1. Описание

Общий план действий:

В нашем проекте мы решили взять сайт Ломоносовской олимпиады школьников (ЛОМа), где содержатся результаты участников и распределение их «статусов» (победитель/призёр): сайт олимпиады. Интерес школьников к олимпиадам не угасает с каждым годом, так как они позволяют поступить на бюджет в лучшие университеты России без вступительных экзаменов. При этом «внутренние» правила соревнований не остаются неизменными: каждый олимпиадный сезон ЛОМ меняет количество призёров и победителей, проходные баллы на заключительный этап. Мы решили изучить эту динамику на доступных нам данных, чтобы абитуриенты могли более точно рассчитать вероятность их «победы» и решить, сосредотачиваться ли им только на олимпиадах или уделять больше времени ЕГЭ. Более того, отборочный и заключительный этап различаются по уровню сложности в разных «пропорциях» каждый год, поэтому мы также хотим посмотреть, какое количество победителей «отборочного этапа» становятся победителям заключительного этапа. Наш проект может помочь не только школьникам, но и: 1) организаторам ЛОМа сделать выводы о том, насколько их задания соответствуют общему уровню подготовки олимпиадников; 2) государственным образовательным органам сделать вывод о «степени» олимпиады (все олимпиады по уровню сложности и, соответственно, привилегиям, которые могут принести, делятся на первую, вторую и третью степень).

Сбор данных и описание модулей:

В качестве данных мы взяли результаты отборочного и заключительного этапов по биологии школьников 10-11 классов за 2022 год.

1) С помощью библиотеки *request* можно направлять сообщения к серверу, чтобы получить от него реакцию, используя несложный синтаксис.

Библиотеки этого модуля часто привлекаются при парсинге, так как препятствуют влиянию посторонних служб, работа кода не замедляется.

2) Важнейшая библиотека при парсинге – *Beautiful Soup*, которая используется при работе с HTML документами, потому что преобразует адрес в объекты, с которыми программисту уже удобно работать.

Интерпретация собственных данных:

1) Распределение баллов на отборочном этапе ЛОМа по биологии

Ожидаемо, чем выше балл, тем меньше количество участников, набравших такой результат. По графику видно, что почти все участники набирают 53 балла, поэтому дифференциация «знаний» начинается именно после этого числа. Самый большой скачок в количестве людей, набравших определенных балл, наблюдается примерно между 72 и 75 баллами (68-72 балла набрали примерно 65 человек, а 72-76 – уже 40 человек) и между 76 баллов и 78 баллов (76-78 баллов набрали примерно 40 человек, а 78-83 баллов – 19 человек). Можно предположить, что, набрав более 72 баллов, можно уже претендовать на призёра/победителя, так как такого результата достигает более чем в 3 раза меньше участников, по сравнению с набравшими от 50 до 54 баллов. Больше 78 баллов набирает меньше 25 участников отборочного этапа, поэтому в этом случае вероятность прохода на заключительный этап велика (при общем количестве участников более 600 человек).

2) Распределение баллов на заключительном этапе ЛОМа по биологии

По графику видно, что 57-65 баллов набирает большее количество участников (более 90), чем в других категориях, разбитых по баллам. Здесь уже нет обратной зависимости балла и количества участников, получивших такой результат. Это можно объяснить тем, что на заключительный этап попадают школьники, посвятившие какое-то время подготовке и рассчитывающие на победу, нежели просто участие. Так, школьников,

набравших до 42-43 баллов, примерно 83 в сумме. Самый резкий скачок между категориями происходит между участниками, набравшими 65-72 балла (77 человек) и набравшими 72-81 балл (уже 38 человек). Вероятно, в районе этого предела и будет точка диверсификации статусов. При этом вряд ли БВИ по этой олимпиаде будут получены только теми участниками, кто набрал более 80 баллов, потому что таких школьников меньше 20.

3) Распределение статусов «победитель» и «призёр» на отборочном этапе ЛОМа по биологии

Можно заметить, что статусов «победитель» на отборочном этапе около 20, тогда как статуса «призёр» более 600. Вспоминая диаграмму 1, можно предположить, что пограничным баллом для становления «победителем» был как раз 78, так как такой и выше результат получили примерно 25 человек в целом.

4) Взаимосвязь результатов на отборочном и заключительном этапах ЛОМа

Рассматривая категорию участников, набравших в первом этапе от 50 до 70 баллов, мы можем отметить, что подавляющее большинство получило результат в таких же пределах на заключительном этапе. Участники, набравшие в отборочном этапе от 30 до 50 баллов, в среднем, улучшили свой результат (нет ни одного человека, набравшего на заключительном этапе менее 50 баллов), однако большинство все равно не превысило 60 баллов. Интересно, что среди этой категории есть пять «отличающихся» наблюдений: участники, набравшие на втором этапе более 75 баллов. Это близко к «точке диверсификации», о которой писалось ранее. Напомню, что выше мы сделали следующие выводы по заключительному и отборочному этапам соответственно: 1) статус «победителя» начинается примерно с 80 баллов; 2) статус «победителя» начинается примерно с 78 баллов. По графику взаимосвязи видно, что из людей, набравших на отборочном этапе более 78 баллов, только 2 человека стали «победителями» и на

заключительном этапе. Большинство школьников, набравших более 80 баллов на втором этапе, изначально имели результат от 40 до 70 баллов. Можно сделать вывод, что «победитель» на отборе совсем не значит, что с высокой вероятностью можно стать «победителем» на заключительном этапе.

Трудности:

В проекте у нас присутствовали данные разных форматов. В данных отборочного этапа присутствуют только номер заявки, статус участника (призёр/победитель) и средний балл. В данных заключительного этапа, помимо этого, были также пол, возраст. Для того, чтобы склеить таблицы в Excel, нам пришлось пренебречь некоторыми социальными характеристиками участников, так как в этом проекте мы ориентировались именно на средний балл.

Предложение на будущее:

Наш проект собирает данные только по введённому предмету за один год. Можно было бы посмотреть изменение среднего проходного балла по дисциплине за несколько лет. Также можно было бы сравнить средний проходной балл и распределение результатов между разными предметами, чтобы понять, какая дисциплина даётся, в среднем, легче/сложнее. Или же сравнить количество участников по каждому предмету, чтобы понять, какой из них популярнее среди школьников (это также поможет понять востребованность университетских образовательных программ среди молодёжи). Более того, так как в данных заключительного этапа имеется информация о половозрастном составе участников, можно было бы проверить, действительно ли одиннадцатиклассники, в среднем, «обгоняют» десятиклассников или подобные олимпиады не коррелируют с возрастом. Также можно было бы проворить теорию «стеклянного» потолка для участников женского пола.

2. Документация

Описание функций, которые мы используем в коде

1) os.makedirs("olymplom\zakl")
   os.makedirs("olymplom\otbor")

Модуль os, откуда мы импортируем функции выше, включает в себя функции для работы с операционной системой. Мы используем os.makedirs() для того, чтобы создать директорию, где создадутся папки отдельно для отборочного и заключительного этапов (в качестве аргумента – названия папок). Папки появляются в той же директории, где лежит сам код. В папках создадутся csv файлы с данными олимпиады по выбранному предмету.

2) session = requests.Session()

   retry = Retry(connect=3, backoff_factor=0.5)

   adapter = HTTPAdapter(max_retries=retry)

   session.mount('http://', adapter)

   session.mount('https://', adapter)

Мы используем функцию Session из библиотеки request для повышения производительности запросов и для их детализации. Так, мы сможем обойти блокировку со стороны сайта ЛОМа и подключиться к нему несколько раз с небольшими паузами. Также мы используем HTTPAdapter, чтобы установить, сколько раз мы хотим повторить запрос к определенному API. Адаптер HTTP протокола позволит в session передать считывание сайтов по данном протоколу, настраиваем протокол по session.mount (сайты могут быть как http, так и https), этому методу скармливаем заданный параметр.

В качестве аргумента к функции HTTPAdapter мы используем аргумент retry. Эта переменная получается с помощью Retry из модуля urllib3.util.retry: она обращается к сайту еще три раза, если до этого она завершилась неудачно. У неё первый аргумент – количество обращений, второй - экспоненциальное увеличение перерыва между попытками захода.

3) soup = BeautifulSoup(src,'lxml')
   table_head = soup.find(class_="table").find("tr").find_all('th')

Из модуля bs4 мы импортируем класс BeautifulSoup. С помощью функции BeautifulSoup мы преобразуем HTML-данные в нужный нам структурированный формат (lxml). В переменной src лежит текст сайта ЛОМа.

С помощью метода soup.find() мы извлекаем первый элемент документа, который отфильтрован нашим аргументом (то есть в помощью регулярного выражения или сочетания букв). Find.all() ищет все выражения, соответственно. Конкретно эта строчка нам нужна, чтобы найти теги в таблице, отвечающие за заголовки.

4) writer = csv.writer(file)
   writer.writerow((name, point, status))
   Мы используем модуль csv, чтобы записать результаты парсинга в таблицу в Excel, которая будет удобна позже для выведения статистик. Так как данных много, их неэффективно хранить просто в списке в Питоне, поэтому функции этого модуля будут возвращать информацию из таблицы с разделителями строк для дальнейшего преобразования. С помощью неё мы уже будем строить графики. Во второй строчке создаем таблицу со столбцами "имя, балл, статус".
5) product_tds, name1, point1, status1 = assign(item)
   dfotbor = pd.read_csv('olymplom\otbor\lom_otbor_Биология.csv')
   result = pd.merge(dfotbor, dfzakl, on="Биология")

Assign – метод из модуля Pandas, который добавляет участника в таблицу с указанными параметрами (имя, балл, статус). Мы импортируем модуль pandas для работы с данными, которые были получены с помощью парсинга с сайта ЛОМа и выгружены в csv таблицы. Pandas как раз использует данные, которые уже структурированы, а не разбросаны хаотично. Во второй строчке мы как раз открываем csv документ, куда сохранили данные с сайта. В третьей строчке мы склеиваем столбцы вместе.

6) plt.hist(dfotbor['Балл'], color='c')

plt.title('Распределение баллов на отборочном этапе олимпиады Ломоносов, Биология')

plt.xlabel('Балл')

plt.ylabel('Количество')

Мы импортируем библиотеку Matplotlib для визуализации полученных статистик (графики распределения баллов, статусов на этапах олимпиады и взаимосвязь баллов на отборочном этапе и баллов на заключительном этапе).

Это функции из matplotlib: строим гистограмму с помощью первой строчки. Вторая строчка задает заголовок, третья и четвертая – названия осей икс и игрек.

Описание наших функций

Функция get_lom_otbor принимает на вход необходимую ссылку. Чтобы избежать блокировки, обходит блок парсера со стороны сайта, использует функции requests.Session(), retry, HTTPAdapter и mount.

Далее получает lxml код с сайта при помощи BeautifulSoup. В классе таблиц ищет необходимые места в коде, чтобы получить шапку таблицы и далее поиндексно делит полученную шапку на имя школьника, балл и статус

Кроме имени, балла и статуса функция возвращает и саму таблицу

Функция get_lom_data_zakl работает похожим образом, но парсит данные не отборочного, а заключительного этапа, которые имеют другую структуру. В данных заключительном этапа не указан статус, поэтому функция возвращает только имя, балл и саму таблицу.

Функции assign и assignzakl создают объекты, которым присваиваются имя, балл и статус участника.