Курс "Основы Data Science": тестовое задание

Добро пожаловать!

Data Science - необычайно интересная область знаний, состоящая из трёх основных компонентов: **алгоритмы**, **инженерия (программирование)**, и **понимание бизнеса**.

Данный тест призван оценить ваши навыки по этим направлениям.

Несколько моментов про выполнение задания:

- несмотря на то, что многие псевдофункции и определения приведены на русском языке, убедительная просьба в самом коде использовать исключительно английский язык;

- если Вы хотите применить какую-либо стороннюю библиотеку при выполнении заданий - не вопрос, только укажите, пожалуйста, комментарии в строке импорта, какую версию библиотеки Вы используете;

- Вы можете выполнять это задание прямо в этой тетради, либо же (если Вы пользуетесь PDF версией) Вы можете создать новую, и просто указывать, к какому пункту относится ваше текущее решение;

- Если Вы знакомы с Jupyter Notebook - отлично, значит, проблем вообще не должно возникнуть. Если же нет - то Вы можете воспользоваться Google Colabs, mybinder.org или любым другим сервисом, который позволяет запускать Jupyter Notebook онлайн. В самом крайнем случае (ну вот если совсем не получается с Jupyter Notebook) Вы можете сделать задание в виде скрипта/скриптов .py;

И последнее: с одной стороны, мы живём в век Google, с другой - невозможно запомнить абсолютно всё, так что Вы можете спокойно пользоваться сторонними материалами при выполнении этого теста. Главное и обязательное требование при этом - чтобы Вы понимали свой ответ и могли его объяснить.

УДАЧИ!

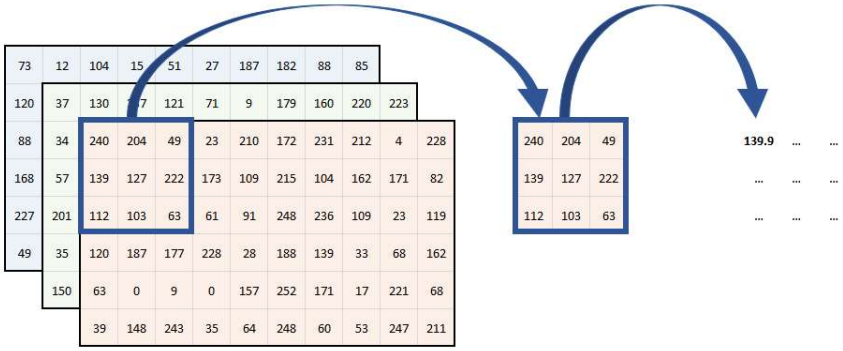
1. Python и ООП

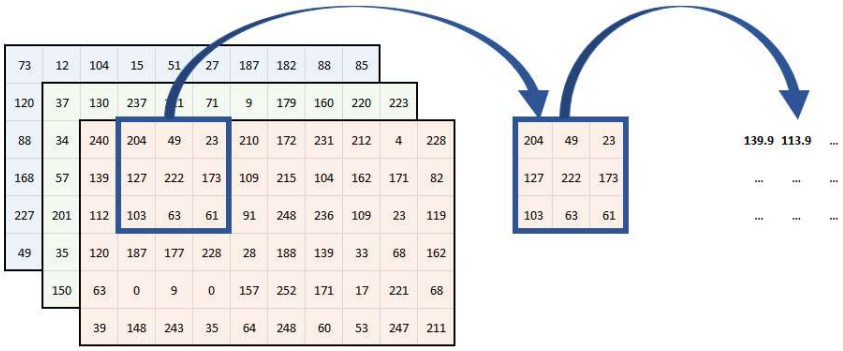
1.1 Конволюции

Вам дана трёхмерная матрица размера **3**x**M**x**N**.

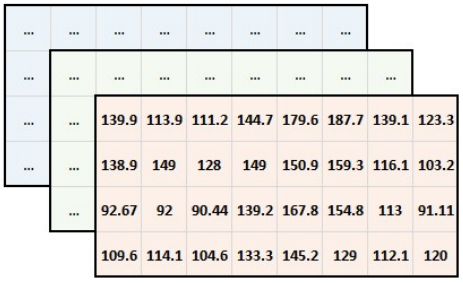
Ваша задача - написать функцию, которая сканирует эту матрицу при помощи скользящего окна заданного размера (в данном примере пусть размер будет жёстко задан, **3**x**3**, как на изображении) и вычисляет среднее значение элементов в данном окне.

Функция начинает с "верхнего левого угла" и затем "скользит" вправо, пока не достигнет границы. Затем окно смещается на одну строку вниз и сканирует её подобным образом. Каждый из трёх слоёв матрицы обрабатывается независимо от других.





В результате ваша функция должна вернуть матрицу размером **3**x**(M-2)**x**(N-2)**



В качестве примера для ввода можете использовать приведённую выше матрицу:

```
matrix = [[[240, 204,  49,  23, 210, 172, 231, 212,   4, 228],
           [139, 127, 222, 173, 109, 215, 104, 162, 171,  82],
           [112, 103,  63,  61,  91, 248, 236, 109,  23, 119],
           [120, 187, 177, 228,  28, 188, 139,  33,  68, 162],
           [ 63,   0,   9,   0, 157, 252, 171,  17, 221,  68],
           [ 39, 148, 243,  35,  64, 248,  60,  53, 247, 211]],

          [[ 37, 130, 237, 121,  71,   9, 179, 160, 220, 223],
           [ 34,  52, 209, 179, 102, 224, 163, 167,  54,  53],
           [ 57, 138, 192, 132,  74, 124, 170,  64,  17, 247],
           [201, 113, 106, 175,  15, 223, 212,  14, 133,  53],
           [ 35, 250, 183,  28,  72, 223, 202, 162, 252,   2],
           [150, 248,  29,   5,   7,  48, 209, 239, 183, 119]],

          [[ 73,  12, 104,  15,  51,  27, 187, 182,  88,  85],
           [120, 206,   3, 126, 200, 201, 153,  28,   9, 100],
           [ 88, 142,  14, 179, 193, 244,  30, 251,   2,  86],
           [168, 174,  30, 102,  24,  45, 174, 165,  90,  67],
           [227,  94, 130, 131,  56, 108, 235, 213,   9,  41],
           [ 49,  89,  16, 190, 207, 136, 176,  58, 153,  57]]]
```

1.2 Необитаемый остров

Следующее задание должно показать ваш уровень понимания и владения базовыми понятиями ООП, а именно: классы и наследование

Рядом с необитаемым островом затонул корабль с N амишами в возрасте от 20 до 70 лет.
Хорошая новость: все выжили и остров полон ресурсов. Плохая - связи с внешним миром нет, и поэтому пассажиры решили основать своё островное государство-общину.

1.2.1 Модель социума

Первая задача - **постройте набор классов для описания островитян**.
Типы: мужчина, женщина, девочка, мальчик (все, кому меньше 18 лет - дети).
Всем им присущи:

- имя
- фамилия
- дата рождения
- цвет глаз
- цвет волос.

Только взрослые могут состоять в браке
Только женщины могут быть беременны

Действия, которые люди могут совершать:

- Все могут идти, есть, спать
- Только дети могут играть на площадке
- Только Взрослые могут работать в поле
- Только мужчины (взрослые и мальчики) могут ходить на охоту (без сексизма, просто для примера)

Применение наследования крайне приветствуется!

1.2.2 Социум в динамике

Вторая задача - симулировать развитие нашего острова, а именно - прирост его населения.
Законы государства в области брака просты:

- в брак можно вступать с 18 лет
- браки моногамные
- нет однополых браков
- никто не разводится
- брак между ближайшими родственниками (брат-сестра) запрещены
- раз в месяц все незамужние жители встречаются на центральной площади и пытаются найти себе пару

При встрече незамужних мужчины и женщины, базовая вероятность, что они поженятся = 0.25.
Если у пары одинаковый цвет глаз / волос, то шанс возрастает до 0.35.
Дети рождаются только у замужних пар. Максимальный возраст деторождения - 50 лет (для женщин и мужчин).
Вероятность зачатия - 0.3 (каждый месяц).
Беременность - 9 месяцев (неожиданно).
Средняя продолжительность жизни на острове - 80 лет, со среднеквадратическим отклонением - 8.3 года.
На острове идеальный порядок, все законопослушные и нет несчастных случаев, так что все смерти - естественные.
Ваше задание: напишите функцию / функции / классы / ..., которые позволят симулировать состояние общества в конце каждого месяца.

Пример: в начале месяца на острове было 500 мужчин и 450 женщин, 150 мальчиков и 160 девочек с определёнными параметрами. Из взрослых 225 женщин и мужчин состояли в браке, из этих 225 женщин 42 - на том или ином месяце беременности.
В конце месяца состояние было следующее: 2 мужчины и 3 женщины покинули этот мир, из 42 беременностей 7 закончились (3 девочки и 4 мальчика), и т.д. и т.п.

Пример псевдокода механизма знакомства:

```
замужние женщины = [...]
женатые мужчины = [...]

для каждого мужчины из всех неженатых мужчин:
    для каждой женщины из всех незамужних женщин:
        вероятность того, что они понравились друг другу = ...
        если эта вероятность выше базовой вероятности (с учётом вышеуказанного фактора) - они поженились
            добавляем их в список замужних / женатых
            убираем их из списка незамужних / неженатых
```

Это задание достаточно сложное!
Советую сперва определиться с архитектурой ваших классов и функций, и уже после этого приступать к эмуляции. Ниже приведён один из примеров, как Вы можете сымитировать знакомство (на псевдокоде)

1.3 Работа с исключениями и логами

Вы работаете в большой команде. От ваших коллег Вы получили несколько функций, которые они разработали, и которые Вы должны имплементировать в свой код.
Проблема в том, что эти функции иногда "падают".
После подробного изучения, Вы обнаружили, что Вы можете столкнуться со следующими исключениями:

- *ValueError*
- *TypeError*
- *KeyError*
- *ZeroDivisionError*

Вы **не можете** изменить код самих функций, которые Вы получили. Так что ваша задача - разработать механизм и логику, как Вы будете на возможные исключения.

Для себя Вы определили, что если происходит *ValueError*, *ZeroDivisionError* или *KeyError*, то в целом вы можете продолжать выполнение вашего кода (после каких-либо манипуляций, особо их указывать не надо).
Но если Вы получаете *TypeError*, то делать уже нечего и операцию надо прекращать.

Также Вы решили, что в любом случае хотите сохранить в логах подробности о произошедшем исключении.

Ваша задача - модифицировать её для достижения поставленных целей

Внизу Вы можете увидеть пример псевдокода вашей функции до внедрения отработки ошибок:

```
import чужая_забагованная_функция_1  # может выкинуть `ValueError`, `KeyError`

import чужая_забагованная_функция_2  # может выкинуть `TypeError`

import чужая_забагованная_функция_3  # может выкинуть `ZeroDivisionError`


def моя_прекрасная_функция(x: int, y: int) -> int:

    z_1 = чужая_забагованная_функция_1(x)

    z_2 = чужая_забагованная_функция_2(x)

    для i в диапазоне(1, z_1):

        z_2 += чужая_забагованная_функция_3(y) / z_1

    return z_2
```

2. Математика и логика

2.1 Статистика случайной величины

Напишите функцию, которая на входе получает лист случайных значений, и возвращает её математическое ожидание и среднеквадратическое отклонение.

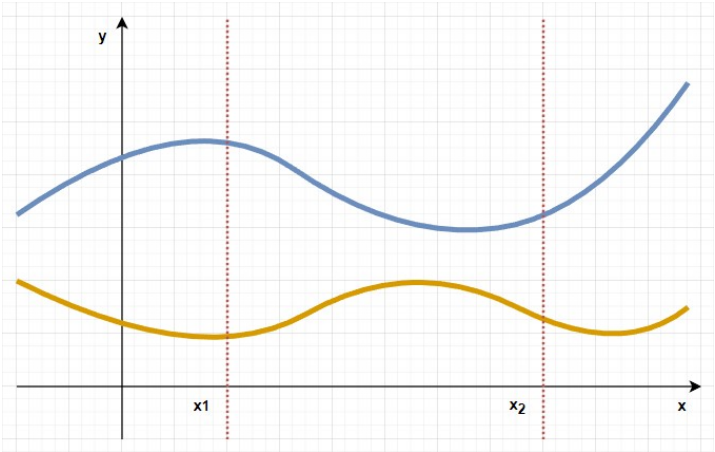
Псевдокод ожидаемой функции:

```
from typing import List

def calc_statistics(values: List[int]) -> ...:

    ...
```

2.2 Площадь между кривыми двух функций



Напишите функцию, которая на входе получает:

- функцию верхней (синей) линии (принимает на вход значение **X** и на выходе отдаёт значение этой функции **Yu** в заданной точке)
- функцию нижней (оранжевой) линии (принимает на вход значение **X** и на выходе отдаёт значение этой функции **Yb** в заданной точке)
- Границы интервала **X1** и **X2**

Функция должна вернуть приблизительную площадь фигуры, ограниченной заданными кривыми на данном интервале

Пример определения функции кривой:

```
from typing import Callable

def plot_function(x: float) -> float:

    ...
```

Пример определения функции интеграла:

```
from typing import Callable

def calculate_integral(func_a: Callable, func_b: Callable, x_1: float, x_2: float) -> float:
    ...
```

Особо подчеркну: функция интеграла принимает в качестве параметров две другие функции

2.3 Вероятность и математическое ожидание

Вы играете со своим другом в настольную игру. В вашем распоряжении два кубика:

- первый - с 16 равнозначными гранями и значениями от 1 до 16
- второй - с 7 "смещёнными" гранями со значениями 10, 20, 30 ... 70. Вероятность выпадения каждой из них определяется по формуле:

P(val) = (80 - val) / 280

(то есть, например, вероятность того, что выпадет 50, равна (80 - 50) / (280) = 0.107)

Каждый раунд один игрок бросает оба кубика и записывает полученный результат. Второй игрок должен выбрать: либо он тоже бросает оба кубика, либо один из них от одного до трёх раз (сколько именно, игрок может решать по результатам предыдущих бросков).

Цель - получить максимально близкое значению к тому, что получил ваш оппонент.

Ваша задача - написать функцию, которая поможет Вам выбрать подходящую стратегию (сколько кубиков бросать) исходя из полученного вашим оппонентом результата

Маленький совет: не ограничивайте себя рамками формул, будьте изобретательны)

3. Вопросы на бизнес-логику

Приведённые ниже вопросы основаны на реальных кейсах. В данном блоке нет правильных / неправильных ответов, ваш ответ должен продемонстрировать ваш анализ и рассуждения.

3.1 У нас есть данные!

Вы специалист по Data Science, приглашены на встречу с новым клиентом.
После вступительной части, он Вам говорит:
" Наша IT система работает безостановочно вот уже три года, и мы накопили просто огромное количество данных. Теперь мы на их основе разработать классные ML-модели! И будет просто супер, если там будут нейросети! Сейчас все используют их, они меняют мир! "

Вопросы:

Как бы вы продолжили этот разговор?
Какие вопросы Вы хотели бы задать?
На чём сосредоточить своё внимание?

3.2 Провинциальный университет

Ректор одного из провинциальных университетов решил провести исследование, как результат ЕГЭ и средний балл аттестата взаимосвязаны с успеваемостью студентов.
Собрав достаточно данных и проведя анализ, он с удивлением обнаружил, что существует обратная связь между баллом по ЕГЭ и средним баллом по экзаменам в университете (коэффициент корреляции = -0.4). При этом средняя оценка по школьному аттестату, как и предполагалось, имеет положительную связь с успеваемостью.

Вопрос:

Как вы думаете, как можно объяснить подобные результаты? Все ли факторы учёл ректор при проведении исследования?

УДАЧИ!