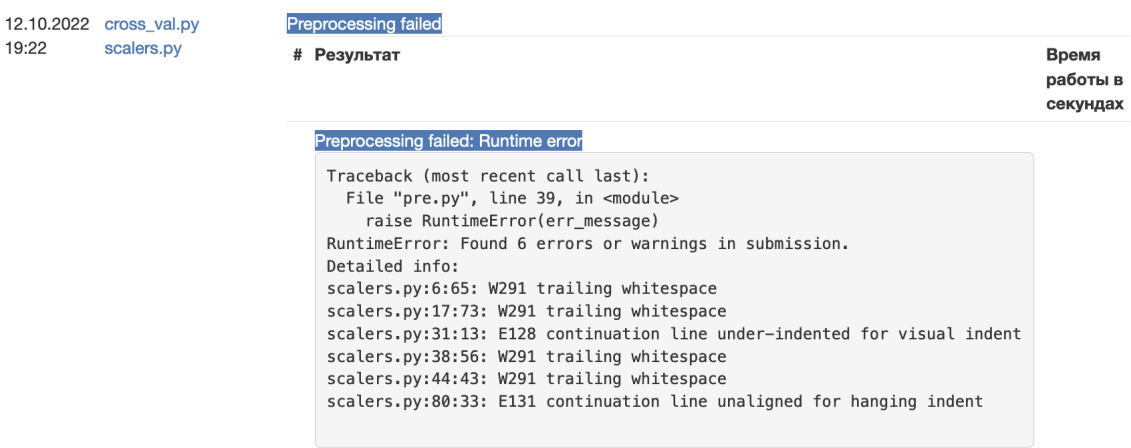
При выполнении задач типа ML-задания вам необходимо будет соблюдать определенный стиль программирования (codestyle). В данном случае мы выбирали PEP8 как один из популярных стилей для языка Python. Зачем мы это вводим? Хорошая читаемость кода – не менее важный параметр, чем работоспособность кода :) 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:



Также посылки, 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 вместе дефолтных 79):

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

6 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 следуй-

те инструкциям по запуску тестирования.

Если всё сделано правильно, то при переходе в соответствующую папку в консоли и запуске команды ‘python run.py‘ Вы не должны получать сообщений об ошибках. Учтите, что после запуска скрипта будет создано несколько дополнительных файлов и директорий (это связано с работой тестирующей системы).

Запуск тестов производится командой

```
python run.py
```