A. Прямая снова крутится

Рассмотрим линейный классификатор на плоскости $\mathbb{R}^2$, задаваемый прямой $x + Ay = 0$. Точка (x, y) относится к первому классу, если $x + Ay > 0$. В противном случае, если $x + Ay \leqslant 0$, точка относится ко второму классу.

Приведите пример таких **целых** чисел A, b и c, что для трёхчлена $P(x) = x^2 + bx + c$ точка $(2, P(2))$ относится к первому классу, а точки $(1, P(1))$ и $(3, P(3))$ — ко второму классу.

B. Генератор случайности

Генератор случайных чисел работает следующим образом. Если задать ему натуральное число $m \geqslant 6$, он выдаст последовательно 10 натуральных чисел. Каждое число выбирается им независимо и равновероятно из множества $\{1, 2, \ldots, m\}$.

В силу системной ошибки, результат генератора выводится на экран следующим образом: числа $1, 2, 3, 4, 5$ выводятся без изменений, а вместо любого другого числа выводится число 6.

При каком натуральном $m \geqslant 6$ вероятность получить на экране последовательность чисел

6, 2, 6, 6, 1, 6, 4, 6, 3, 5

максимальна?

C. Градиентный спуск на листочке

Рассмотрим функцию $f(x, y) = x^{20} + y^{26}$. Ниже приведён алгоритм, который по начальной точке (a, b) строит последовательность точек (X_i, Y_i), $i = 0, 1, 2, \ldots$
Ниже в коде `decrease_lr = False` для пункта (a), и `decrease_lr = True` для пункта (b).

```
X_0, Y_0 ← a, b
dx, dy ← 0, 1
lr ← 1
for i = 1, 2, 3, ... do
    dx, dy ← dy, -dx
    if decrease_lr and i · lr ⩾ 2 then:
        lr ← lr / 2
    end if
    X_i, Y_i ← X_{i-1}, Y_{i-1}
    if f(X_i + dx · lr, Y_i + dy · lr) < f(X_i, Y_i) then
        X_i ← X_i + dx · lr
        Y_i ← Y_i + dy · lr
    end if
end for
```

Докажите, что, какая бы точка ни была начальной, справедливы следующие утверждения.

(a) Найдётся такой номер N, что расстояние от точки (X_N, Y_N) до начала координат не превосходит 1.

(b) Найдётся такой номер N_0, что для любого номера $N \geqslant N_0$ расстояние от точки (X_N, Y_N) до начала координат не превосходит $\frac{1}{1000}$.

D. Лазер

Пол прямоугольной камеры с зеркальными стенами имеет форму клетчатого прямоугольника 1000×3000. В **узлах сетки** расположены датчики температуры (всего $1001 \cdot 3001$ датчиков). Изначально все датчики показывают температуру 0°C.

В **центрах некоторых клеток** расположено по одному устройству, испускающему лазерные лучи. Каждое устройство испускает лазерный луч по диагонали клетки, в центре которой оно находится (то есть в одном из четырех возможных направлений, в сторону одного из узлов соответствующей клетки). При этом то же устройство поглощает лучи, приходящие из противоположного узла. На лучи трёх других направлений устройство не влияет.

Всего установлено 300 устройств, которые излучают по одному лучу. Каждый луч отражается зеркально от стенок (попадая в угол, луч отражается в противоположном направлении), пока не поглотится каким-то из устройств.

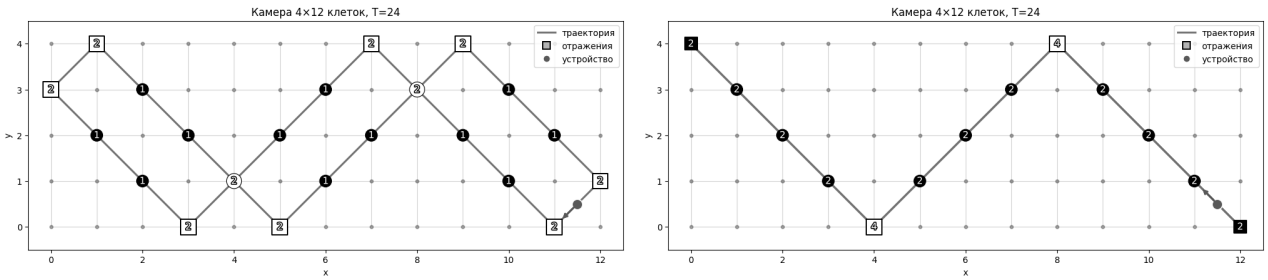
Лазерный луч, «посещая» очередной датчик температуры, увеличивает его показание на 1°C. Если же в точке, где располагается датчик, луч отражается, он увеличивает показание соответствующего датчика сразу на 2°C. Назовём *тепловой картой* матрицу A размера 1001×3001, элементы которой равны итоговым показаниям соответствующих датчиков температуры.

Для проверки корректности показаний датчиков, полученные значения анализируют следующим образом. Выбирается квадратная матрица K размера $s \times s$, называемая *ядром* размера s, после чего вычисляется *свёртка* матрицы A с ядром K, то есть матрица $B = A \star K$ размера $(1002 - s) \times (3002 - s)$, где

$$B[i][j] = \sum_{u=0}^{s-1} \sum_{v=0}^{s-1} A[i+u][j+v]K[u][v], \qquad 0 \leqslant i \leqslant 1001 - s, \quad 0 \leqslant j \leqslant 3001 - s.$$

Назовём ядро K *определяющим*, если для любой тепловой карты A все элементы матрицы B равны нулю, при этом не все элементы ядра K равны 0. При каком наименьшем значении s существует определяющее ядро размера s?

Ниже на рисунке вы можете видеть как запущенный луч изменит температуру в датчиках до момента поглощения.



E. Новая выборка

Модель по вещественному числу x предсказывает вещественное число методом *линейной регрессии*, то есть по формуле $f(x) = ax + b$, где $a, b \in \mathbb{R}$ — параметры модели f.

Дана *обучающая выборка* $(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)$, где $x_1, \dots, x_n, y_1, \dots y_n \in \mathbb{R}$. Модель f обучается на этих данных: параметры a и b выбираются *методом наименьших квадратов*, то есть так, чтобы значение выражения

$$\sum_{i=1}^{n} (y_i - f(x_i))^2$$

было минимальным.

Определим *коэффициент детерминации* R_0^2, посчитанный на этой выборке, следующим образом:

$$R_0^2 = 1 - \frac{\sum\limits_{i=1}^{n} (y_i - f(x_i))^2}{\sum\limits_{i=1}^{n} (y_i - \bar{y})^2}, \quad \text{где} \quad \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i.$$

Построим новую выборку, добавив к исходной n объектов $(x_i, f(x_i)), i = 1, 2, \dots, n$. Пусть на получившейся выборке из $2n$ объектов обучается модель $g(x) = cx + d$ также методом наименьших квадратов. Обозначим через R_1^2 коэффициент детерминации для модели g, посчитанный на выборке из всех $2n$ объектов.

Предполагается, что параметры обеих моделей удалось подобрать методом наименьших квадратов единственным образом.

Выразите R_1^2 через R_0^2.

F. Вымышленная ситуация

Артур и Таня готовят вычислительные мощности к практическому туру для 1000 участников финала всероссийской олимпиады школьников по ИИ. Изначально они хотели развернуть всю инфраструктуру в облаке и выдавать видеокарты (GPU) из общего набора, но, чтобы исключить сетевые задержки, было решено собрать каждому участнику персональный кластер из 8 устройств.

На складе партнёров олимпиады есть по 8000 устройств каждого из трех типов.

- Тип 1 (зеленые) — CUDA, быстрые, но горячие.
- Тип 2 (красные) — поддерживают открытые стандарты, подходят для обучения больших языковых моделей (LLM), но требуют сложной настройки драйверов.
- Тип 3 (желтые) — экспериментальные ускорители (TPU).

По условиям поставки устройства типа 1 можно получать только партиями по 7 штук; устройств каждого типа должно быть не меньше, чем 2026; всего со склада следует взять ровно 8000 устройств. Если в соответствии с этими условиями можно привезти со склада y_1 устройств первого типа, y_2 — второго и y_3 — третьего, будем называть тройку целых чисел (y_1, y_2, y_3) *допустимой*, то есть:

$$y_1 + y_2 + y_3 = 8000, \qquad y_t \geqslant 2026 \ (t = 1, 2, 3), \qquad y_1 \vdots 7.$$

Каждый участник имеет персональный шифр $i = 1, 2, \ldots, 1000$. Устройства, поставленные со склада, распределяются между участниками. Пусть участник i получает персональный кластер $S_i = (x_{i,1}, x_{i,2}, x_{i,3})$, в котором $x_{i,1}, x_{i,2}, x_{i,3}$ — количество устройств первого, второго и третьего типа соответственно. Числа $x_{i,1}, x_{i,2}, x_{i,3}$ — целые неотрицательные, причём $x_{i,1} + x_{i,2} + x_{i,3} = 8$.

Оказалось, что кто-то из участников предпочитает «зеленых», кто-то фанат открытых стандартов и выбирает «красных», а кто-то любит необычных «жёлтых». Для каждого участника заданы *предпочтения* устройств $u_{i,1}, u_{i,2}, u_{i,3}$, указанные во входном файле `gpu.csv`.

Файл содержит заголовок и 1000 строк с 3 столбцами с данными: в i-й строке записаны три целых числа $u_{i,1}, u_{i,2}, u_{i,3}$ (именно в таком порядке), соответствующие предпочтениям участника i.

Полезность кластера $S = (x_1, x_2, x_3)$ для участника i задаётся формулой

$$V_i(S) = u_{i,1}x_1 + u_{i,2}x_2 + u_{i,3}x_3.$$

Распределение устройств называется *честным*, если неравенство $V_i(S_i) \geqslant V_i(S_j)$ справедливо для любых двух участников i, j.

(a) Приведите пример допустимой тройки значений (y_1, y_2, y_3), при которой честного распределения не существует, и докажите это.

(b) Существует ли допустимая тройка (y_1, y_2, y_3), при которой существует честное распределение?

A. Что посмотреть?

Условие

В онлайн-кинотеатре пользователю нужно рекомендовать один фильм из нескольких возможных вариантов.

Для этого система подбирает для пользователя пять фильмов-кандидатов и задаёт правило, по которому нужно выбрать один из них. Такую задачу выбора будем называть **запросом**.

Каждый запрос описывается следующими данными:

- `user_id` — идентификатор пользователя;
- `query_type` — тип запроса, то есть правило выбора;
- c_1, c_2, c_3, c_4, c_5 — идентификаторы пяти различных фильмов-кандидатов.

Каждая строка файла `queries_A.csv` задаёт один запрос: для указанного пользователя из пяти предложенных фильмов нужно выбрать один.

Для каждого запроса нужно вывести `item_id` выбранного фильма.

Формат ввода

Используйте приложенные к задаче файлы:

1. `queries_A.csv`
2. `items_A.csv`
3. `events_A.csv`
4. `item_meta_A.json`
5. `sessions_A.json`

Примечания

К задаче приложен файл `baseline_A.ipynb` с примером чтения входных файлов, фильтрацией запросов по типу для каждой подзадачи и подробным описанием данных.

Формат вывода

Нужно вывести CSV-файл с заголовком

```
query_id,item_id
```

Для каждого `query_id`, относящегося к соответствующей подзадаче, в ответе должна быть ровно одна строка. Лишних строк быть не должно.

Система оценивания

Максимальный балл за каждую подзадачу — 20.

Решение считается верным, если для каждого `query_id`, относящегося к соответствующей подзадаче, указан правильный `item_id` по правилам задачи.

Итоговые баллы за задачу выставляются по **лучшей** посылке.

A1. Подзадача 1

При решении этой подзадачи следует отфильтровать строки `queries_A.csv`, оставив только запросы типа `favorite_genre`, и вывести ответы только для них.

Для запроса `favorite_genre` для каждого из пяти кандидатов берётся его жанр, после чего для этого жанра вычисляется сумма весов всех событий данного пользователя по фильмам того же жанра, где

- `open = 1`
- `finish = 2`
- `like = 3`

Выбирается кандидат, для жанра которого эта сумма максимальна.

При равенстве:

1. фильм с меньшей `duration`;
2. фильм с меньшим `item_id`.

A2. Подзадача 2

При решении этой подзадачи следует отфильтровать строки `queries.csv`, оставив только запросы типа `actor_match`, и вывести ответы только для них.

Для каждого запроса `actor_match` формируется множество `watched_actors` актёров фильмов, которые пользователь

1. Или посмотрел — `finish`;
2. Или отметил как понравившиеся — `like`.

Рассматриваются только актёры из поля `actors` файла `item_meta_A.json`.

Если поле `actors` отсутствует, его нужно считать пустым списком.

Для кандидата `score` равен числу общих элементов множеств:

множества актёров этого кандидата; множества `watched_actors`.

При равенстве выбирается фильм с минимальным `item_id`.

A3. Подзадача 3

При решении этой подзадачи следует отфильтровать строки `queries_A.csv`, оставив только запросы типа `next_in_session`, и вывести ответы только для них.

Для пользователя рассмотрим все соседние пары фильмов $f_1 \to f_2$, извлечённые из всех списков `path` в файле `sessions_A.json`.

Если `path` = $[x_1, x_2, \ldots, x_k]$, то из него извлекаются пары:

$$x_1 \to x_2$$
$$x_2 \to x_3$$
$$\vdots$$
$$x_{k-1} \to x_k$$

Для кандидата f_2 его score — это количество таких пар $f_1 \to f_2$, где фильм f_1 пользователь когда-либо досмотрел, то есть для него есть событие `finish` в файле `events_A.csv`.

Каждое вхождение пары в `sessions_A.json` учитывается отдельно.

Если у пользователя нет подходящих пар, то score всех кандидатов считается равным 0.

Выбирается кандидат с максимальным score.

При равенстве выбирается фильм с минимальным `item_id`.

B. Сейсмоактивный остров

Условие

Исследователи из центра сейсмологического мониторинга анализируют записи землетрясений на острове Террамотус. Для каждого события данные собирались на двух станциях наблюдения, расположенных в разных концах острова. На каждой станции установлены два измерительных прибора, размещённые в разных частях станции.

Данные передавались со станции в лабораторию по двум каналам: по оптоволоконному кабелю и по беспроводной связи. Но в беспроводной связи были помехи, и некоторые моменты времени в данных были утеряны.

Каждый прибор фиксирует 10 величин каждую минуту в течении 100 минут.

Чтобы ускорить разбор архива, исследователи привлекли ИИ-агента. Исследователи неаккуратно написали запрос и дали агенту слишком много прав. Затем, в процессе работы агент зачем-то переименовал все файлы. После этого исследователям стало непонятно, какие записи относятся к одному и тому же землетрясению: файлы перемешаны, а исходная группировка утрачена. Исследователи запаниковали, но затем взяли себя в руки и обратились за помощью к школьнику Олегу — эксперту в машинном обучении. Помогите ему кластеризовать данные и починить последствия неконтролируемых экспериментов с ИИ.

Известно, что каждому землетрясению соответствуют ровно 8 наблюдений: по 2 прибора на каждой из 2 станций и по 2 способа передачи. Вам нужно понять, какие наблюдения относятся к одному и тому же землетрясению.

Формат ввода

К задаче прикреплены файлы:

- `data_B.npy`, содержащий массив наблюдений $240 \times 10 \times 100$ — 240 наблюдений по 10 значений в течение 100 минут.
- `baseline_B.ipynb` — ноутбук с базовым решением задачи.
- `submission_B.csv` — пример решения, которое нужно отправить в тестирующую систему.

Формат вывода

Для проверки необходимо загрузить архив `solution_B.zip`.

Архив должен содержать:

1. Файл `submission_B.csv` с двумя колонками:
   - `ID` — номер наблюдения в `data_B.npy`;
   - `target` — предсказанное значение целевой переменной.
2. Файл `solution_B.ipynb` — Jupyter Notebook с вашим решением.

Допускается добавлять в архив дополнительные файлы, необходимые для работы решения. При этом архив должен содержать ровно один файл с расширением `.csv` и ровно один файл с расширением `.ipynb`.

Система оценивания

За эту задачу можно получить до 60 баллов.

Данные разбиты на публичную и приватную части. Когда вы отправляете `submission_B.csv`, вам показывается результат на **публичной** части. После завершения этапа результат будет пересчитан на **приватной** части. Публичная и приватная часть не пересекаются.

После окончания этапа ваша метрика будет приведена к 60-балльной шкале по следующему правилу:

- результат **baseline-решения** со значением ARI $\leqslant X$ оценивается в **0 баллов**;
- результат со значением ARI $\geqslant Y$ оценивается в **60 баллов**;
- если значение ARI находится между X и Y, количество баллов вычисляется по формуле линейной интерполяции:

$$\text{Score} = 60 \cdot \frac{\text{ARI} - X}{Y - X}.$$

Значения метрик X и Y будут доступны в тестирующей системе.

Итоговые баллы за задачу выставляются по **последней** посылке.

Метрика оценивания точности ответа

В этой задаче используется метрика **ARI (Adjusted Rand Index)**. Чем большее количество пар объектов участник правильно распределяет по кластерам (например, если оба объекта находятся в разных кластерах и участник также относит их к разным кластерам, ИЛИ оба объекта находятся в одном кластере и участник тоже относит их к одному кластеру), тем выше эта метрика. ARI принимает значение 0 для случайного разбиения на кластеры, значение 1 для идеально правильного разбиения и может принимать отрицательные значения в случае разбиения хуже случайного.

Пример расчета метрики **ARI** на ‘Python‘:

```python
from sklearn.metrics import adjusted_rand_score

labels_true = [0, 0, 1, 1, 2, 2]
labels_pred = [1, 1, 0, 0, 2, 2]

ari = adjusted_rand_score(labels_true, labels_pred)
print("ARI =", ari)
```

C. Ошибка новичка

Условие

Машу позвали на летнюю стажировку в научный институт, который занимается исследованием ареалов обитания животных. В качестве примера интересного задания, с которым Маше предстоит работать, исследователи поделились наблюдениями за местами обитания нового вида ленивцев (*Bradypus procrastinator shkolnikus*). Совсем недавно этот вид был выделен из уже известного ранее. Теперь учёные хотят обнаружить и другие потенциальные зоны его обитания.

Для удобства Маши данные были разбиты на `train` и `test`. Поскольку у исследователей есть только собственные данные о локациях, где уже живут ленивцы, они попросили статистическое агентство Южной Америки предоставить данные и по другим локациям, чтобы расширить тестовый датасет. Каждая строка в данных описывает некоторую локацию и содержит измеренные параметры, характеризующие климат, погоду, растительный и животный мир. Работники института пожаловались, что у них никак не получается построить хорошую модель для этой задачи. Маша посмотрела на данные и сразу заметила две типичные ошибки новичка, допущенные сотрудниками института.

Помогите Маше построить модель, которая по имеющимся данным предсказывает, относится ли локация к потенциальной зоне обитания нового вида ленивцев.

Формат ввода

К задаче прикреплены файлы:

- `map_C.json` — файл следующей структуры:

```
{
    "coastline": [...]
}
```

  `coastline` — массив координат береговой линии, используемый для визуализации.

- `train_C.csv` — обучающая выборка. Каждый объект содержит признаки локации и целевую переменную `target` (бинарная метка `0` или `1`);
- `test_C.csv` — тестовая выборка. Каждый объект содержит идентификатор `id` и признаки локации;
- `baseline_C.ipynb` — ноутбук с базовым решением задачи;
- `submission_C.csv` — пример решения, которое нужно отправить в тестирующую систему.

Формат вывода

Для проверки необходимо загрузить архив `solution_C.zip`.

Архив должен содержать:

1. Файл `submission_C.csv` с двумя колонками:
   - `ID` — идентификатор объекта из тестовой выборки `test`;
   - `target` — предсказанная бинарная метка класса (`0` или `1`).
2. Файл `solution_C.ipynb` — Jupyter Notebook с вашим решением.

Допускается добавлять в архив дополнительные файлы, необходимые для работы решения. При этом архив должен содержать ровно один файл с расширением `.csv` и ровно один файл с расширением `.ipynb`.

Система оценивания

За эту задачу можно получить до 60 баллов.

Данные разбиты на публичную и приватную части. Когда вы отправляете `submission_C.csv`, вам показывается результат на **публичной** части. После завершения этапа результат будет пересчитан на **приватной** части. Публичная и приватная часть не пересекаются.

После окончания этапа ваша метрика будет приведена к 60-балльной шкале по следующему правилу:

- результат **baseline-решения** со значением $\mathrm{F1} \leqslant X$ оценивается в **0 баллов**;
- результат со значением $\mathrm{F1} \geqslant Y$ оценивается в **60 баллов**;
- если значение F1 находится между X и Y, количество баллов вычисляется по формуле линейной интерполяции:

$$\mathrm{Score} = 60 \cdot \frac{\mathrm{F1} - X}{Y - X}.$$

Значения метрик X и Y будут доступны в тестирующей системе.

Итоговые баллы за задачу выставляются по **последней** посылке.

Метрика оценивания точности ответа

В этой задаче используется метрика **F1-score**. Участник отправляет бинарные метки класса (`0` или `1`).

Обозначим:

- `TP` — количество истинно положительных предсказаний;
- `FP` — количество ложно положительных предсказаний;
- `FN` — количество ложно отрицательных предсказаний.

Тогда

$$\mathrm{Precision} = \frac{TP}{TP + FP},$$

$$\mathrm{Recall} = \frac{TP}{TP + FN},$$

а

$$\mathrm{F1} = \frac{2 \cdot \mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}.$$

Значение **F1-score** лежит в диапазоне от 0 до 1. Чем выше значение, тем лучше качество классификации.

Пример расчёта метрики **F1-score** на Python:

```
from sklearn.metrics import f1_score

score = f1_score(y_true, y_pred)
print("F1-score =", score)
```

D. Наследие Человечества

Условие

Извержение вулкана Везувий в 79 году нашей эры засыпало пеплом и уничтожило город Помпеи и многие другие города, в том числе Геркуланум, где в 1750 году была обнаружена древнеримская вилла с большим количеством античных папирусов. Свитки, хранившиеся в библиотеке, обуглились и превратились в хрупкие чёрные цилиндры. В наши дни учёные стремятся прочитать их, используя методы рентгеновской томографии, компьютерной реконструкции и машинного обучения.

Когда Вадим узнал об этом, он подумал, что тоже хотел бы поучаствовать в процессе расшифровки. Узнав, что учёные уже решают задачу восстановления текста внутри каждого отдельного папируса, он понял, что следующий шаг — восстановить правильный порядок самих папирусов, ведь античные трактаты обычно состояли из нескольких частей.

Поскольку Вадим не знает древнегреческого, он решил сначала рассмотреть аналогичную задачу на русском языке. Для этого он взял сборник стихотворений, часть стихотворений оставил неизменной (`train_D.json`). Каждое из оставшихся стихотворений (`test_D.json`) он разделил на две части, а затем перемешал между собой все левые и правые части. Теперь задача Вадима — восстановить исходные пары и собрать стихотворения обратно.

Формат ввода

К задаче прикреплены следующие файлы:

- `model_D.zip` — веса для большой языковой модели.
- `baseline_D.ipynb` — ноутбук с базовым решением задачи и примером использования большой языковой модели `model_D.zip` для получения эмбедингов текста
- `train_D.json` — дополнительные стихотворения целиком;
- `test_D.json` — перемешанные левые и правые части стихотворений;
- `submission_D.csv` — пример решения, которое необходимо отправить в тестирующую систему.

Формат вывода

Для проверки необходимо загрузить архив `solution_D.zip`.

Архив должен содержать:

1. Файл `submission_D.csv` с двумя колонками:
   - `left_id` — ID левой половины из `test_D.json`;
   - `right_id` — ID правой половины из `test_D.json`.
2. Файл `solution_D.ipynb` — Jupyter Notebook с вашим решением.

Допускается добавлять в архив дополнительные файлы, необходимые для работы решения. При этом архив должен содержать ровно один файл с расширением `.csv` и ровно один файл с расширением `.ipynb`.

Система оценивания

За эту задачу можно получить до 60 баллов.

Данные разбиты на публичную и приватную части. Когда вы отправляете `submission_D.csv`, вам показывается результат на **публичной** части. После завершения этапа результат будет пересчитан на **приватной** части. Публичная и приватная часть не пересекаются.

После окончания этапа ваша метрика будет приведена к 60-балльной шкале по следующему правилу:

- результат **baseline-решения** со значением Accuracy $\leqslant X$ оценивается в **0 баллов**;
- результат со значением Accuracy $\geqslant Y$ оценивается в **60 баллов**;
- если значение Accuracy находится между X и Y, количество баллов вычисляется по формуле линейной интерполяции:

$$\text{Score} = 60 \cdot \frac{\text{Accuracy} - X}{Y - X}.$$

Значения метрик X и Y будут доступны в тестирующей системе.

Итоговые баллы за задачу выставляются по **последней** посылке.

Метрика оценивания точности ответа

В этой задаче оценивается корректность сопоставления левых и правых частей стихотворений.

Пусть:

- N — общее количество левых частей в `test.json`;
- K — количество правильно восстановленных пар, то есть таких пар `(left_id, right_id)`, которые совпадают с истинным соответствием.

Тогда метрика вычисляется по формуле

$$\text{Accuracy} = \frac{K}{N}.$$

Иными словами, метрика показывает долю правильно восстановленных стихотворений.

Значение **Accuracy** лежит в диапазоне от 0 до 1. Чем выше значение, тем лучше качество решения.

Пример расчёта **Accuracy** на `Python`:

```python
import pandas as pd

merged = true_pairs.merge(pred_pairs, on="left_id", suffixes=("_true",
    "_pred"))
correct = (merged["right_id_true"] == merged["right_id_pred"]).sum()
accuracy = correct / len(merged)

print("Accuracy =", accuracy)
```

E. Подозрительные пирожные

Условие

Саша часто болеет и пытается понять причину. Он подозревает, что проблема может быть связана с пирожными, которые он регулярно ест.

Недавно Саша купил новую партию пирожных, и среди них могли оказаться необычные, которые для него вредны. Саша очень хочет съесть все пирожные, но не хочет снова заболеть. Для этого Саше нужно научиться определять, какие пирожные отличаются от остальных.

Саша уверен, что подозрительные пирожные можно отличить визуально. Чтобы научиться их находить, он решил обучить свёрточную нейросеть анализировать изображения пирожных. Однако для обучения у Саши есть только «нормальные» пирожные, про которые известно, что они ему точно не вредны. Всего Саша ест 10 видов пирожных, поэтому он обучил нейросеть классифицировать «нормальные» пирожные на 10 классов. Теперь Саше нужно придумать, как с помощью этой модели находить «вредные» пирожные. Саша просит вашей помощи, чтобы найти подозрительные пирожные среди новой партии.

Вам даны изображения пирожных из новой партии и веса предобученной нейросети. Нейросеть обучена только на «нормальных» пирожных и решает задачу классификации на 10 классов. Вам нужно определить, какие пирожные из новой партии отличаются от «нормальных». Известно, что подозрительных пирожных ровно 1000.

Формат ввода

К задаче прикреплены файлы:

- `public_test_package_E.npz` — тестовые изображения размера 32×32;
- `model_weights_E.pt` — веса предобученной CNN;
- `baseline_E.ipynb` — базовый ноутбук с примером решения;
- `submission_E.csv` — пример файла ответа.

Формат вывода

Для проверки необходимо загрузить архив `solution_E.zip`.

Архив должен содержать:

1. Файл `submission_E.csv` с двумя колонками:
   - `id` — индекс объекта от 0 до $N - 1$;
   - `is_outlier` — ваш прогноз:
     - `1`, если объект является подозрительным;
     - `0`, если объект считается нормальным.
2. Файл `solution_E.ipynb` — Jupyter Notebook с вашим решением.

В файле `submission_E.csv` должно быть ровно 1000 строк со значением `is_outlier = 1`.

Допускается добавлять в архив дополнительные файлы, необходимые для работы решения. При этом архив должен содержать ровно один файл с расширением `.csv` и ровно один файл с расширением `.ipynb`.

Система оценивания

За эту задачу можно получить до 60 баллов.

Если в отправленном файле количество объектов, помеченных как подозрительные, **отличается от** 1000**, то за задачу выставляется** 0 **баллов**.

Данные разбиты на публичную и приватную части. Когда вы отправляете `submission_E.csv`, вам показывается результат на **публичной** части. После завершения этапа результат будет пересчитан на **приватной** части. Публичная и приватная часть не пересекаются.

После окончания этапа значение метрики будет приведено к 60-балльной шкале по следующему правилу:

- результат **baseline-решения** со значением Recall@K $\leqslant X$ оценивается в **0 баллов**;
- результат со значением Recall@K $\geqslant Y$ оценивается в **60 баллов**;
- если значение Recall@K находится между X и Y, количество баллов вычисляется по формуле линейной интерполяции:

$$\text{Score} = 60 \cdot \frac{\text{Recall@K} - X}{Y - X}.$$

Значения метрик X и Y будут доступны в тестирующей системе.

Итоговые баллы за задачу выставляются по **последней** посылке.

Метрика оценивания точности ответа

В этой задаче используется метрика **Recall@K**.

Участник должен отметить ровно $K = 1000$ объектов как подозрительные. Пусть hits@K — число правильно найденных подозрительных объектов среди отмеченных. Тогда

$$\text{Recall@K} = \frac{\text{hits@K}}{K}.$$

Иными словами, чем больше настоящих подозрительных объектов вы найдёте среди выбранных 1000, тем выше результат.