**> Введение**

Сегодня мы будем  своими руками реализовывать градиентный бустинг на основе вычисления Lambda!

Используется всё тот же набор данных, что и в ДЗ к третьему занятию. Более подробно прочитать про градиентный бустинг и принципы его работы можно [по ссылке](https://habr.com/ru/company/ods/blog/327250/). В качестве базового алгоритма для бустинга будем использовать DecisionTreeRegressor из библиотеки sklearn. Как было сказано в лекции, единственное существенное отличие — это целевые метки, на которые обучается каждое дерево: вместо типичных для бустинга ошибок (невязок) используются Lambda-значения. Функцию вычисления лямбд мы рассмотрели на практическом занятии. В решение необходимо осмысленно перенести реализацию в метод \_compute\_lambdas класса Solution.

# > Детали реализации и описание

## Параметры класса

n\_estimators — количество деревьев, которые будут строиться в рамках бустинга.

lr — Learning Rate, коэффициент, на который умножаются предсказания каждого нового дерева в алгоритме (каждое дерево учится предсказывать значение lambda, но не факт, что добавление к текущим предсказаниям такого значения даст оптимум, поэтому весь “путь” оптимизации разбивается на маленькие шаги).

subsample — доля объектов от выборки, на которых обучается каждое дерево (доля одинакова для всех деревьев, но сама подвыборка генерируется на каждом шаге отдельно).

colsample\_bytree — доля признаков от выборки, на которых обучается каждое дерево (доля одинакова для всех деревьев, но сама подвыборка генерируется на каждом шаге отдельно).

Совокупность двух вышеуказанных параметров позволяет реализовать метод случайных подпространств (смотрите описание [по ссылке](https://habr.com/ru/company/ods/blog/324402/#:~:text=Метод%20случайных%20подпространств%20позволяет%20снизить,которые%20также%20выделяются%20случайным%20образом) при необходимости). Понятно, что для применения деревьев (получения предсказания) нужно хранить индексы использованных признаков (но не объектов).

max\_depth и min\_samples\_leaf — параметры DecisionTreeRegressor, отвечающие за глубину построения дерева и минимальное количество в терминальных (финальных) листьях дерева соответственно.

## Методы класса

\_get\_data, \_prepare\_data, \_scale\_features\_in\_query\_groups, \_ndcg\_k вам уже знакомы — можно перенести их реализацию из прошлого домашнего задания с тем лишь отличием, что для удобства срезов по индексам размерности ys\_train и ys\_test должны быть N∗1, где N-количество объектов (без этого грейдер будет отчитываться об ошибке).

save\_model и load\_model — методы, отвечающие за сохранение и загрузку модели. Вам необходимо самостоятельно определить набор полей (подсказка: их минимум 3), которые нужно сохранять после тренировки и загружать для предсказания. После load\_model необходимо добиться, чтобы модель могла давать адекватные предсказания (те же самые, что и до сохранения). Сохранение и загрузку реализуйте через модуль pickle. Пример:

state = {…} f = open(path, 'wb') pickle.dump(state, f)

Реализация этих методов необходима, поскольку вам предстоит локально обучить модель, сохранить её и отправить вместе с кодом на проверку. Критерий по достигаемой метрике смотрите в конце текста задания. Модель должна загружаться inplace, т.е. непосредственно инициализировать поля объекта класса Solution, у которого вызван метод (а не порождать новый объект).

Предсказания формируются в методе predict. На вход поступает тензор данных размерности N∗D, где N — количество объектов, D — количество признаков. На выходе ожидается применённый алгоритм бустинга, т.е. тензор предсказаний.

Расчёт метрики по набору данных должен производиться методом \_calc\_data\_ndcg — в нём необходимо проитерироваться по группам запросов, посчитав в каждой NDCG, после чего вернуть усреднённое значение метрики.

## Методы для тренировки

\_train\_one\_tree — метод для тренировки одного дерева. Принимает на вход cur\_tree\_idx — номер текущего дерева, который предлагается использовать в качестве random\_seed для того, чтобы алгоритм был детерминирован. train\_preds — суммарные предсказания всех предыдущих деревьев (для расчёта лямбд). В рамках метода необходимо рассчитать лямбды для каждой группы в тренировочном наборе данных, затем применить метод случайных подпространств, сделав срез по признакам (случайно выбранная группа, размер которой задан параметром colsample\_bytree) и по объектам (тоже случайно выбранная группа, размер зависит от параметра subsample). Затем произвести тренировку одного DecisionTreeRegressor. Возвращаемые значения — это само дерево и индексы признаков, на которых обучалось дерево.

fit — генеральный метод обучения K деревьев, каждое из которых тренируется с использованием метода \_train\_one\_tree. Изначальные предсказания до обучения предлагается приравнять к нулю и от этих значений отталкиваться при обучении первого дерева. Все обученные деревья необходимо сохранить в список, хранящийся в атрибуте trees класса Solution. Для простоты и ускорения работы предлагается рассчитывать предсказания для всех тренировочных и валидационных данных после обучения каждого дерева (но досчитывать только изменения за последнее дерево, храня в памяти предсказания всех предыдущих). Следите за лучшим значением NDCG (хранить в переменной best\_ndcg) — после окончания тренировки нужно обрезать те последние N деревьев, которые лишь ухудшают метрику на валидации. Например, вы обучили 100 деревьев, и лучший результат был достигнут на 78-м. Тогда self.trees нужно обрезать до 78-го дерева, чтобы модель при предсказании работала лучше всего.

Как было сказано выше, необходимо локально обучить и сохранить модель, которая на валидации показывает NDCG не меньше, чем 0.431. За это даются дополнительные баллы, это необязательная часть домашнего задания. Для подбора параметров обучения такой модели после реализации класса Solution предлагается использовать библиотеку hyperopt — краткий ознакомительный туториал можно найти [по ссылке](https://github.com/Yorko/mlcourse.ai/blob/master/jupyter_russian/tutorials/hyperopt_tutorial_stalkermustang%20.ipynb).

Ваше решение должно за 100 деревьев и не более чем за 5 минут с нуля обучаться до NDCG=0.405. Иначе баллы не будут начислены.