**Доповідь**

**Слайд 1:**

Pandas vs NumPy у світі аналізу даних

Підготували ФБ-21 Редько-Шпак Родислав та ФБ-25 Слобода Ірина

**Слайд 2:**

Сьогодні ми обговоримо дві важливі бібліотеки для аналізу даних у Python: Pandas і NumPy. Аналіз даних є невід'ємною частиною сучасного світу, допомагаючи приймати обґрунтовані рішення на основі великих обсягів інформації. Pandas і NumPy є двома найпоширенішими інструментами для роботи з даними, кожен з яких має свої переваги та недоліки.

**Слайд 3:**

Метою цієї презентації є:

* Порівняти Pandas і NumPy у контексті аналізу даних, висвітлюючи їхні сильні та слабкі сторони. У епоху великих даних, правильний вибір інструментів для обробки та аналізу є критично важливим для отримання максимальної цінності з наявних даних.
* Надати реальні приклади використання обох бібліотек. Презентація буде багата на код та практичні демонстрації, які покажуть, як працювати з Pandas та NumPy в реальних ситуаціях аналізу даних.
* Визначити, коли слід використовувати одну бібліотеку (інструмент), а коли іншу.

**Слайд 4:**

Наша доповідь буде побудована із трьох частин.

**Слайд 5:**

Перша: Загальний огляд Pandas та Numpy в Python

**Слайд 6:**

Почнемо з того, що таке Python?

Python - це високорівнева об'єктно-орієнтована мова програмування, яка має дуже простий синтаксис, що сприяє її легкому сприйняттю. Оскільки в її основі лежить проста англійська, Python є однією з найпростіших для вивчення мов програмування. Також Python - одна з найпопулярніших мов для задач машинного навчання та аналізу даних. Вона потужна завдяки своїм бібліотекам, які надають користувачеві повний контроль над даними.

**Слайд 7:**

Що таке Python бібліотека?

У програмуванні "бібліотека" - це набір коду, що складається з десятків або навіть сотень модулів, які надають різноманітні функціональні можливості. Кожна бібліотека містить заздалегідь скомбіновані коди, використання яких допомагає заощадити час розробників. Бібліотеки особливо цінні, оскільки надають доступ до попередньо написаного коду, який можна неодноразово використовувати у різних проектах. Завдяки цьому програмісти уникають ситуацій, коли їм доводиться створювати однакові рішення з нуля для кожної нової задачі.

**Слайд 8:**

NumPy - це потужна бібліотека Python з відкритим вихідним кодом, яка активно використовується протягом останніх кількох років. На офіційному сайті зазначено, що NumPy є "фундаментальним пакетом для наукових обчислень на Python". За допомогою NumPy можна легко виконувати операції над великими багатовимірними масивами та матрицями. Більше того, NumPy також надає нам величезний набір високорівневих математичних функцій, наприклад, sin(), sort() і т.д. для роботи з цими масивами та їх елементами.

**Слайд 9:**

В основі Numpy лежить клас **ndarray** - багатовимірний контейнер елементів одного типу даних і розміру. Об'єкти цього класу займають менше пам'яті, а операції над ними виконуються швидше, ніж зі звичайними списками Python. Менша витрата пам'яті забезпечується за рахунок завжди точно заданого розміру, оскільки всі елементи мають один тип (наприклад, int32 або float32). Швидкість пояснюється уникненням пітонівських циклів. NumPy використовує низькорівневу мову програмування С, а багатовимірний об'єкт ndarray розгортається в одновимірний масив.

Список (list) у Python "під капотом" містить масив покажчиків на об'єкти. Тому вилучення елементів списку передбачає звернення до об'єкта та перевірку типів. Проведемо побутову аналогію: уявіть, що вам потрібно перетягнути купу коробок, які знаходяться за 20 метрів від вас. При цьому потрібно заглядати в кожну коробку, щоб перевірити її вміст. З точки зору мови С це виглядало б так: коробки знаходяться прямо перед вами, заглядати всередину не потрібно, до того ж ви завжди знаєте точну їх кількість і масу кожної коробки. Саме тому масиви NumPy такі ефективні, але накладають певні обмеження.

**Слайд 10:**

Для демонстрації функціонала бібліотеки Numpy створимо масив (трьохвимірний) за допомогою ф-ії nparray. Як ми бачимо, синтаксис дуже простий.

**Слайд 11:**

Ця картинка ілюструє концепцію багатовимірних масивів у NumPy та показує різницю між одновимірними (1D), двовимірними (2D) та тривимірними (3D) масивами.

**Слайд 12:**

Pandas - одна з найпопулярніших програмних бібліотек Python, яку можна використовувати для маніпулювання та аналізу даних, оскільки вона надає розширені структури даних для зберігання різних типів маркованих та реляційних даних, а також дозволяє виконувати багато операцій. Pandas був розроблений Весом МакКінні у 2008 році. Він був побудований на основі пакету NumPy з Python (Pandas не може бути використаний без використання NumPy).

**Слайд 13:**

На відміну від NumPy, бібліотека Pandas не вимагає знань лінійної алгебри або вмінь працювати з багатовимірними масивами. DataFrame можна уявити як таблицю, де кожен рядок представляє одиничне спостереження. А з рядками і стовпцями зручно працювати. Python-бібліотека Pandas підтримує такі самі операції, як і NumPy: злиття, приєднання, зміна форми, індексація масивів, статистичні дослідження, роботу з порожніми (Nan) значеннями, зміну форми масивів, витяг окремих елементів, конкатенацію. Крім цього, Pandas також дає змогу: читання і запис даних, наприклад, csv або excel-таблиця; групування даних; засоби візуалізації даних; роботу з датами; роботу з рядковими значеннями тощо.

Існує кілька мов, які використовуються для написання Pandas, включаючи Python, Cython і C. Pandas підтримує імпорт даних з декількох форматів файлів, включаючи SQL, JSON, Microsoft Excel тощо.

**Слайд 14:**

Для демонстрації функціонала бібліотеки Pandas створимо датафрейм, в якому є три стовбці - Ім’я, Оцінки та Стать. Як ми бачимо, синтаксис дуже простий і одразу видно, що дана бібліотека створена для взаємодії з табличними типами даних.

**Слайд 15**

**Слайд 16**

**Слайд 17:**

Проведемо порівняння створення масиву та ініціалізації даних. Синтаксис схожий для сіріес та нампай. Коли ми хочемо створити пандас датафрей, то вказуємо ще й назви стовпців.

Отже, NumPy є більш ефективним для створення простих масивів числових даних, тоді як Pandas зручніший для ініціалізації табличних даних з індексами та назвами стовпців.

**Слайд 18:**

**Знову ж таки, бачимо, що синтаксис схожий для нампай да пандас сіріес, окрім пандас датафрейм. Там вибірка береться за стовпчиками**

а) тобто NumPy має перевагу в продуктивності для векторизованих операцій над масивами числових даних. Pandas також підтримує векторизовані операції, але може бути повільнішим для великих наборів даних.

б)В свою чергу Pandas забезпечує більш гнучку та зручну індексацію за мітками (labels), що є перевагою для роботи з табличними даними. NumPy підтримує лише індексацію за позиціями.

**Слайд 19**

Далі демонструємо, що Pandas має вбудовані інструменти для зручної роботи з пропущеними значеннями у табличних даних, що є важливим аспектом аналізу даних.

**Слайд 20**

Pandas надає зручні методи для маніпулювання рядками та стовпцями у табличних даних, включаючи додавання, видалення, застосування функцій тощо.

Код показує можливість застосування фугкції до кожного рядка датафрейма

Pandas надає потужні інструменти для злиття та об'єднання різних наборів табличних даних, що є важливим етапом в аналізі даних.

В другому коді продемонстрована можливість злиття двох датафреймів

**Слайд 21**

Pandas дозволяє легко групувати дані за одним або декількома стовпцями та застосовувати різні агрегуючі функції, такі як сума, середнє, максимум тощо.

Також має вбудовану підтримку часових рядів, що робить її ідеальним вибором для аналізу часових даних, таких як фінансові дані, метеорологічні дані тощо.

Цей код якраз таки показує можливість групування та роботи з часовим рядом

**Слайд 22:**

**На картинках**

**Слайд 23:**

\*показую код\*

Отже, NumPy є кращим вибором для швидких чисельних обчислень над масивами, тоді як Pandas забезпечує більш зручні та гнучкі інструменти для роботи з табличними даними та аналізу даних.

**Слайд 24:**

У цьому прикладі ми створюємо великий DataFrame Pandas з 10 мільйонами рядків і трьома стовпцями. Потім ми:

* Фільтруємо DataFrame, залишаючи лише ті рядки, де значення стовпця 'A' більше 50. Pandas має вбудовану ефективну реалізацію цієї операції, тоді як для NumPy ми повинні явно перетворити DataFrame на масив і виконати фільтрацію з використанням маски.
* Групуємо дані за стовпцем 'A' і обчислюємо середнє значення для кожної групи у стовпці 'B'. Для Pandas це робиться однією командою df.groupby('A')['B'].mean(), тоді як для NumPy ми повинні явно знайти унікальні значення в 'A', розбити масив на групи за цими значеннями та обчислити середнє для кожної групи.

**Як ви можете бачити, для операцій фільтрації та групування/агрегації даних Pandas значно випереджає NumPy в швидкості. Це пояснюється тим, що Pandas був спеціально розроблений для ефективної роботи з табличними даними та має оптимізовані алгоритми для таких операцій.**

Отже, хоча NumPy може бути швидшим для базових числових обчислень над масивами, Pandas може значно перевершити його продуктивність при роботі з табличними даними, особливо для операцій, які вимагають сегментації, фільтрації, групування та агрегації даних. (не обов’язково)

**Слайд 27:**

У світі аналізу даних важливо розумно поєднувати можливості різних інструментів для максимальної ефективності.

**NumPy та Pandas** - два потужних помічника, які доповнюють один одного.

**NumPy** найкраще підходить для роботи з великими масивами чисел та швидких математичних обчислень. Завдяки оптимізованій реалізації ця бібліотека забезпечує високу продуктивність під час виконання складних операцій над числовими даними.

**З іншого боку, Pandas** спеціалізується на аналізі табличних даних. Зручні структури даних, гнучка індексація, робота з пропущеними значеннями та часовими рядами - все це робить Pandas незамінним помічником для маніпулювання та перетворення структурованих наборів даних.

Тож для максимальної користі в проектах аналізу даних слід комбінувати ці два інструменти. Наприклад, спочатку можна використати NumPy для початкової обробки числових даних, а потім передати результати у Pandas для подальшого аналізу, фільтрації й візуалізації. Така гібридна стратегія дозволяє отримати переваги високої швидкодії та універсальності в одному робочому процесі. Завдяки тісній інтеграції NumPy та Pandas, перехід між ними є безпроблемним.