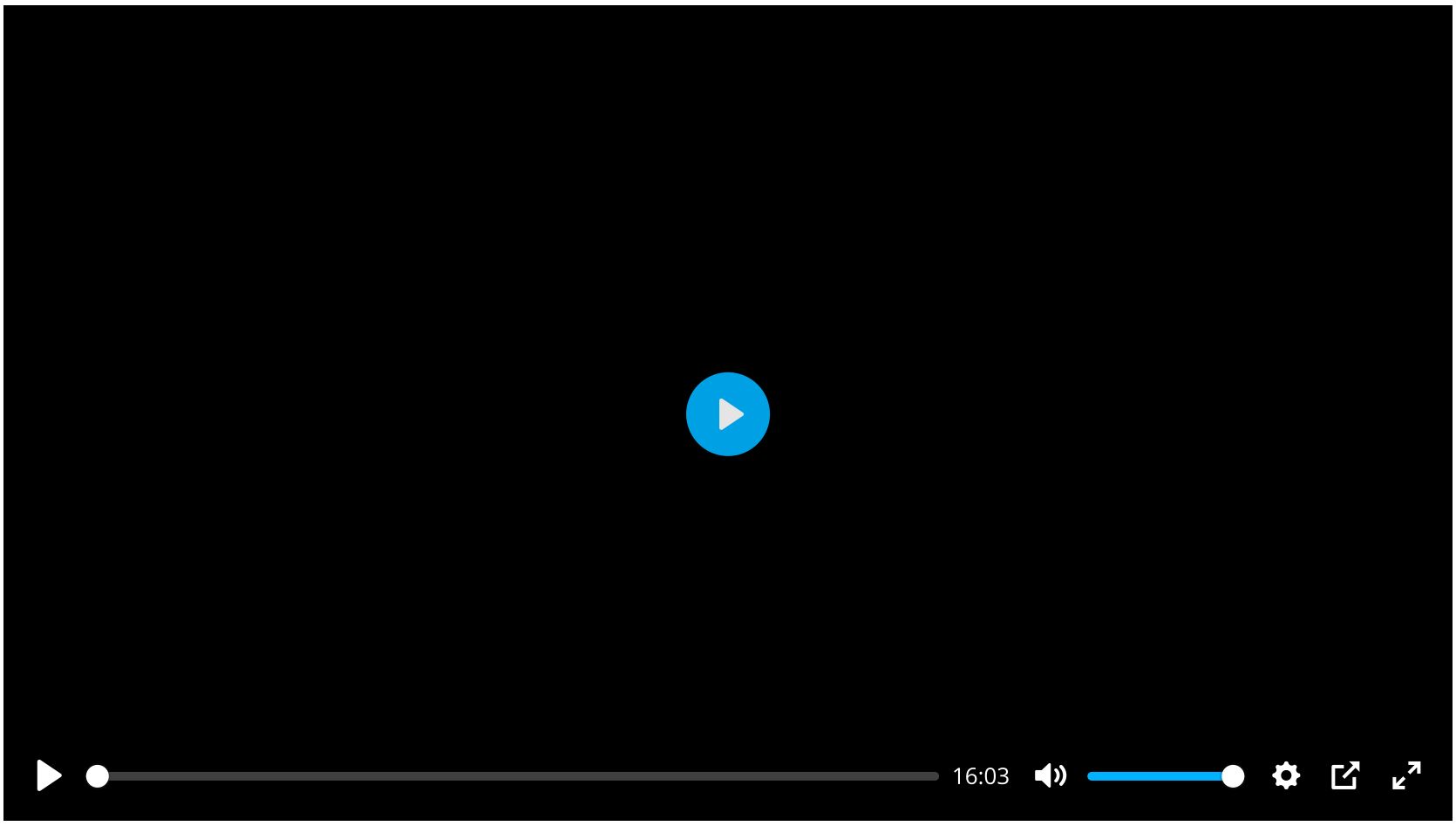
2.2 Инструменты экосистемы Python для работы с данными



После видео:

Курс для тех, кто хочет больше знать Numpy

Набор задачек по Numpy

Текстовая расшифровка видео:

ИЗВЕСТНЫЕ ПАКЕТЫ PYTHON ДЛЯ РАБОТЫ С ДАННЫМИ

План:

- Python как один из самых популярных языков программирования;
- Numpy и SciPy;
- Pandas/Polars/PyArrow;
- Pandas для DataEngineer’a;
- Итоги.

Python как один из самых популярных языков программирования

В компьютере существуют вычислительные устройства: CPU и GPU, мы хотим максимально эффективно использовать это устройство для процессинга данных: для того, чтобы считать, для обучения моделек в машинном обучении и т.д. С недавних пор, в современных аппаратных архитектурах появились расширенные наборы команд, которые не только позволяют делать одну маленькую операцию с одним значением, но и позволяют применять операции сразу к нескольким значениям (к вектору значений). Данный подход получил название «**SIMD**» (Single Instruction, Multiple Data), где за один такт процессора процессится целый ряд данных.

С 1970 годов разрабатывались библиотеки/стандарты, которые описывали правила запуска подобных операций на аппаратных архитектурах. Такой набор стандартов получил название «**BLAS**» (Basic Linear Algebra Subprograms). Позже стали появляться многочисленные реализации BLAS, например, OpenBLAS, LAPACK, Atlas, Intel MKL (коммерческое, закрытое решение, распространяющееся с платным компилятором), CuBLAS (библиотека для реализации линейной алгебры поверх технологии CUDA) и т.д. Подобные стандарты и библиотеки существуют с давних времен и активно развиваются, поэтому в разных языках (особенно в академической сфере) начали использовать эти подходы. Например, зная

Matlab или схожий язык, вы знаете, что даже на уровне переменных, на уровне операндов, которые мы можем ставить в уравнение, у нас могут выступать как скалярные величины, так и векторы и целые матрицы, из-за чего мы можем оперировать ими как хотим.

Вследствие этого, появилось понимание: если мы хотим, чтобы язык стал удобным в применении в вычислительной сфере (аналитике), ему необходимо иметь библиотеки, которые, во-первых, эффективно работают с математическими абстракциями, реализуя линейную алгебру, во-вторых, абстрагируют внутри себя BLAS, а внешне выдают понятный и удобный интерфейс. В Python и есть эта библиотека, которая привела к взрывному росту популярности и спросу к данному языку программирования.Называется эта библиотека – «Numpy». Позже появилась библиотека «SciPy», в которой вынесли отдельные разные научные вычисления.

Numpy и SciPy

Numpy – библиотека, внутри которой используется оптимизированные векторные вычисления. Она существует в разных сборках (например, как с OpenBLAS, так и с Intel MKL).

Пример кода:

```
# матрица 8x8 - шахматная доска
Z = np.zeros((8, 8), dtype=int)
Z[1::2, ::2] = 1
Z[::2,1::2] = 1

# она же
np.tile(np.array([[0, 1], [1, 0]]), (4, 4))
```

Данный пример (на двух вариантах кода) – пример того, как сделать матрицу в виде шахматной доски (8x8).

Мы можем наблюдать работу сразу с несколькими измерениями, мы можем читать их и присваивать им значения. Также мы можем с помощью np.tile «замостить» матрицу маленькими фрагментами.

Основная идея Numpy заключается не столько в возможности внутреннего использования оптимизированной векторной операции, сколько в наличии низкоуровневой системы типов, которую Numpy привносит в Python.

В Python цельночисловой тип – int, который может хранить любые числа (знаковые/беззнаковые и т.д.), упрощающий работу. Однако, когда речь идет о больших датасетах, необходимо понимать их количество в памяти. Если мы будем знать особенности данных, то сможем использовать более специфические типы, которые занимают меньше места, а также, в некоторых случаях быстрее процессятся. Поэтому, в Numpy можно использовать типы данных, совпадающих с другими языками (например, с «C» (Си)). Это позволяет экономить память. Также вся низкоуровневая «магия» написана внутри BLAS’a на Си и на Фортране, вследствие чего, мы получаем высокую производительность при научных вычислениях на Python.

Главное правило работы с данными в векторном формате – всегда использовать вызовы методов фреймворка вместо циклов самого ЯП.

Так как все написано на низкоуровневых языках, для эффективности вычислений матриц и операций линейной алгебры необходимо произвести сдвиг парадигмы и при оптимизации кода не использовать примитивы Python. Например, не стоит делать перемножение матриц с циклами, так как Numpy обладает разными вызовами функций, которые ускорят этот процесс.

Для любой операции в Numpy есть готовая реализация, которую нужно лишь найти. При ручном написании не будут использованы низкоуровневые оптимизации. Это может привести к замедлению работы кода.

С точки зрения дата-инженеров важно понимать следующее: если приходит запрос на оптимизацию кода, необходимо посмотреть, во-первых, какой используется BLAS, какая версия Numpy и с чем она собрана, во-вторых, постараться использовать внутренние методы из библиотеки Numpy/SciPy, избегая

использования «Питоновских» типов.

Pandas/Polars/PyArrow

Numpy/SciPy – библиотеки, которые в первую очередь работают с числовыми матрицами, однако в реальной жизни данные выглядят немного иначе. Это не просто числа в двумерном представлении. Данные могут быть разных видов. Например, категориальными, почисловыми, строковыми, логическими (true/false) и т.д.

Часто данные представлены примерно следующим образом:

Name	Thread pitch (mm)	Minor diameter tolerance	Nominal diameter (mm)	Head shape	Price for 50 screws	Available at factory outlet?	Number in stock	Flat or Phillips head?
M4	0.7	4g	4	Pan	$10.08	Yes	276	Flat
M5	0.8	4g	5	Round	$13.89	Yes	183	Both
M6	1	5g	6	Button	$10.42	Yes	1043	Flat
M8	1.25	5g	8	Pan	$11.98	No	298	Phillips
M10	1.5	6g	10	Round	$16.74	Yes	488	Phillips
M12	1.75	7g	12	Pan	$18.26	No	998	Flat
M14	2	7g	14	Round	$21.19	No	235	Phillips
M16	2	8g	16	Button	$23.57	Yes	292	Both
M18	2.1	8g	18	Button	$25.87	No	664	Both
M20	2.4	8g	20	Pan	$29.09	Yes	486	Both
M24	2.55	9g	24	Round	$33.01	Yes	982	Phillips
M28	2.7	10g	28	Button	$35.66	No	1067	Phillips
M36	3.2	12g	36	Pan	$41.32	No	434	Both
M50	4.5	15g	50	Pan	$44.72	No	740	Flat

Примерно так выглядят данные в Excel у аналитиков. Подобное представление данных с наличием типизированных колонок и индивидуальной колонкой, представляющей отдельную сущность/эксперимент с параметрами называется «Датафрейм», который появился сравнительно давно.

Библиотеки в Python, позволяющие работать с данными такого вида:

Pandas – часто используемая библиотека для аналитики. Данная библиотека не самая быстрая. Внутри себя она использует Numpy-вектаризацию, однако для удобства использует «обвязки» Python.

Способы ускорения Pandas:

- Использовать примитивы Numpy (вызвать подлежащие методы напрямую);
- Использовать Polaris (быстрее по производительности, но многое не поддерживает);
- Использовать PyArrow (цель – компактное хранение данных, выступает в роли общего языка для разных систем);
- Использовать DASK (позволяет разбивать данные на кусочки и обрабатывать на разных машинах).

Pandas для DataEngineer’a

- Данные хранятся в виде более низкоуровневых типов. Это позволяет эффективнее оперировать ими в памяти.
- Интеграция с PyArrow позволяет эффективнее работать с данными в колоночных форматах, например, Parquet или формат SAS.
- Идет работа по более тесной интеграции Pandas и Spark API.

Пример кода:

```
# чтение датасета напрямую по URL
iris = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv
# чтение из базы данных
conn = MySQLdb.connect(host='localhost', port=3306, user='vasya', passwd='pass', db='zoo')
pd.read_sql('select * from animals;', con=conn)
```

Мы можем прочитать данные как по ссылке, так и создав подключение к базе данных. При этом системы могут быть разными, так как поддерживаются разные источники. Как инженеры мы можем интегрировать одно с другим.

Как вам урок?

Далее