escola britânica de artes criativas & tecnologia

Profissão: Analista de dados

escola britânica de artes criativas & tecnologia

FUNDAMENTOS DE MATEMÁTICA

GUIA DA AULA 1

escola britânica de artes criativas & tecnologia

Conheça a vetorização de problemas

- **Abordagens estatísticas**
- **Introdução a vetorização**
- **Pacote NumPy**

Acompanhe aqui os temas que serão tratados na videoaula

Abordagens estatísticas

Há dois tipos de abordagens estatísticas:

1. **Descritiva**: foco no passado para entender o presente.
2. ****Preditiva****: foca no passado para inferir o futuro.

Introdução à vetorização

Derivado da matemática computacional, vetorização é o processo de transformar um código escalar em sua forma vetorial. Na prática, geralmente eliminamos operações matemáticas que lidam com um conjunto reduzido de números por laço de repetição (escalar) por uma única operação com todos os números (vetor).
A vetorização (geralmente) é mais rápida e utiliza menos memória.

Exemplo:
Uma *fintech* quer saber a qualidade dos investimentos dos seus clientes. Para tanto, é proposta a seguinte análise: para um determinado mês, deve-se comparar o retorno de cada cliente obtido através do investimento com o valor corrigido pela inflação.

Deve-se, então, calcular quantos clientes "perderam" para inflação.

- Para 1 cliente:

In [ ]:
```
montante = 1000
montante_final = 1001.5 # ~ 0.15% de retorno
```

In [ ]:
```
ipca = 0.25 / 100
montate_inflacao = montante * (1 + ipca)

print(montate_inflacao)
```

In [ ]:
```
print(montante_final < montate_inflacao)
```

- Para 100 clientes:

In [ ]:
```
from random import random, randint

montante_lista = [randint(0, 5000) for _ in range(0, 100000)]
montante_final_lista = [round(montante * (1 + (0.3 * random() / 100)),
2)
                        for montante in montante_lista]
```

In [ ]:
```
print(montante_lista[0:5])
print(montante_final_lista[0:5]
)
```

O laço repetição *for/in* opera de forma escalar, ou seja, processa um conjunto limitado de números por repetição.

In [ ]:

```
from time import time

perdeu = list()

# calculo
inicio = time() # inicio da contagem do tempo
for montante, montante_final in zip(montante_lista,
  montante_final_lista): perdeu.append(montante_final - montante * (1 +
  ipca))
fim = time() # fim da contagem do tempo

# resultado
perdeu = list(filter(lambda val: True if val < 0 else False,
perdeu)) print(f"{len(perdeu)} cliente perderam para inflação")

# tempo
tempo_lista = fim -
inicio print(fim -
inicio)
```

Pacote NumPy

Pacote Python para manipulação numérica construído na linguagem de programação C, muitos pacotes o utilizam como base, como o Pandas e SciPy. A documentação pode ser encontrada no *link* https://numpy.org/.

A abstração base do NumPy é o *array*, uma estrutura de dados Python de duas ou mais dimensões utilizado para representar vetores, matrizes, tensores etc.

Vamos vetorizar nosso cálculo através da criação de vetores NumPy.

In [ ]:

```
import numpy as np

montante_array = np.array(montante_lista)
montante_final_array =
np.array(montante_final_lista)
```

In [ ]:
```
print(montante_array)
print(montante_final_array
)
```

Vamos então fazer o mesmo cálculo com os vetores NumPy de forma vetorial, ou seja, processando todos os números de uma vez.

In [ ]:
```
from time import time

# calculo
inicio = time() # inicio da contagem do tempo
perdeu = montante_final_array - montante_array * (1 +
ipca) fim = time() # fim da contagem do tempo

# resultado
perdeu = list(filter(lambda val: True if val < 0 else False,
perdeu)) print(f"{len(perdeu)} cliente perderam para inflação")

# tempo
tempo_array = fim - inicio
print(f"Duração:
{tempo_array}")
```

Por fim, vamos comparar os tempos de execução das operações escalares e vetoriais.

In [ ]:

```
tempo_lista / tempo_array
```