Python e seu ecossistema

Foco: análise de dados

Introdução:

- Utilização do Python (automação, desktop, Web, ciências, ciências de dados)
- Facilidade (foco na solução do problema, não na complexidade da programação)
- Python no mundo da datascience
- Conceito de módulos e pacotes (Boas práticas de modularizar - reuso de código)
- Biblioteca Padrão (Baterias inclusas)
- Bibliotecas de terceiros (Python Index Package)
- Comentar sobre ambiente virtualizado - Python virtual environment - venv (Boas práticas)

Por que recorrer ao Numpy e ao Pandas se o Python já oferece tanto?

- **Desempenho** superior em **operações numéricas**
- Manipulação **eficiente** de **grandes volumes de dados**
- **Funcionalidades prontas** para análise **estatística**
- **Operações vetorizadas** e **sem loops** explícitos
- **Integração** fácil com outros **pacotes de ciência de dados**

NumPy - Numerical Python

Quando trabalhamos com **ciência de dados** ou sistemas transacionais, precisamos lidar com **grandes volumes de informações estruturadas**.

Em geral, podemos dizer que os **modelos de Machine Learning** são grandes **modelos estatísticos**. Eles **sabem trabalhar** muito bem **com números**. Então acontece uma **tradução**, **independente do formato** que está se trabalhando (**textos, imagens, vídeos**), para **valores numéricos**.

Sobre:

- O **NumPy** é uma biblioteca voltada para computação numérica de alto desempenho.
- É uma biblioteca de terceiro, portanto, preciso instalá-la via pip para poder usá-la: (Ambiente local: pip install numpy, No Notebook: !pip install numpy. Obs.: Google Colab ou Ambiente Anaconda já traz instalada por padrão)
- Sua estrutura principal é o **array multidimensional (ndarray)**, que é **otimizado** para **armazenar** e **manipular grandes volumes de dados numéricos de forma eficiente**.
- Ele é amplamente utilizado para **cálculos matemáticos** e **estatísticos**, **álgebra linear** e **manipulação de grandes matrizes**.
- Exemplos de aplicação:
    - Cálculo de estatísticas em séries temporais financeiras
    - Processamento de imagens e sinais
    - Simulações científicas e computação vetorizada

Estrutura de um ndarray e suas diferentes dimensões

É o tipo de dado principal do NumPy: uma **estrutura de dados em forma de array (vetor ou matriz)**, capaz de armazenar **valores numéricos de forma eficiente** e realizar operações vetoriais com desempenho muito superior às listas do Python.

Exemplo com 1 dimensão (Vetor - Array simples):

```
import numpy as np

vetor = np.array([10, 20, 30, 40])
print(vetor)
print("Shape:", vetor.shape)
print("Dimensões:", vetor.ndim)
```

Entendendo o código-fonte:

- `vetor` é um array com **4 elementos**.
- Tem **1 só eixo**, como uma **linha de valores**.
- `.shape` mostra a **forma** do array (4 linhas) → `(4,)`
- `.ndim` mostra a quantidade de **dimensões** (profundidade 1) → 1

Exemplo com 2 dimensões (Matriz - Array de arrays):

```
import numpy as np

matriz = np.array([[1, 2, 3], [4, 5, 6]])
print(matriz)
print("Shape:", matriz.shape)
print("Dimensões:", matriz.ndim)
```

Entendendo o código-fonte:

- *matriz é uma matriz com 2 linhas e 3 colunas.*
- *matriz.shape → (2, 3) → 2 linhas, 3 colunas*
- *matriz.ndim → 2 dimensões*

Dê-me dois motivos do porquê é importante aprender NumPy:

Motivos: Desempenho e facilidade, veja:

Problema real: você tem uma lista de preços de produtos comercializados por uma empresa e recebe a missão de aplicar 20% de aumento no preço destes produtos sem perder o valor antigo.

Vamos analisar duas abordagens diferentes para resolução deste problema.

Mais lento, percorrendo item a item, por meio do laço de repetição for:

```
tabela_precos = [100, 88, 72, 23, 99, 300, 1000, 8000, 4000, 999, 333,
849, 777, 555, 444, 333, 222, 111]
nova_tabela_precos = []

%time
for preco in tabela_precos:
    novo_preco = preco * 1.25
    nova_tabela_precos.append(novo_preco)

print(nova_tabela_precos)
```

Mais rápido, utilizando a estrutura ndarray do NumPy sem utilizar:

```
import numpy as np

tabela_precos_2 = np.array([100, 88, 72, 23, 99, 300, 1000, 8000, 4000, 999, 333, 849, 777,
555, 444, 333, 222, 111])

%time
nova_tabela_precos_2 = tabela_precos_2 * 1.25
print(nova_tabela_precos_2)
```

Importante:

- Compare os resultados da contagem do tempo de execução dos dois códigos. Tempo em microssegundos (µs é uma unidade de tempo que representa um milionésimo de um segundo 0,000001 s).

- O %time é um “magic command” disponível em Notebooks (Jupyter Notebook ou Colab) que mede o tempo de execução da próxima linha. O resultado é apresentado posteriormente à execução da linha que está sendo executada.

Navegando, manipulando e operando na estrutura de um ndarray com um problema do mundo real:

O problema:

Uma rede de lanchonetes tem 3 lojas. Você recebeu a planilha com as vendas de hambúrgueres de cada loja ao longo da semana (segunda a domingo). A matriz representa os dados assim:

	Seg	Ter	Qua	Qui	Sex	Sab	Dom
Loja 1	50	60	55	58	80	120	100
Loja 2	40	45	50	48	70	110	90
Loja 3	35	38	40	42	60	105	85

Código-fonte com a representação dos dados estruturados em Python:

```
import numpy as np

# Criando o array com os dados de vendas
vendas = np.array([
    [50, 60, 55, 58, 80, 120, 100],  # Loja 1
    [40, 45, 50, 48, 70, 110, 90],   # Loja 2
    [35, 38, 40, 42, 60, 105, 85]    # Loja 3
])

dias_semana = ['Seg', 'Ter', 'Qua', 'Qui', 'Sex', 'Sab', 'Dom']
lojas = ['Loja 1', 'Loja 2', 'Loja 3']
```

Acessar elemento específico (Ex: vendas da Loja 2 na sexta-feira):

```
print(‘Vendas da Loja 2 na sexta-feira:’, vendas[1, 4], ‘hambúrgueres’)
```

Alterar um valor (Ex: corrigir vendas da Loja 3 no domingo para 88):

```
vendas[2, 6] = 88
print(‘Novo valor para Loja 3 no domingo:', vendas[2, 6]’)
```

Somatório das vendas por loja:

```
print(‘Total por loja:’, np.sum(vendas, axis=1))  # eixo 1 = linha
```

Somatório das vendas por dia, independentemente da loja:
```
print("Total por dia:", np.sum(vendas, axis=0))  # eixo 0 = coluna
```

Média de vendas por loja:
print("Média por loja:", np.mean(vendas, axis=1))

Média de vendas por dia, independente da loja:
```
print("Média por dia:", np.mean(vendas, axis=0))
```

Mediana geral das vendas (Explicar a importância do uso da mediana, principalmente quando se trata de valores heterogêneos com alta discrepância - útil quando for trabalhar com tratamento de dados - Abrir parênteses aqui para exemplo com salários de uma empresa):
```
print('Mediana geral das vendas:', np.median(vendas))
```

Mediana das vendas por loja independente do dia:
print('Mediana por loja:', np.median(vendas, axis=1))

Estamos analisando dados como se estivéssemos numa planilha. O NumPy nos permite acessar qualquer valor com [linha, coluna], assim como uma coordenada em uma tabela. E além disso, conseguimos fazer operações como somatório, média e mediana com **pouquíssimas linhas de código** e de forma **muito rápida**.

Agora é a sua vez. Vamos praticar com um exercício que envolve um problema do mundo real:

Exercício: Análise de Vendas de Sorvetes nas Férias de Verão.

Você é um analista de dados contratado por uma franquia de sorveterias chamada GelatoTech. Durante as férias de verão, eles monitoraram as vendas de sorvetes em 4 lojas diferentes, ao longo de 7 dias consecutivos (de segunda a domingo).

Sua missão é analisar os dados de vendas e extrair insights importantes para a diretoria da empresa.

Código-fonte com os dados estruturados em um Array NumPy:

```
import numpy as np

# Vendas em R$ mil (linhas: lojas | colunas: dias da semana)
vendas_sorvete = np.array([
    [12, 15, 14, 10, 20, 35, 30],  # Loja 1
    [10, 11, 12,  9, 18, 32, 28],  # Loja 2
    [ 9,  8, 11, 10, 19, 30, 25],  # Loja 3
    [13, 14, 13, 11, 21, 36, 31]   # Loja 4
])

dias = ['Seg', 'Ter', 'Qua', 'Qui', 'Sex', 'Sab', 'Dom']
lojas = ['Loja 1', 'Loja 2', 'Loja 3', 'Loja 4']
```

1. Qual loja vendeu mais no total da semana?
Dica: np.sum(..., axis=1) retorna a soma por loja.

2. Qual foi o dia de maior faturamento geral?
Dica: some por coluna (eixo 0) e encontre o maior valor.

3. Qual foi a média de vendas por loja ao longo da semana?

4. Qual loja teve a menor mediana de vendas diárias?

5. Quantos dias a Loja 1 vendeu mais do que R$ 25 mil?

6. Crie uma nova matriz com as vendas acrescidas de 10% (simulando um reajuste nos preços dos sorvetes).
Dica: vendas_sorvete * 1.10

7. [DESAFIO] A empresa quer premiar a loja que mais cresceu nas vendas do final de semana (Sábado e Domingo).
Calcule o total do fim de semana para cada loja e indique a campeã.

Entrega do exercício:
Você pode entregar suas respostas dentro de um notebook do Google Colab, com comentários explicando seus resultados de forma clara, como se estivesse apresentando para o seu chefe. Compartilhe com: robertson@ersistemas.info e alexandre.o.zamberlan@gmail.com