ICC204 - Aprendizagem de Máquina e Mineração de Dados

Python, Jupyter e Weka

Prof. Rafael Giusti
rgiusti@icomp.ufam.edu.br

Primeiramente, obrigado

- Obrigado a todos que responderam o questionário
- Eu analisei todas as respostas...
- Mas apenas "corrigi" a primeira para indicar que eu já havia olhado a submissão
- O propósito desse questionário não é dar nota
- Quem fizer questão de um *feedback* pode me pedir no fim da aula ou mandar um email

Primeiramente, obrigado

- Obrigado a todos que responderam o questionário
- Minhas conclusões foram...
  - Quase todos preferem usar quatro espaços para indentação do código
  - Os nomes a e b são seus parâmetros preferidos
    - x e y: 4
    - a e b: 12
    - Outros: 5

Primeiramente, obrigado

- Outras conclusões:
  - Quase todos preferem usar aspas simples para representar *strings*
    - `'hello world'`
    - `"hello world"`
  - Quem usou *lambda* preferiu o Python 3
    - Incluíram `functools` pra poder usar `reduce`

Primeiramente, obrigado

- Mas agora o que eu realmente queria...
  - A turma é bastante heterogênea
  - Consenso é que
    - Todos conhecem algo de Python
    - Sabem definir funções

Primeiramente, obrigado

- Mas agora o que eu realmente queria...
  - Onde teve mais divergência?
    - Como as referêcias do Python são diferentes de referências de outras linguagens
    - Quando alterar um parâmetro formal altera o parâmetro real
    - Como utilizar funções *lambda*
    - O que é **lista + lista**

Primeiramente, obrigado

- Como esse resultado vai influenciar na disciplina?
  - Alguns *slides* foram sacrificados
  - Vamos focar nos pontos em que houve um pouco mais de divergência
  - Vou comentar algumas coisas que eu aprendi com vocês
    - A diferença entre `=` e `+=`
    - Um recurso novo do Python 3.6: f-*strings*

Agenda

- Introdução ligeira ao Python
- Jupyter *notebooks*
- NumPy
- Pandas
- Demonstração do WEKA
- Regressão linear com Scikit-Learn

Agenda

- Introdução ligeira ao Python
- Jupyter *notebooks*
- NumPy
- Pandas
- Demonstração do WEKA
- Regressão linear com Scikit-Learn

Pontos essenciais de Python

- Python é uma linguagem multi-paradigma, mas com um viés para programação orientada a objetos
  - **Tudo** em Python é um objeto
  - Na expressão `a + b` o operador é "substituído" por uma mensagem para o objeto `a`
    - `a.__add__(b)`

Python não tem encapsulamento. Todos os membros são públicos.
Convenciona-se que métodos com __ devem ser tratados como privados

Pontos essenciais de Python

- Python utiliza um conceito chamado *duck typing*
- Vem da frase:
  - *Se alguma coisa se parece com um pato e anda como um pato, então é um pato*
- No duck typing, o que realmente importa é o que o objeto é capaz de fazer, não o que ele é

Pontos essenciais de Python

- Isso implica algumas coisas boas e outras ruins
- As funções não restringem os argumentos que elas recebem; simplesmente tentam utilizá-los

```
>>> def faz_quack(pato):
...     pato.quack()
...
```

```
>>> faz_quack(pato)
'Quack'
```

```
>>> faz_quack(42)
AttributeError
```

Pontos essenciais de Python

- Semelhantemente para operadores

```
>>> def soma(a, b):
...     return a + b
...
```

```
>>> soma(42, 13)
55
```

```
>>> soma('a', 'b')
'ab'
```

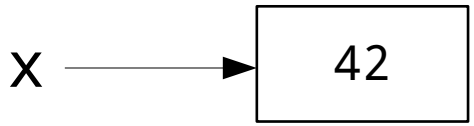
```
>>> soma([0, 1], ['x'])
[0, 1, 'x']
```

```
>>> soma(0, 'x')
TypeError
```

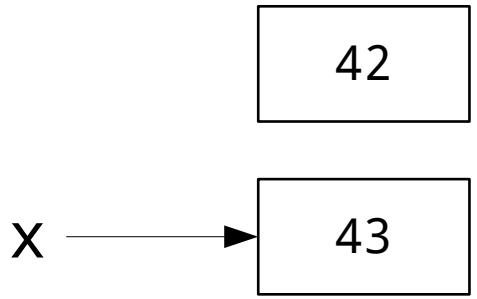
Pontos essenciais de Python

- Python não exige que as variáveis sejam declaradas
- Todas as variáveis são referências aos objetos
- O operador = não altera um objeto, apenas muda a referência

```
x = 42
```




```
x = x + 1
```



Pontos essenciais de Python

- Isso vale também para funções

A variável P passa a referenciar outro objeto...

```
>>> def insere_pilha(P, x):
...     P = P + [x]
...
>>> l = []
>>> insere_pilha(l, 1)
>>> l
[]
```

Mas isso não afeta `l`

Pontos essenciais de Python

- Isso vale também para funções

```
>>> def insere_pilha(P, x):
...     P.append(x)
...
>>> l = []
>>> insere_pilha(l, 1)
>>> l
[1]
```

Pontos essenciais de Python

- Os objetos são definidos como mutáveis ou imutáveis
- Objetos mutáveis são aqueles que possuem métodos que alteram o estado do objeto
- Objetos imutáveis não possuem esses métodos

Pontos essenciais de Python

- Exemplos de objeto imutáveis:
  - Os tipos primitivos **`int`**, **`float`**, **`str`**, **`bool`** e **`complex`**
  - Os tipos compostos **`tuple`** e **`frozenset`**
- Exemplos de objetos mutáveis:
  - Os tipos compostos **`list`**, **`dict`** e **`set`**
  - Classes definidas pelo usuário

Pontos essenciais de Python

- Exemplos de métodos que alteram o estado do objeto:
  - `list.append`
  - `list.pop`
  - `list.__iadd__`

Adição "*in-place*": modifica o objeto que está recebendo a mensagem

- Métodos que não alteram o estado
  - `list.index`
  - `list.__add__`

Adição "comum": apenas retorna um novo objeto que representa o resultado da adição

Pontos essenciais de Python

- Outra forma de fazer a inserção

Será chamado o método `__iadd__`

```
>>> def insere_pilha(P, x):
...     P += [x]
...
>>> l = []
>>> insere_pilha(l, 1)
>>> l
[1]
```

Agenda

- Introdução ligeira ao Python
- Jupyter *notebooks*
- NumPy
- Pandas
- Demonstração do WEKA
- Regressão linear com Scikit-Learn

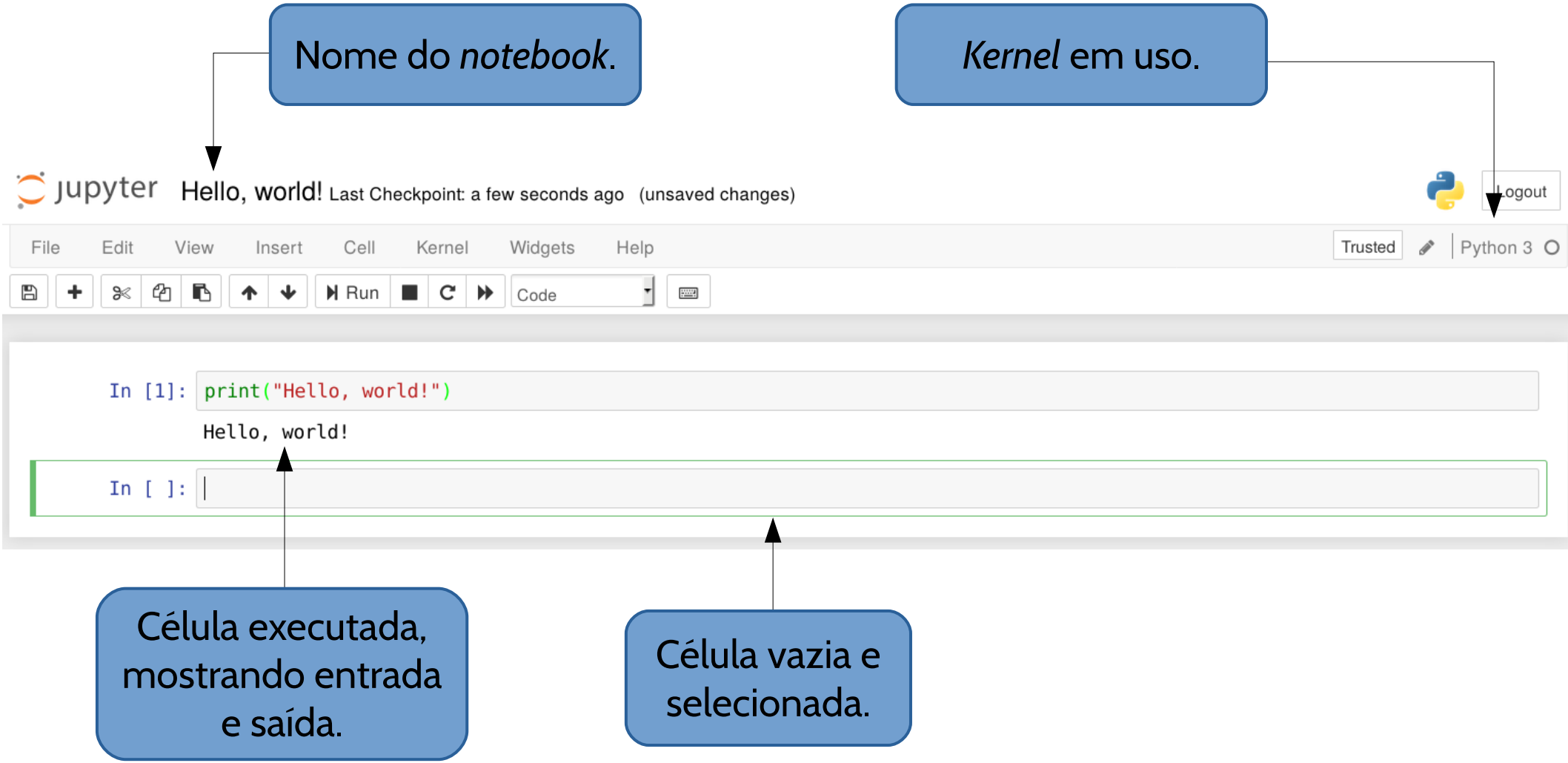
O que é são *notebooks*?

- Cada *notebook* instancia um *kernel* do Python
  - O *kernel* é um estado da máquina virtual do Python que permite a execução interativa de programas
  - Cada comando para o *kernel* pode consultar ou modificar o seu estado

O que é são *notebooks*?

- O *notebook* é organizado como uma coleção de células
  - Cada célula pode conter documentação em formato *markdown* ou código
  - As células de código possuem entrada e saída
  - A saída da célula é sempre o resultado da última expressão avaliada

O que é são *notebooks*?



Jupyter: o bom, o mau e o feio

- Vantagens do Jupyter
  - O navegador é uma interface bastante agradável
    - É fácil navegar entre as células
    - Elas podem ser modificadas, reorganizadas e removidas
  - Integração com bibliotecas que oferecem exibição rica de dados
  - As entradas e saídas ficam salvas no *notebook*

Jupyter: o bom, o mau e o feio

- Desvantagens do *notebook*
  - As células fornecem uma impressão um pouco falsa de independência e permanência
    - É possível executar as células em qualquer ordem, mas a ordem na qual elas são executadas afeta o resultado
    - Ao carregar o *notebook* novamente, todas as saídas estarão lá, mas as variáveis não existem mais

Jupyter: o bom, o mau e o feio

- Um "risco" do Jupyter
  - Existe uma tendência nas áreas de aprendizado de máquina e ciência de dados a fazer "tudo" com *notebooks*
  - Não se renda a essa tendência
    - *Notebooks* são ideais para explorar e estudar
    - Mas não são a ferramenta adequada para entregar produtos

Instalação do Jupyter

- Você pode instalar o Jupyter e os módulos individualmente
- No Debian, é muito fácil...
  - `# apt-get update`
  - `# apt-get install python3 python3-notebook numpy pandas matplotlib`

Instalação do Jupyter

- Existem aplicativos que distribuem diversos módulos ao mesmo tempo
  - Anaconda
    - Inclui Python e R
    - Inclui ferramentas para análise de dados, geração de modelos de aprendizado de máquina e interfaces de desenvolvimento
  - Miniconda
    - Parte do Anaconda que pode ser utilizada separadamente
    - Permite a instalação individual de pacotes

Agenda

- Introdução ligeira ao Python
- Jupyter *notebooks*
- NumPy
- Pandas
- Demonstração do WEKA
- Regressão linear com Scikit-Learn

Motivação para o NumPy

- Não apenas em AM, mas na computação numérica de maneira geral, é preciso manipular matrizes
- Exemplos de treinamento são armazenados em tabelas atributo-valor

$Atributo_1$	$Atributo_2$	...	$Atributo_m$
Potência do sinal do roteador 1	Potência do sinal do roteador 2	Potência do sinal do roteador 3	Potência do sinal do roteador 4

Matrizes em Python

- A coleção fundamental de dados em Python é a lista
  - Uma lista é uma coleção sequencial de referências a objetos
  - Essencialmente, um vetor de referências

Matrizes em Python

- Listas *podem* ser utilizadas para representar matrizes
  - Lista como matriz 3x3
    -
```
>>> [[11, 12, 13],
     [21, 22, 23],
     [31, 32, 33]]
```

11	12	13
21	22	23
31	32	32

11	21	31
12	22	32
13	23	33

Matrizes em Python

- Porém isso acarreta dois problemas
  - Listas são estruturas naturalmente heterogêneas
    - `>>> [[11, 12], 'dois', [31, 32, 33]]`

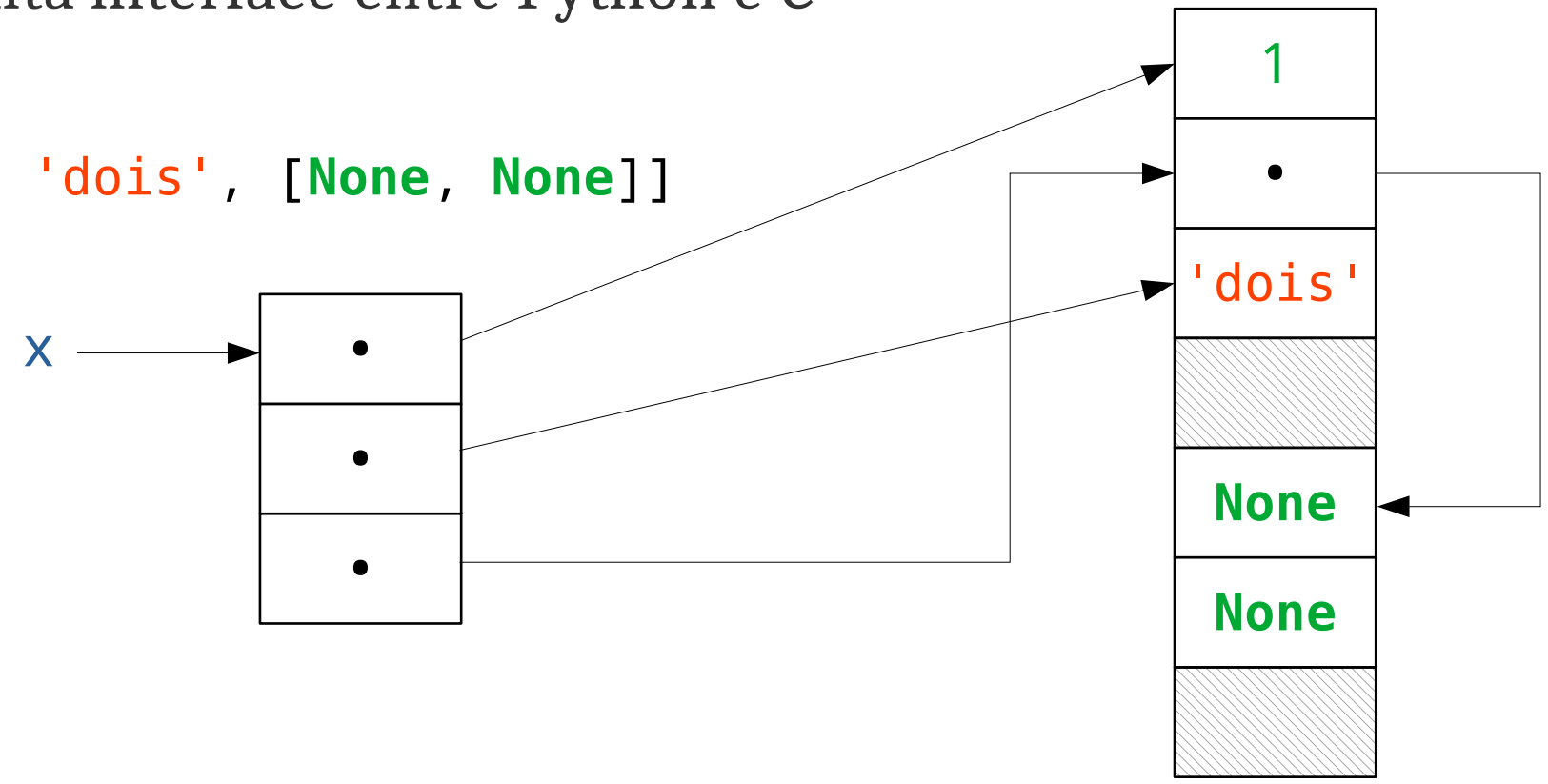
11	12	?
?	?	?
31	32	32

  - O que `lista + lista` deve fazer?

Matrizes em Python

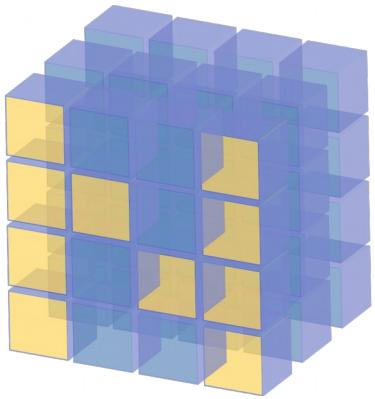
- Além disso, embora as referências sejam contíguas, os objetos podem estar "espalhados" pela memória
  - Isso causa ineficiência no *cache* dos dados
  - Dificulta interface entre Python e C

```
x = [1, 'dois', [None, None]]
```



NumPy, Pandas, TensorFlow

- Então o NumPy utiliza uma representação diferente para os dados matriciais
  - Sempre que possível, coleções de valores primitivos no lugar de objetos Python



NumPy

TensorFlow

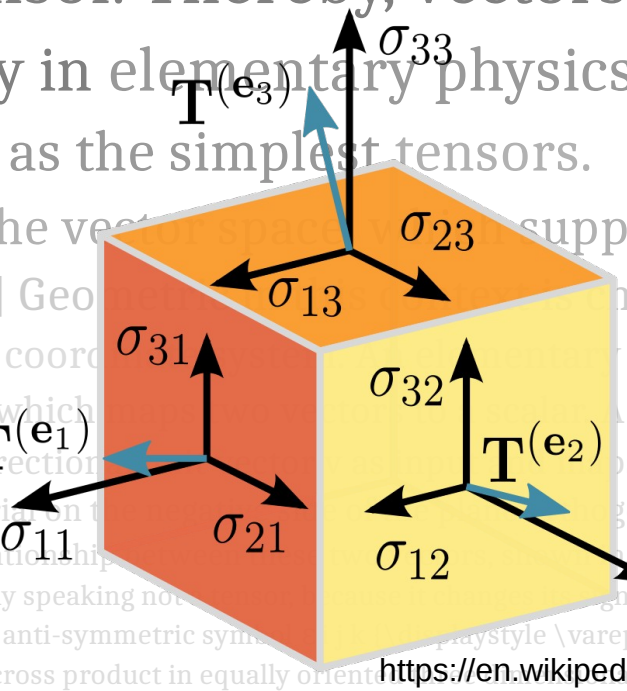
https://en.wikipedia.org/wiki/NumPy#/media/File:NumPy_logo.svg
https://en.wikipedia.org/wiki/TensorFlow#/media/File:TensorFlowLogo.svg

NumPy, Pandas, TensorFlow

- O NumPy oferece a estrutura de dados básica para todas as operações
  - O ***tensor***
  - Essencialmente um "vetor multidimensional"
- O Pandas constrói estruturas de dados de mais alto-nível em cima dos tensores
- Outras bibliotecas fazem uso de NumPy e Pandas conforme apropriado

Mas afinal, o que é um tensor?

- A definição **correta** e **complicada**, de acordo com a Wikipedia:
  - In mathematics, a tensor is a geometric object that maps in a multi-linear manner geometric vectors, scalars, and other tensors to a resulting tensor. Thereby, vectors and scalars themselves, often used already in elementary physics and engineering applications, are considered as the simplest tensors. Additionally, vectors from the dual space of the vector space which supplies the geometric vectors, are also included as tensors.[1] Geometrically, tensors are chiefly meant to emphasize indepen-dence of any selection of a coordinate system. One example of such a mapping, describable as a tensor, is the dot product, which A more complex example is the Cauchy stress tensor T, which takes a direction it to the stress vector T(v), which is the force (per unit area) exerted by material on gonal to v against the material on the positive side of the plane, thus expressing a relationship (right). The cross product, where two vectors are mapped to a third one, is strictly speaking not under those transformations that change the orientation of the coordinate system. The totally anti-symmetric symbol displaystyle \varepsilon _{ijk}} {\ displaystyle \varepsilon _{ijk}} nevertheless allows a convenient handling of the cross product in equally oriented



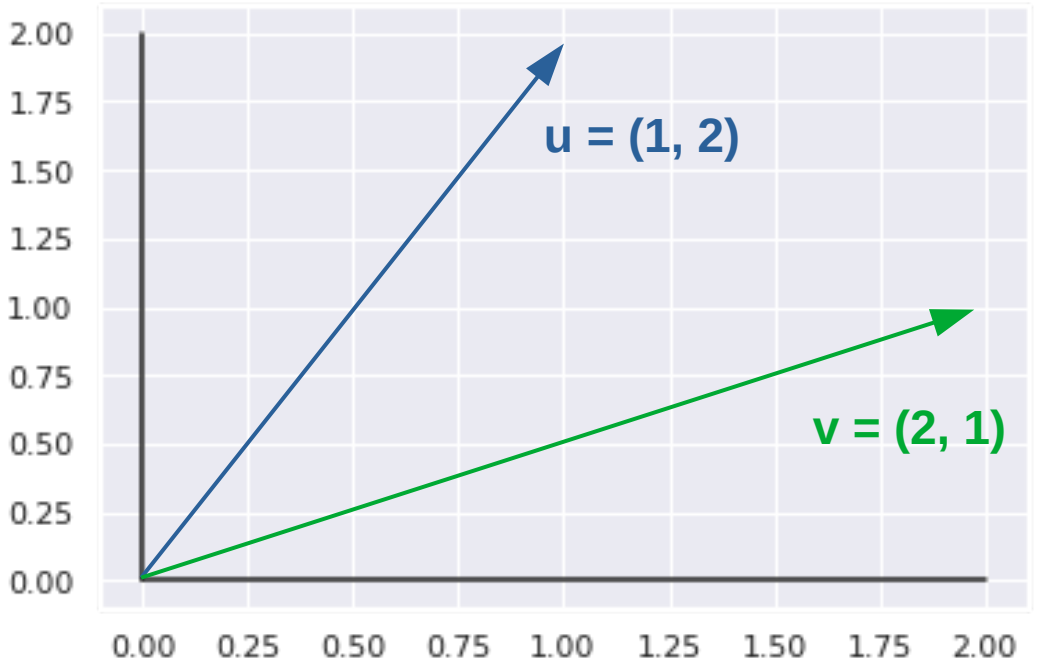
https://en.wikipedia.org/wiki/Tensor

Mas afinal, o que é um tensor?

- A explicação **errada**, mas **simples**:
  - Um tensor é uma matriz multidimensional
- Por que isso é errado?
  - Todo tensor pode ser representado como um vetor multidimensional...
  - Mas nem todo vetor multidimensional é um tensor
- Quando um vetor multidimensional não é um tensor?
  - ¯_(ツ)_/¯

Uma analogia

- Todo vetor, no plano, pode ser representado como um par ordenado



- Todo vetor *n*-dimensional pode ser representado como uma sequência ordenada de *n* elementos

Uma analogia

- Mas toda sequência ordenada é um vetor?
  - (1, 2, 3, 4, 5, 6, 7, 8) é um vetor?
- Depende do contexto
  - O que essa sequência representa?
  - Quais são as operações associadas a essa sequência?
- No jargão da computação, o termo vetor é "abusado" e tratado como sinônimo de "sequência ordenada"

Uma analogia

- Como você responderia a pergunta "quando uma sequência ordenada **não é** um vetor?"

Forçando a analogia

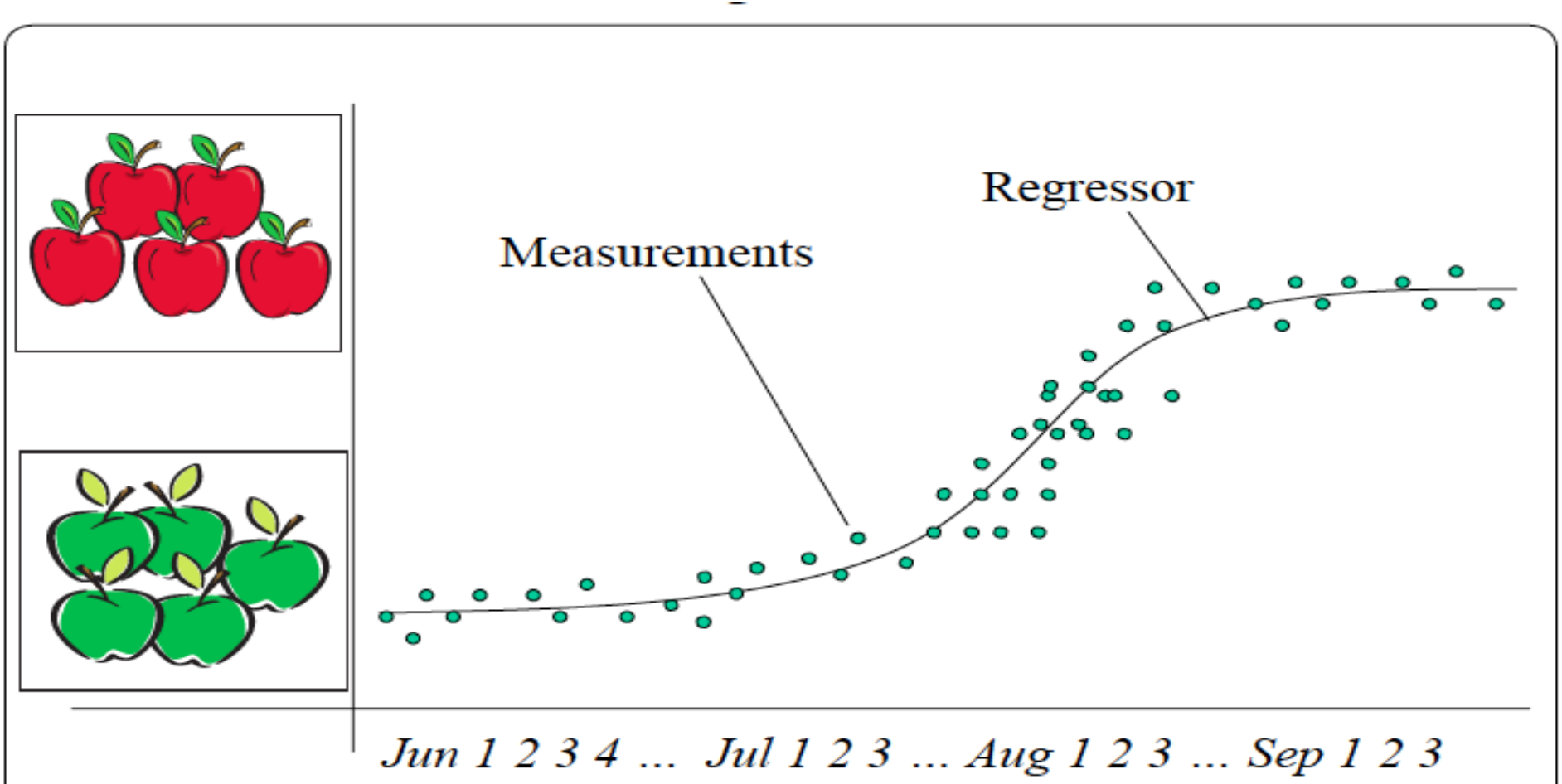
- Usaremos o termo "tensor" como uma generalização do conceito de "vetor"

Tensores

- São definidos por três propriedades
  - Número de dimensões
  - Formato
  - Tipo de dados

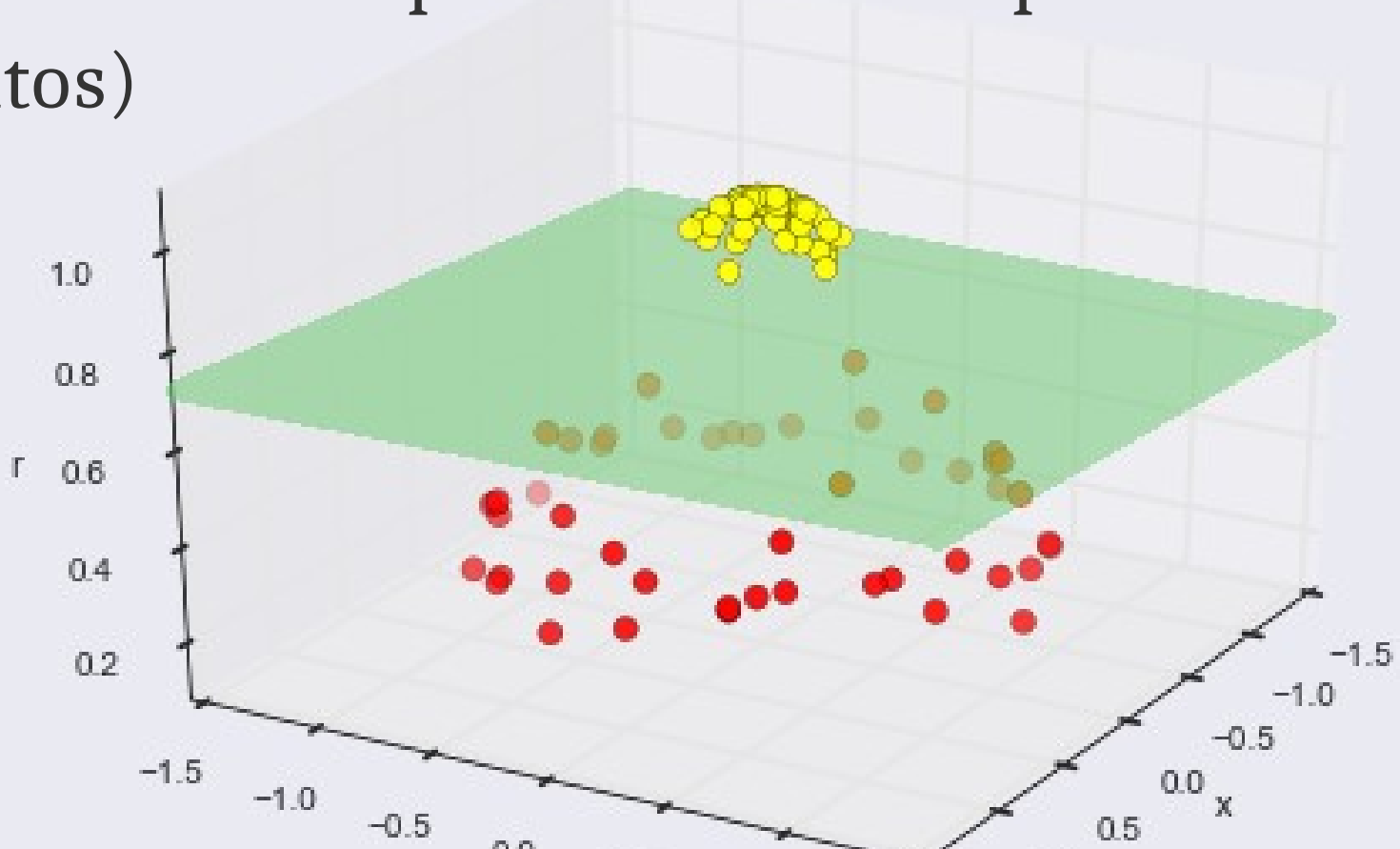
Tensores

- O que representam os tensores?
  - De 1 dimensão: série temporal



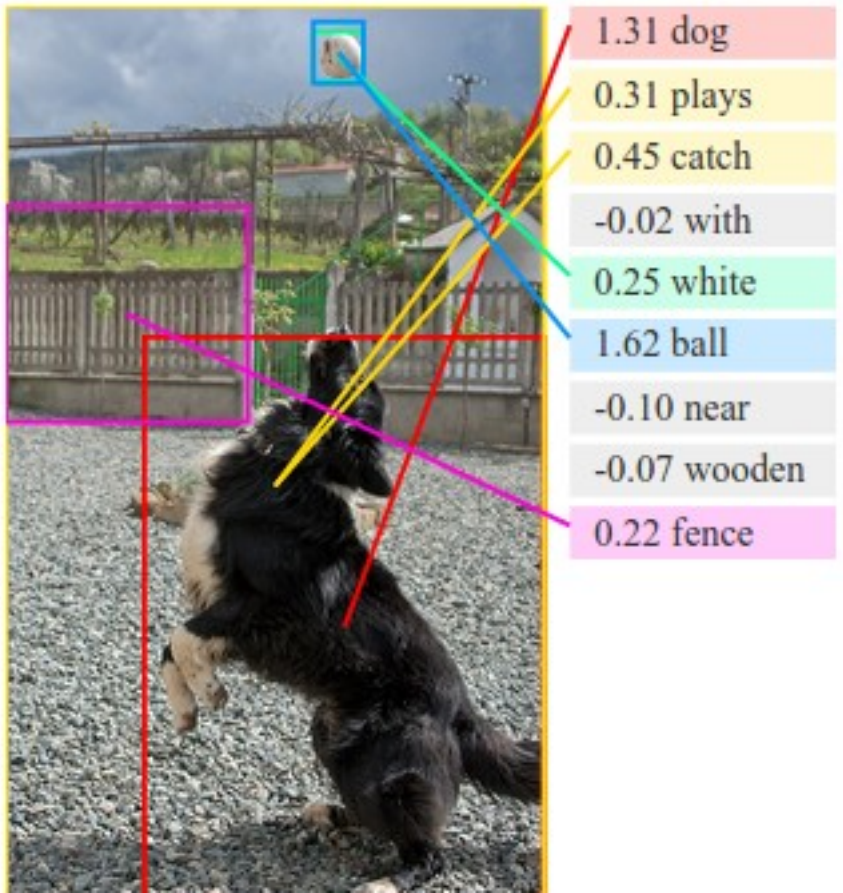
Tensores

- O que representam os tensores?
  - De 2 dimensões: tabela atributo-valor (não necessariamente para dados com apenas dois atributos)



Tensores

- O que representam os tensores?
  - De 3 dimensões: imagem em cores



- Dimensão 0: linhas da imagem
- Dimensão 1: colunas da imagem
- Dimensão 2: cores dos pixels

Tensores

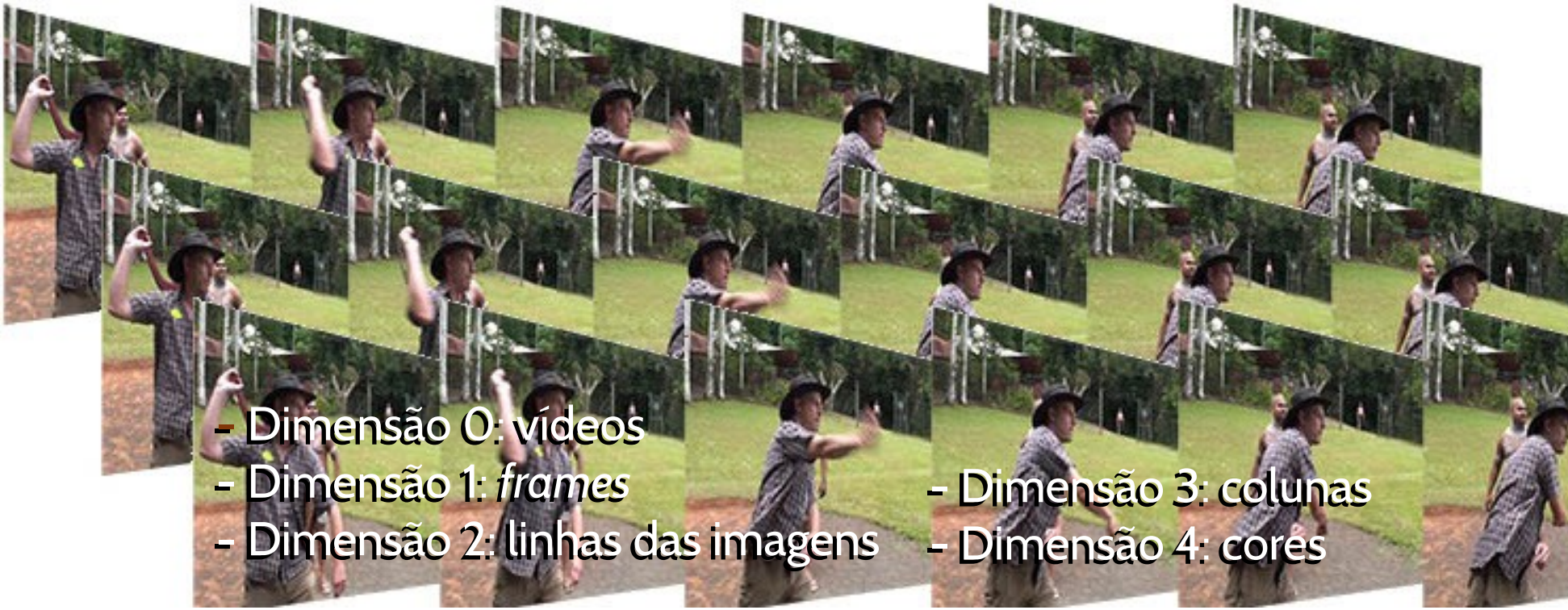
- O que representam os tensores?
  - De 4 dimensões: um vídeo

https://www.mediacollege.com/video/frame-rate/

- Dimensão 0: *frames*
- Dimensão 1: linhas das imagens
- Dimensão 2: colunas da imagens
- Dimensão 3: cores dos pixels

Tensores

- O que representam os tensores?
  - De 5 dimensões: uma coleção de vídeos



Tensores

- Tensores de dimensionalidade mais alta são importantes, principalmente para redes neurais
- Nas primeiras aulas da nossa disciplina, vamos utilizar vetores e matrizes
- Todos os conceitos se estendem, principalmente os de *broadcast* e fatiamento

Agenda

- Introdução ligeira ao Python
- Jupyter *notebooks*
- NumPy
- Pandas
- Demonstração do WEKA
- Regressão linear com Scikit-Learn

Pandas

- Pandas é uma biblioteca para dados estruturados
- O Pandas utiliza o NumPy e acrescenta:
  - Índices sobre os dados
  - Tabelas com tipos de dados heterogêneos

Pandas

- Pandas possui duas classes fundamentais
  - `pandas.Series`
    - Implementa uma série de dados
    - Uma coleção de valores associada a um índice
  - `pandas.DataFrame`
    - Implementa uma tabela
    - Uma coleção de séries associada a dois índices

Pandas: índice

- Primeiro, uma advertência: o termo "índice" pode significar mais de uma coisa
  - Índice
    - Estrutura de acesso no qual chaves ou rótulos permitem acesso eficiente a uma coleção de valores
  - Índice
    - Posição de um elemento em uma coleção com acesso sequencial

Fundamentos

- Essa ambiguidade aparece constantemente, pois todos os elementos em Pandas podem ser acessados de duas formas
  - Por meio do índice
    - Os rótulos são utilizados para encontrar os dados
  - Do jeito "tradicional"
    - Os dados são encontrados através da sua posição com respeito ao ínicio da sequência

Fundamentos

- Podemos gerar uma série a partir de uma lista
  -
```
>>> import pandas as pd
>>> primos = pd.Series([2, 3, 5, 7, 11, 13, 17, 19])
```
  - Coleção sequencial de valores

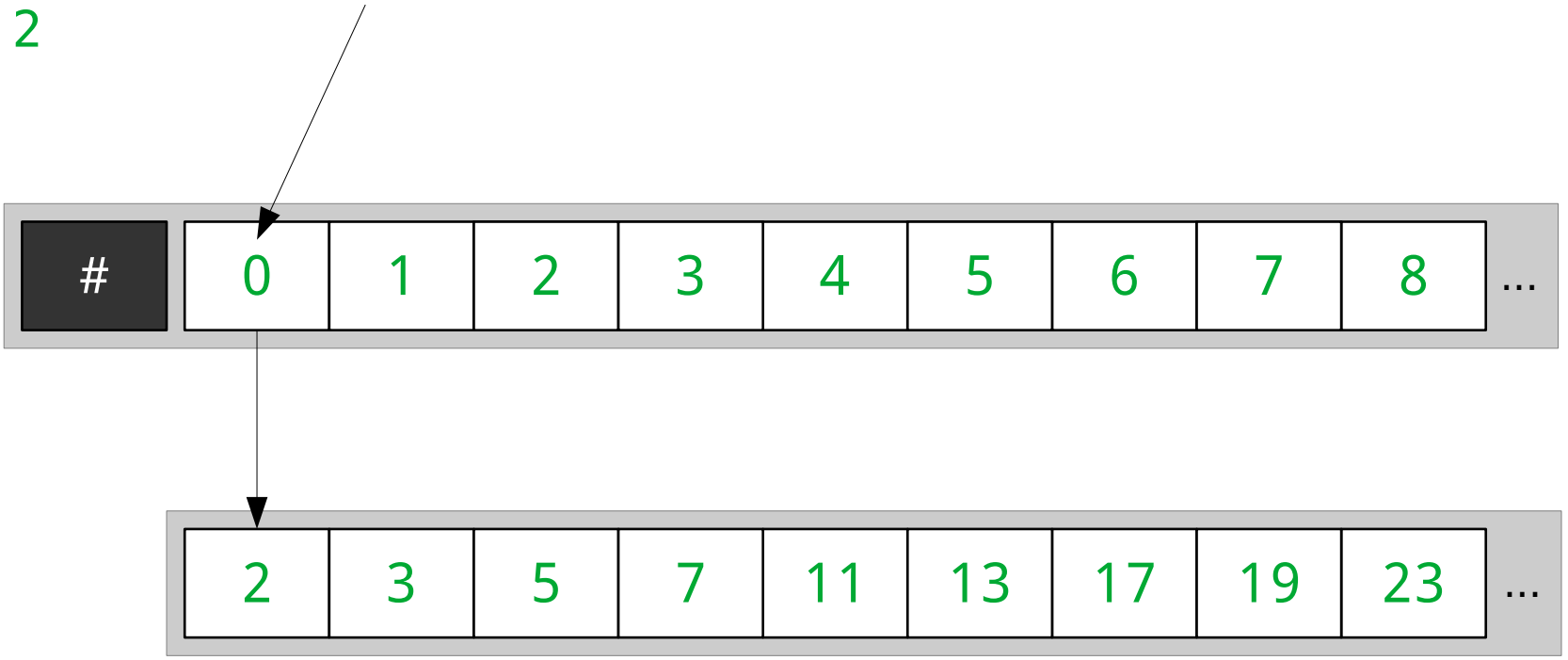
2	3	5	7	11	13	17	19

  - O índice

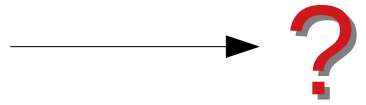
#	0	1	2	3	4	5	6	7

Índices

- A indexação da série utiliza os rótulos do índice
  - `>>> primos[0]`
    `2`



Índices

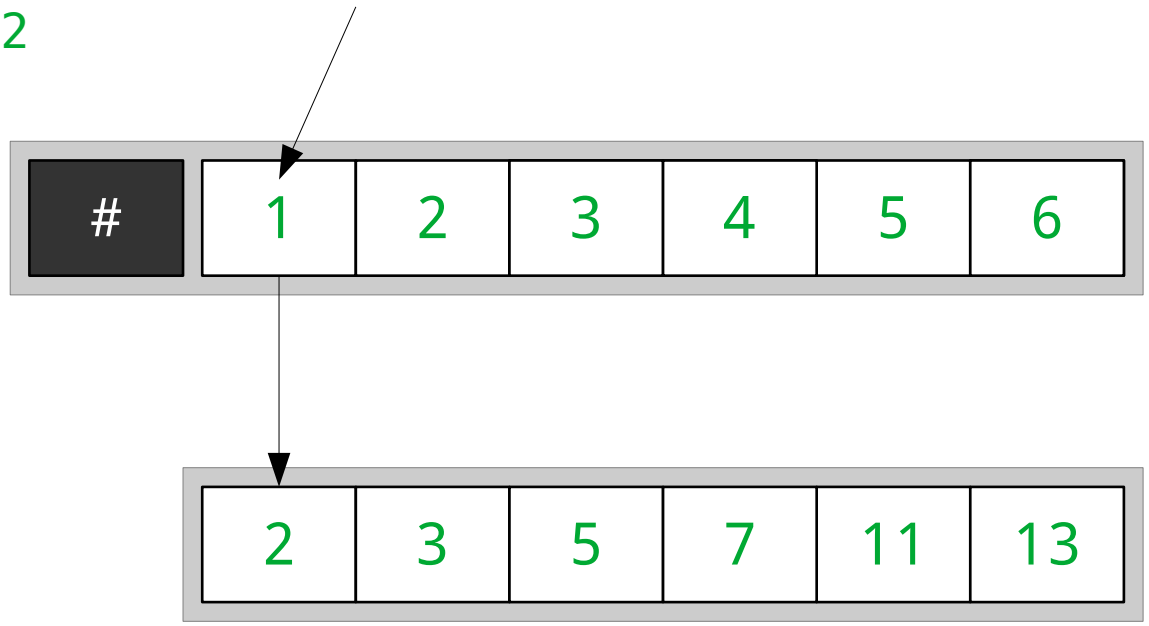
- Nesse caso a indexação tem que ser feita através dos rótulos especificados no índice
  - `>>> primos[0]` 

#	1	2	3	4	5	6

2	3	5	7	11	13

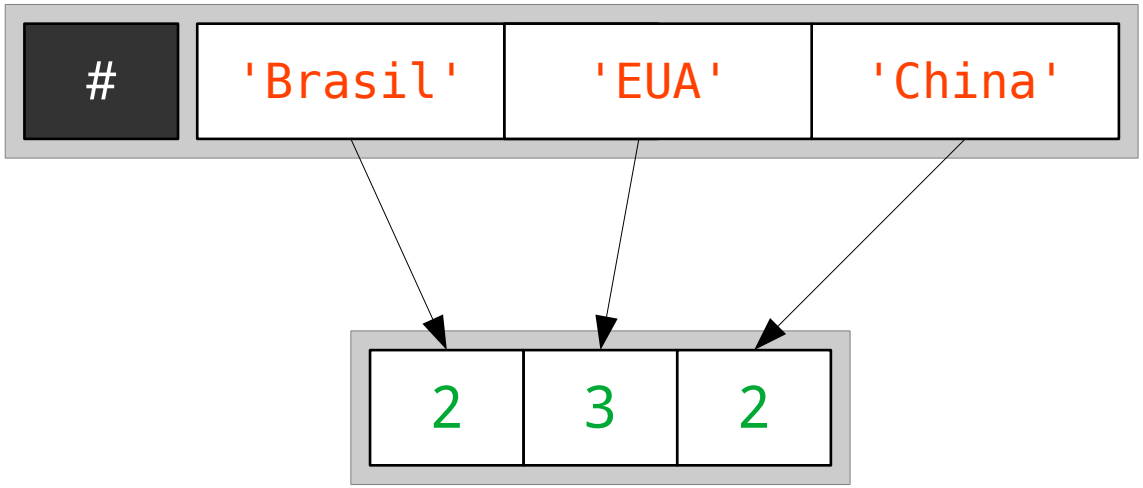
Índices

- Nesse caso a indexação tem que ser feita através dos rótulos especificados no índice
  - `>>> primos[1]`
    `2`



Índices

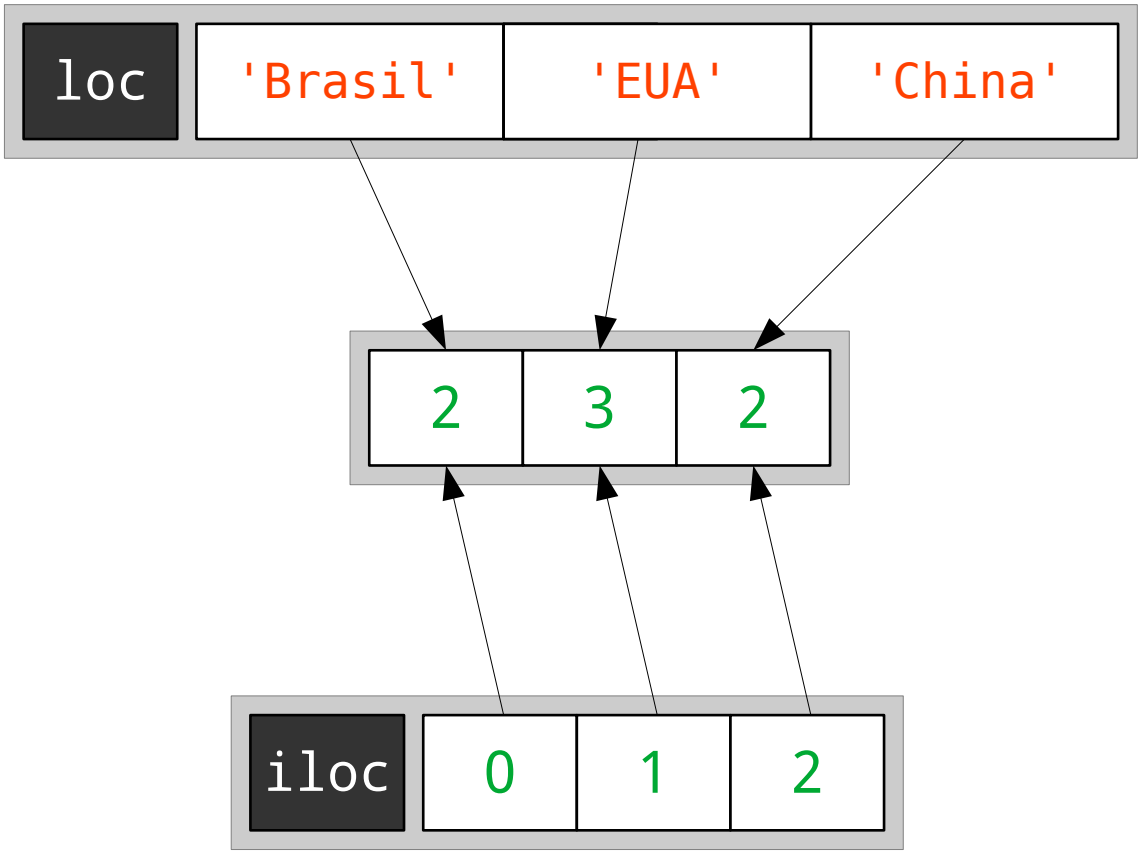
- Os índices não precisam ser números inteiros...
  - Strings podem ser índices
    -
```
>>> idx = ['Brasil', 'EUA', 'China']
>>> val = [2, 3, 2]
>>> dados = pd.Series(val, idx)
```



Índices explícitos

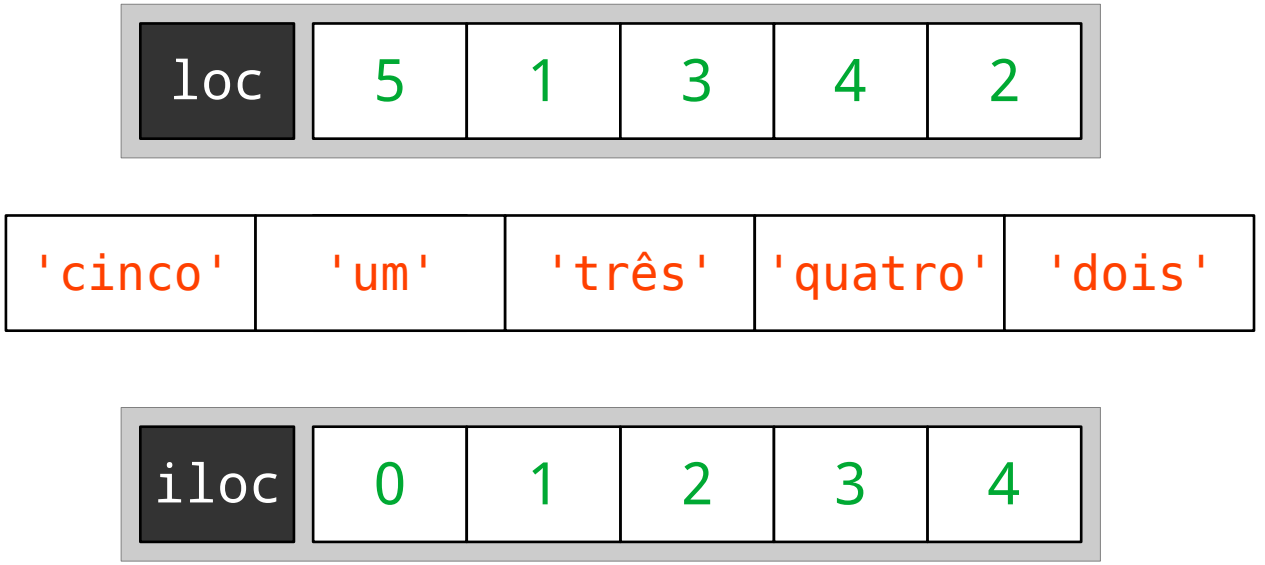
- Os índices do Pandas podem ser acessados diretamente
- Cada objeto da classe Series possui dois índices
  - `Series.loc`: acesso baseado em rótulo
  - `Series.iloc`: acesso posicional
- Utilize os índices explicitamente, quando possível

Índices explícitos



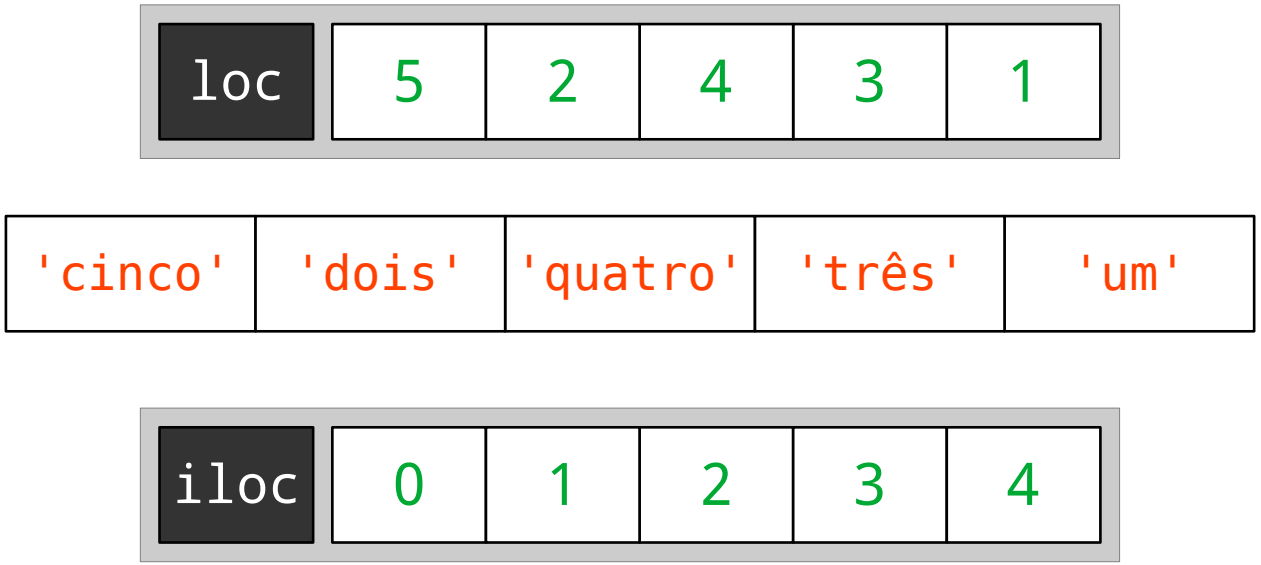
Índices explícitos

- ```
  >>> val = ['cinco', 'um', 'três', 'quatro', 'dois']
  >>> idx = [5, 1, 3, 4, 2]
  >>> numeros = pd.Series(val, index=idx)
  ```


  ```

Índices explícitos

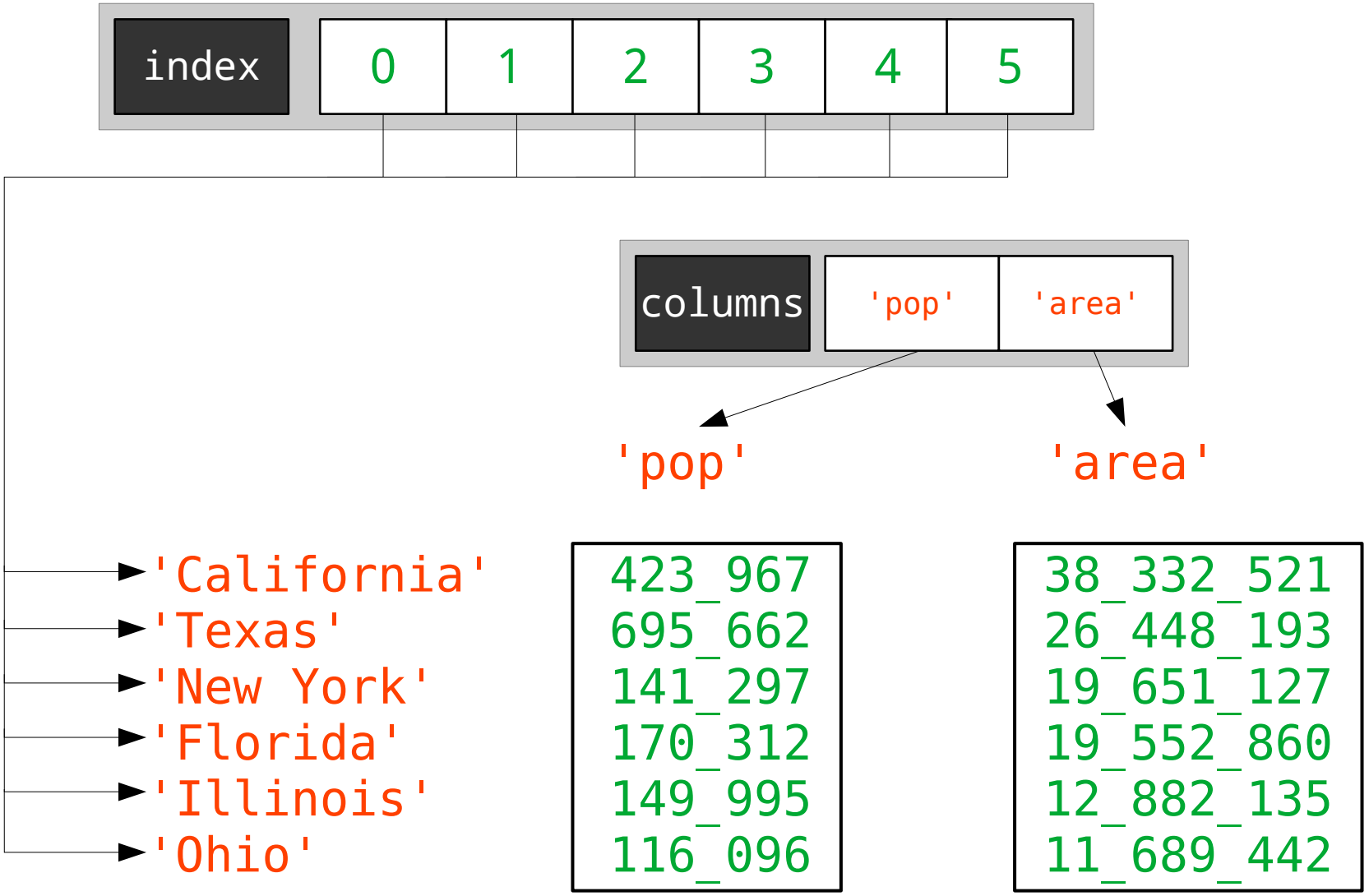
- O `iloc` é **sempre** posicional, então se ordenarmos a série, a relação entre o `iloc` e os elementos pode mudar
  - `>>> numeros.sort_values(inplace=True)`



Data frames

- *Data frames* podem ser encarados como coleções de séries
- Cada coluna pode ser acessada como uma série
- Cada linha também pode ser individualmente acessada como uma série
- Tanto colunas quanto linhas possuem índices

Índices de *data frames*



Agenda

- Introdução ligeira ao Python
- Jupyter *notebooks*
- NumPy
- Pandas
- Demonstração do WEKA
- Regressão linear com Scikit-Learn

O que é Aprendizado de Máquina?

- Você conhece a base de dados *Iris*?
  - É uma das bases de exemplo mais famosas
  - Iris é um gênero de flores

Iris versicolor

Iris virginica

Imagens: https://en.wikipedia.org/wiki/Iris_flower_data_set

O que é Aprendizado de Máquina?

- Você conhece a base de dados *Iris*?
  - A base é uma aplicação de *estatística multivariada*
  - Ela possui dados de 150 flores
    - O comprimento e a largura da sépala
    - O comprimento e a largura da pétala
    - A espécie da flor

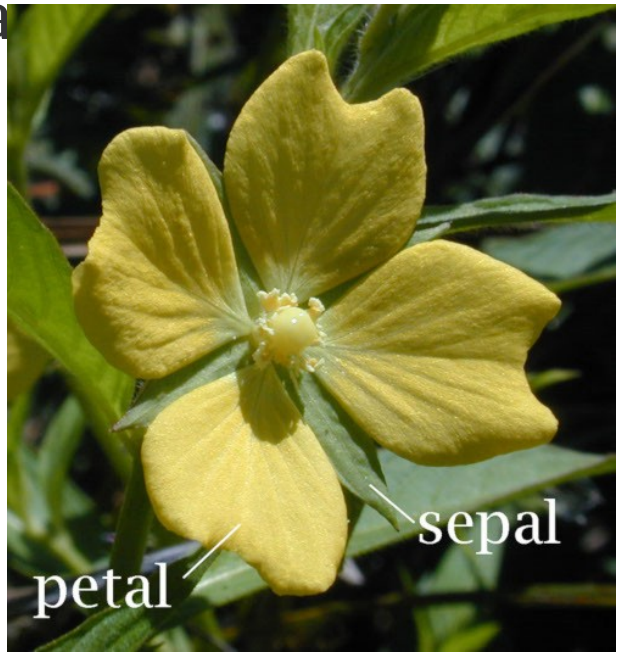
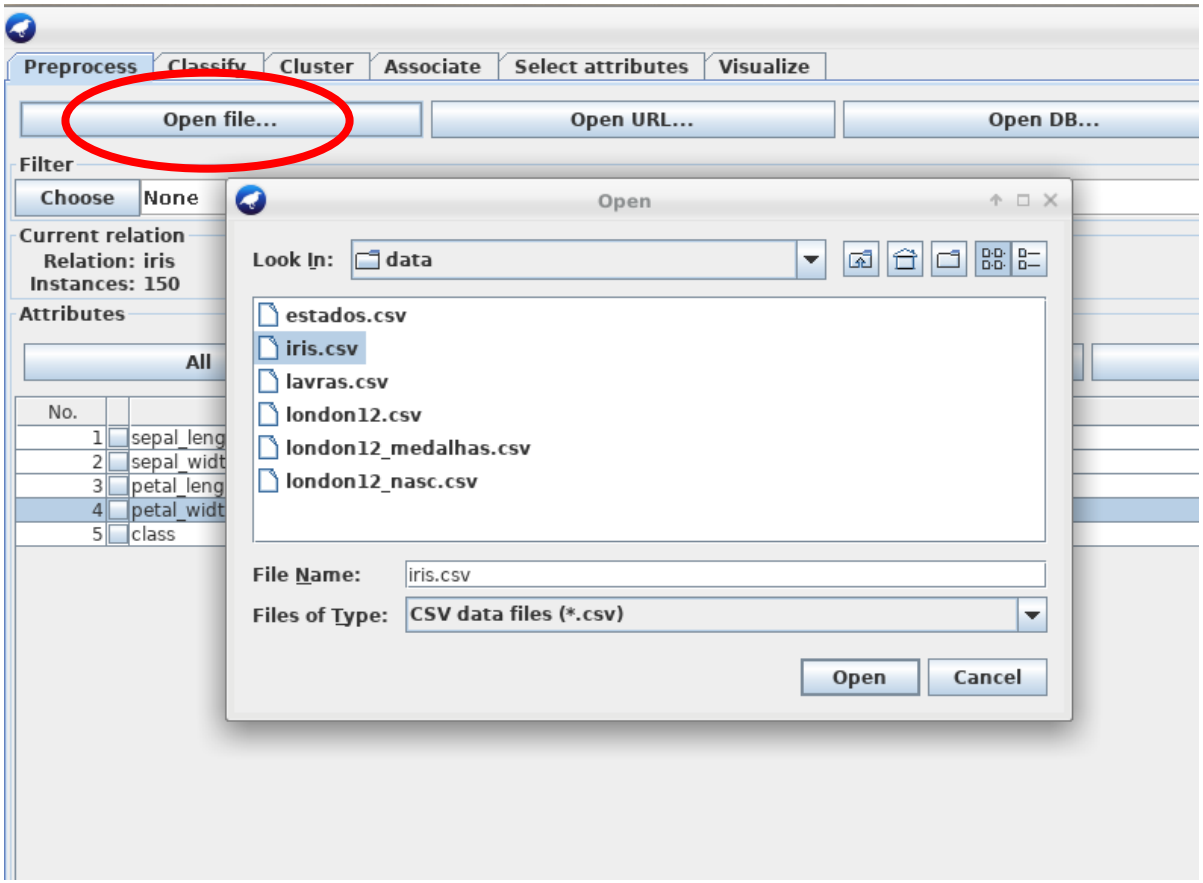


Imagem: https://en.wikipedia.org/wiki/Sepal

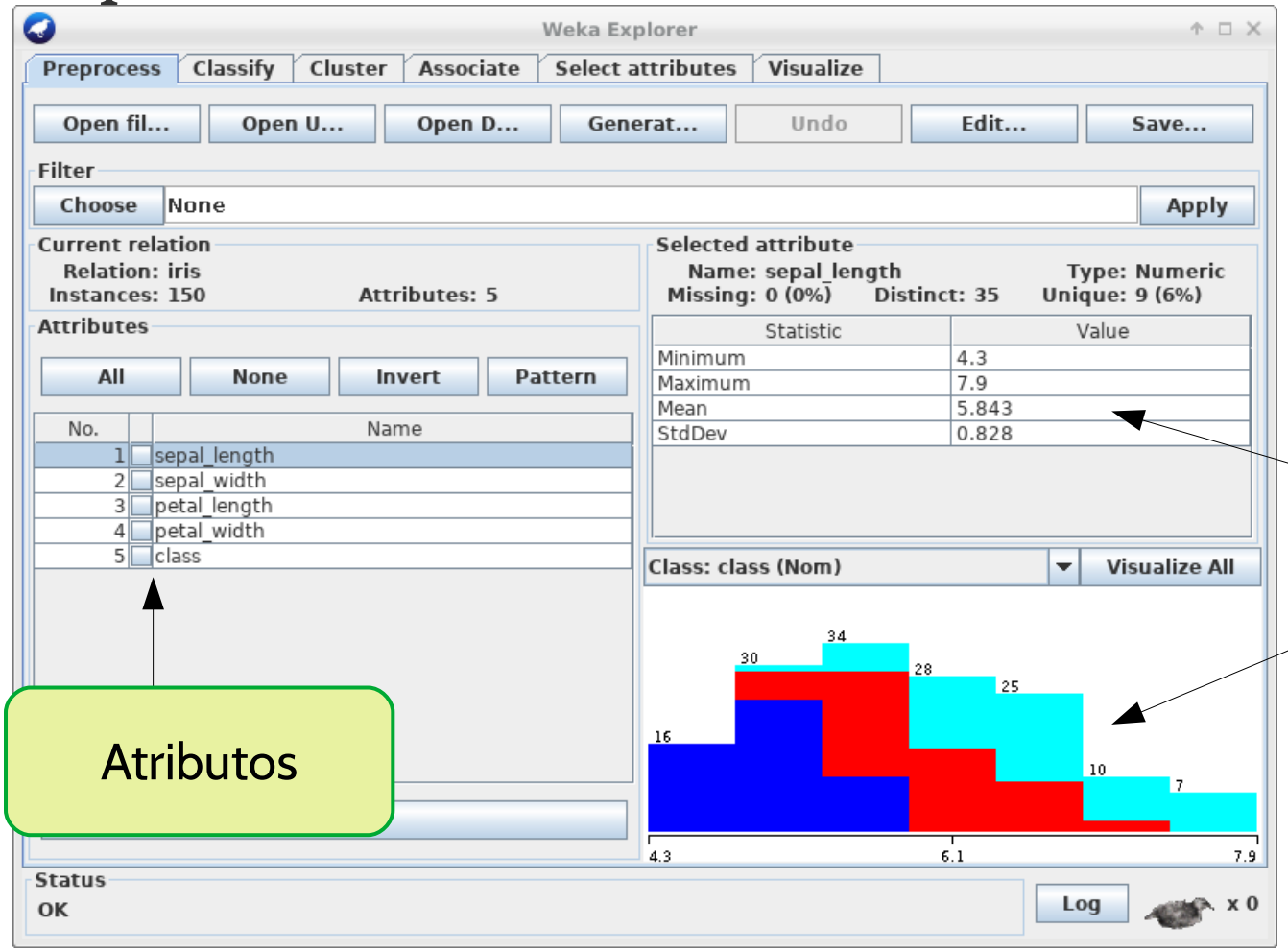
O que é Aprendizado de Máquina?

- Você conhece a base de dados *Iris*?
  - Explore essa base no Weka...



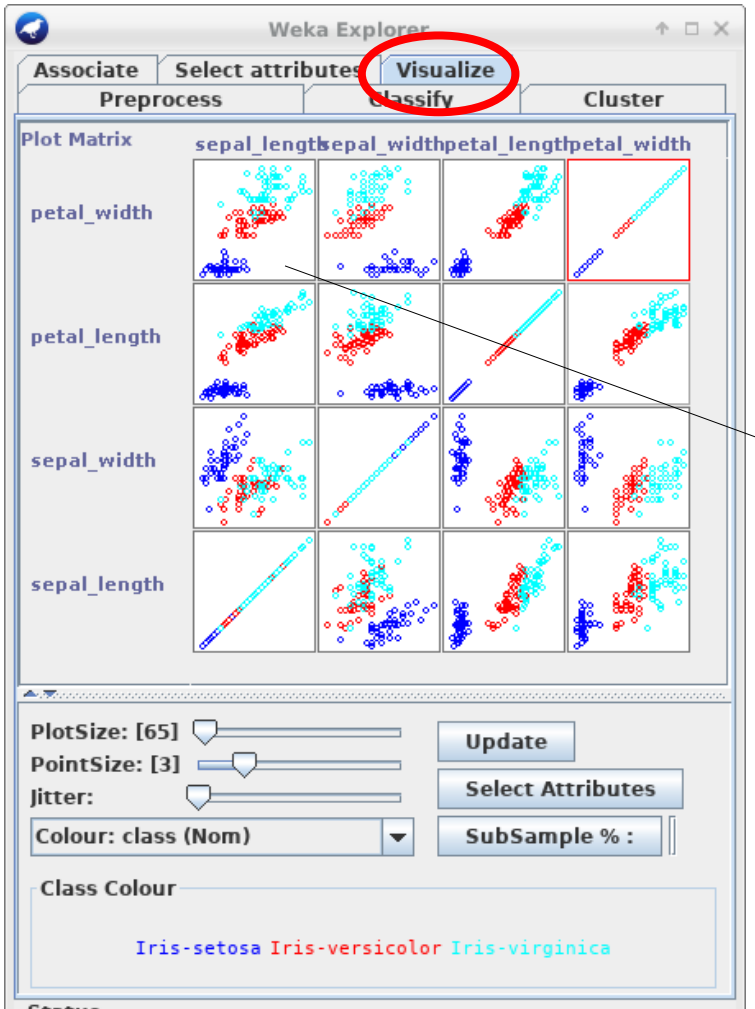
O que é Aprendizado de Máquina?

- Você conhece a base de dados *Iris*?
  - Explore essa base no Weka...

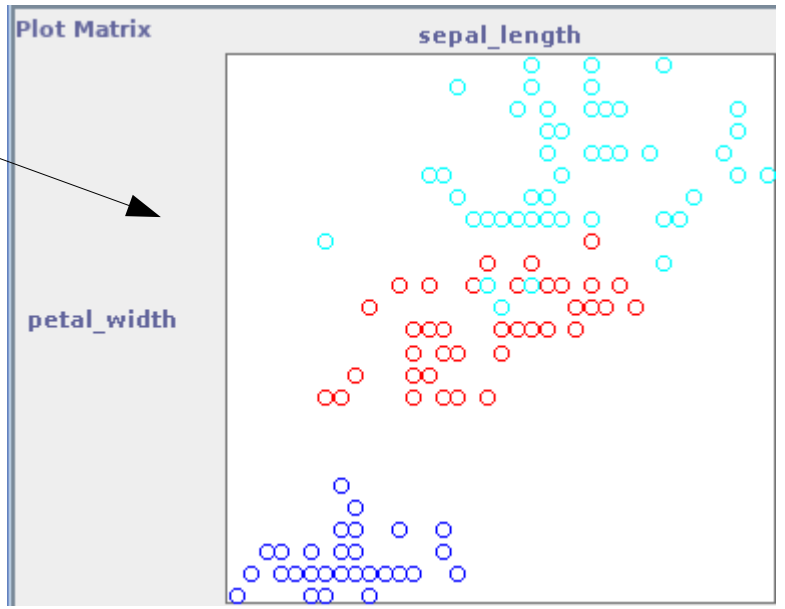


O que é Aprendizado de Máquina?

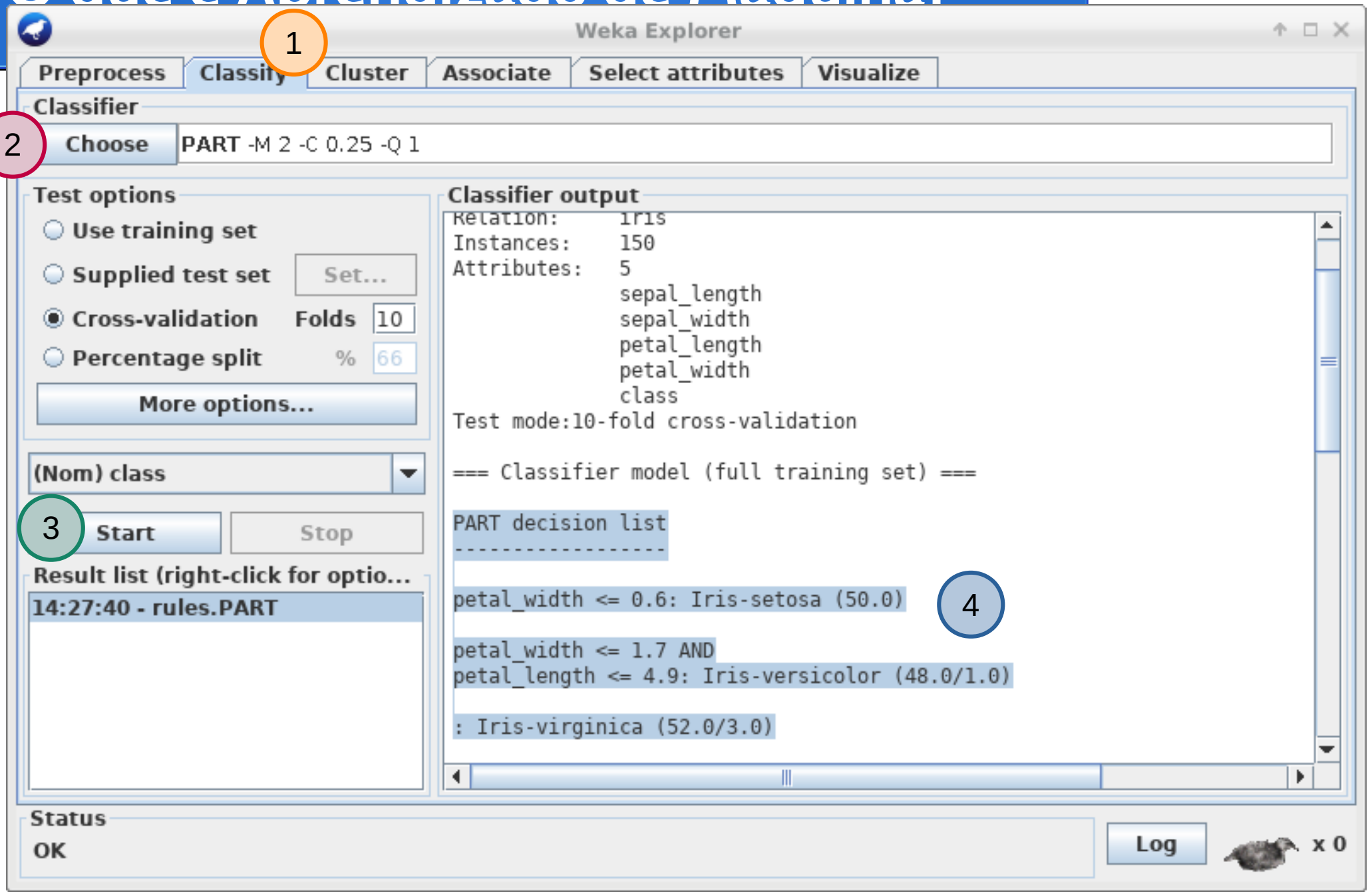
- Você conhece a base de dados *Iris*?
  - Explore essa base no Weka...



Os gráficos de espalhamento permitem visualizar a relação entre os atributos do conjunto



O que é Aprendizado de Máquina?



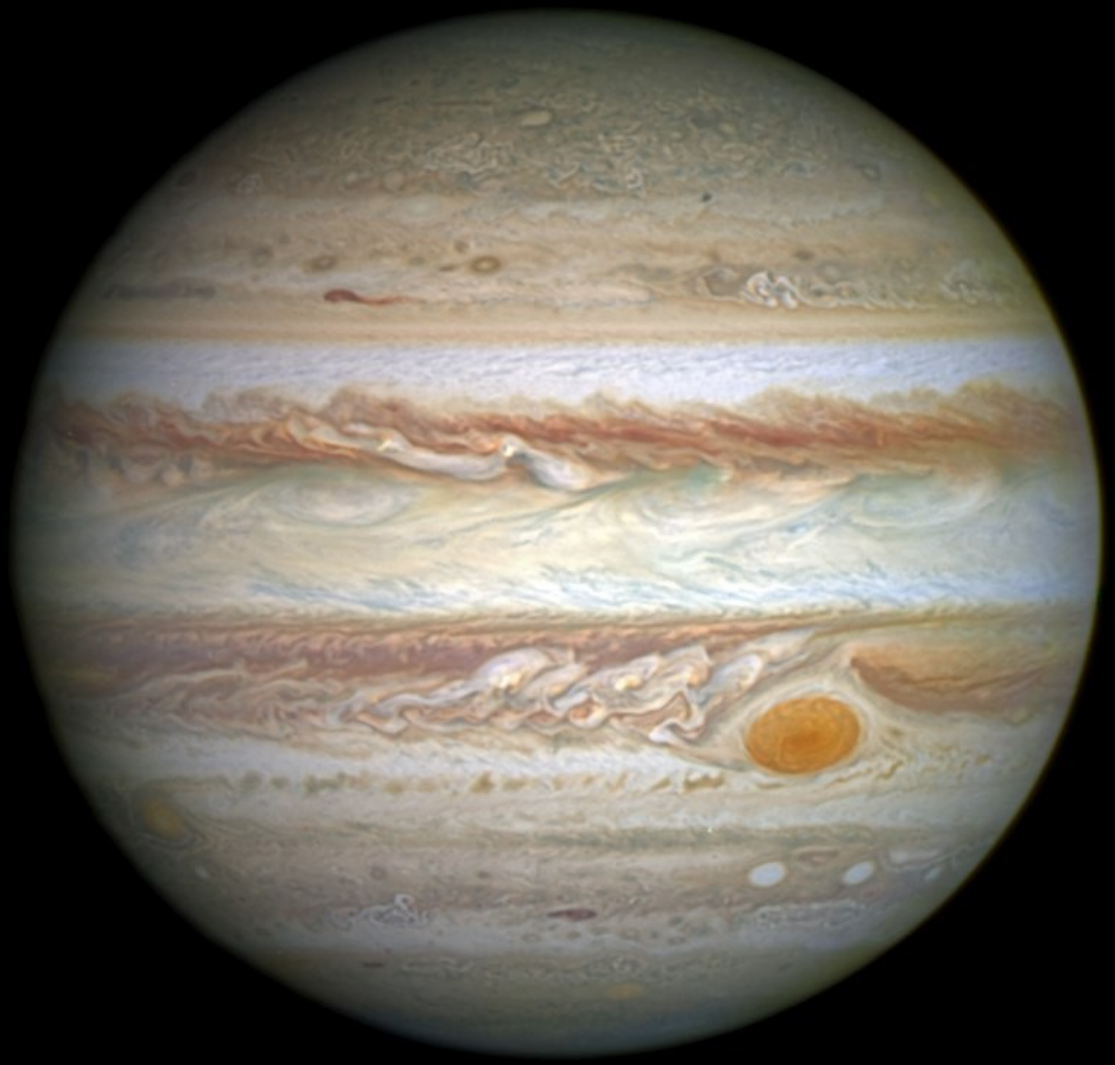
Agenda

- Introdução ligeira ao Python
- Jupyter *notebooks*
- NumPy
- Pandas
- Demonstração do WEKA
- Regressão linear com Scikit-Learn

AM em Python

- Iremos utilizar o Scikit-Learn
- O procedimento geral é (quase sempre) o seguinte
  - Carregar as amostras
  - Escolher uma classe de modelos
  - Importar a classe (Python) que implementa o indutor
    - (Ou as classes dos indutores)
  - Definir hiperparâmetros para configurar o indutor
  - Usar o método `fit` para gerar o modelo adequado aos dados
  - Usar o método `predict` para obter a saída do exemplo em novos dados

Demonstração



https://en.wikipedia.org/wiki/Jupiter#/media/File:Jupiter_and_its_shrunken_Great_Red_Spot.jpg