

Machine Learning com Python

Henry Cagnini



Henry Cagnini

Estudante de doutorado na PUCRS (2017 - presente)

Mestre pela PUCRS (2017)

Bacharel pela UFSM (2014)

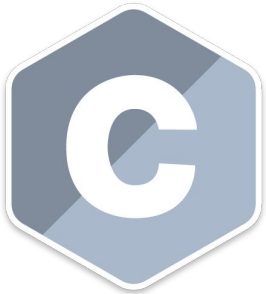
Python

O que é Python?

- Linguagem de programação interpretada
- Foco na legibilidade e (re)usabilidade
- Possui diversas bibliotecas prontas para as mais diversas tarefas

Comparação

Velocidade



Legibilidade

Comparação

C++

```
int a = 1;
```

```
int b = 2;
```

```
int c = a + b;
```

Python

```
a = 1
```

```
b = 2
```

```
c = a + b
```

Comparação

1. Assign <int> 1 to a
2. Assign <int> 2 to b
3. call `binary_add<int, int>(a, b)`
4. Assign the result to c

4 instruções

1. Assign 1 to a
 - a. Set `a->PyObject_Head->typecode` to integer
 - b. Set `a->val = 1`
2. Assign 2 to b
 - a. Set `b->PyObject_HEAD->typecode` to integer
 - b. Set `b->val = 2`
3. call `binary_add(a, b)`
 - a. find typecode in `a->PyObject_HEAD`
 - b. a is an integer; value is `a->val`
 - c. find typecode in `b->PyObject_HEAD`
 - d. b is an integer; value is `b->val`
 - e. call `binary_add<int, int>(a->val, b->val)`
 - f. result of this is result, and is an integer.
4. Create a Python object c
 - a. set `c->PyObject_HEAD->typecode` to integer
 - b. set `c->val` to result

12 instruções

**Como burlar
essa restrição?**

Programando em C

- O nome “completo” da distribuição Python mais popular é **CPython**¹
- Isso porque é possível implementar bibliotecas inteiras em C e chamar funções a partir de uma interface Python

1. As outras podem ser encontradas aqui: <https://www.python.org/download/alternatives/>

Programando em C

Python

```
1. def predict(self, data, dt, inner=False):
2.     tree = dt.tree.node
3.     shape = data.shape
4.     data = data.values.ravel().tolist()
5.
6.     preds = make_predictions(
7.         shape,
8.         data,
9.         tree,
10.         range(shape[0]),
11.         self.dataset_info.attribute_index
12.     )
13.     return preds
```

C

```
1. static PyObject* make_predictions(PyObject *self, PyObject
2.     *args) {
3.     int n_objects, n_attributes;
4.     PyObject *predictions, *tree, *dataset, *attribute_index,
5.         *shape;
6.
7.     if (!PyArg_ParseTuple(
8.         args, "O!O!O!O!O!",
9.         &PyTuple_Type, &shape,
10.        &PyList_Type, &dataset,
11.        &PyDict_Type, &tree,
12.        &PyList_Type, &predictions,
13.        &PyDict_Type, &attribute_index)) {
14.         return NULL;
15.     }
16.
17.     n_objects = (int)PyInt_AsLong(PyTuple_GetItem(shape, 0));
18.     n_attributes = (int)PyInt_AsLong(PyTuple_GetItem(shape,
19.         1));
20.
21.     predict_dataset(n_objects, n_attributes, &dataset[0], tree,
22.         &predictions[0], attribute_index);
23.
24.     return Py_BuildValue("O", predictions);
25. }
```

Programando em C

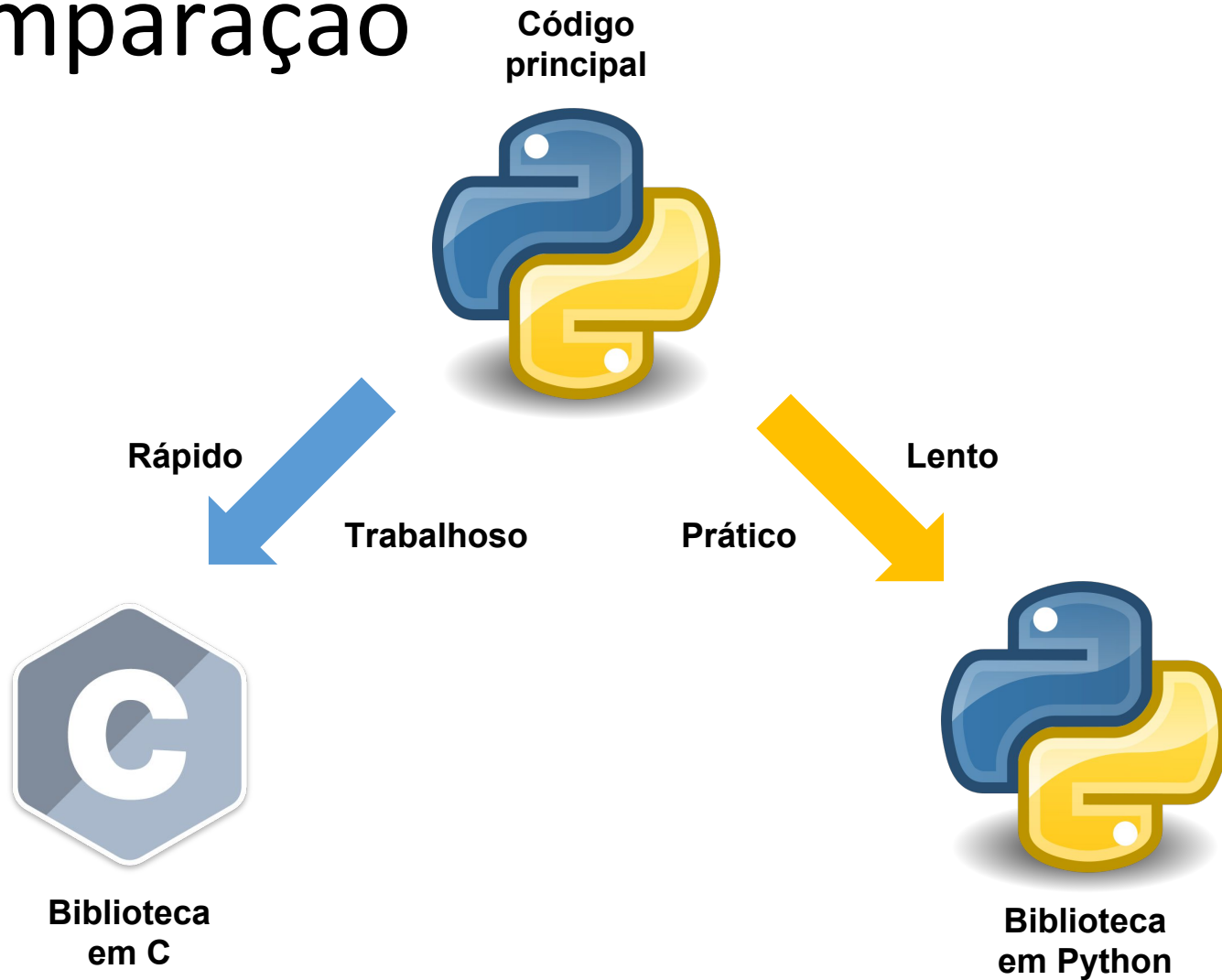
Python

```
1. def predict(self, data, dt, inner=False):
2.     tree = dt.tree.node
3.     shape = data.shape
4.     data = data.values.ravel().tolist()
5.
6.     preds = make_predictions(
7.         shape,
8.         data,
9.         tree,
10.         range(shape[0]),
11.         self.dataset_info.attribute_index
12.     )
13.     return preds
```

C

```
1. static PyObject* make_predictions(PyObject *self, PyObject
2. *args) {
3.     int n_objects, n_attributes;
4.     PyObject *predictions, *tree, *dataset, *attribute_index,
5.     *shape;
6.
7.     if (!PyArg_ParseTuple(
8.         args, "O!O!O!O!O!",
9.         &PyTuple_Type, &shape,
10.        &PyList_Type, &dataset,
11.        &PyDict_Type, &tree,
12.        &PyList_Type, &predictions,
13.        &PyDict_Type, &attribute_index)) {
14.         return NULL;
15.     }
16.
17.     n_objects = (int)PyInt_AsLong(PyTuple_GetItem(shape, 0));
18.     n_attributes = (int)PyInt_AsLong(PyTuple_GetItem(shape,
19. 1));
20.
21.     predict_dataset(n_objects, n_attributes, &dataset[0], tree,
22.     &predictions[0], attribute_index);
23.
24.     return Py_BuildValue("O", predictions);
25. }
```

Comparação



Python para Aprendizado de máquina

Aprendizado de Máquina

A maioria das bibliotecas em Python é escrita em C:

- **scikit-learn** (aprendizado de máquina)
- **pandas** (banco de dados relacional)
- **numpy** (operações matriciais)

Aprendizado de Máquina

1 Sklearn	+ R Packages										* Julia Packages										2 Weka
3 Mahout	4 Torch	Big Data		Lua/JS/Clojure		Computer Vision NLP Libraries		C/C++		5 Orange		8 SM	7 PyMC	8 Theano	9 Factorio	10 Mallet					
11 Conj	12 April	R/Julia		Java		Scala		Python		13 VW		14 PyStan	15 Hebel	16 MDP	17 BIDM	18 Knime					
19 SAMOA	20 ConvJS	21 OpenC	22 Simple	23 CCV	24 VLFeat	25 PIL	26 ImageJ	27 rimage	28 biOps	29 ITK	30 VTK	31 Shogun	32 LibSVM	33 PyB	34 Pylearn	35 Figaro	36 RM				
37 Oryx	38 jsLDA	39 OpenC	40 DLib	41 Eblearn	42 Tesseract	43 MITIE	44 CoreNLP	45 cTAKES	46 OpenN	47 LingPipe	48 ClearT	49 Shark	50 Jubatus	51 VFML	52 MLPy	53 Wolfe	54 Encog				
55 MLlib	56 ml	57 natural	72 Knwl.js	73 Retext	74 NLTK	75 Pattern	76 Quepy	77 TextBlob	78 YAlign	79 Treat	80 ScalaN	81 Caffe	82 OF	83 MLPac	84 MB	85 Predict	86 ELKI				
87 MLBas	88 clj-ml	89 Breeze	104 Chalk	105 UIMA	106 GATE	107 Frame	108 clojure-nlp	109 tmt	110 RiTa	111 nlp_c	112 twitter-text	113 MLC	114 Darwin	115 kml	116 FBM	117 OptiML	118 DL4J				
+		58 arules	59 rf	60 glmnet	61 rpart	62 e1071	63 gbm	64 tgp	65 caret	66 frbs	67 mlr	68 nnet	69 tree	70 bst	71 ipred						
*		90 MCMC	91 GLM	92 kde	93 NMF	94 MLBas	95 Clust	96 SVM	97 RERM	98 MStat	99 DR	100 Mocha	101 MM	102 ANN	103 DA						

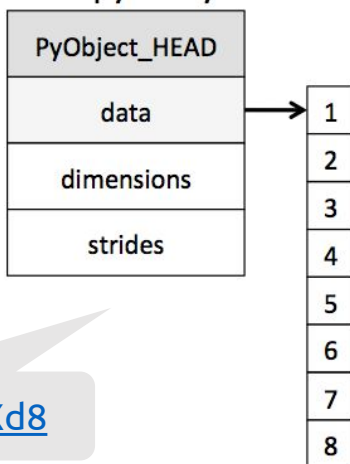
NumPy

*“A biblioteca fundamental para
computação científica em Python”*

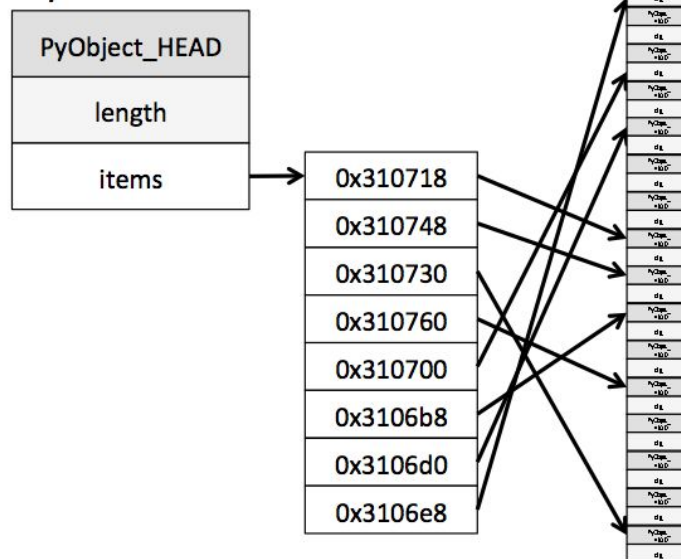
Numpy arrays

- São como **arrays** em C++/Java
- Alocam espaços **contíguos** em memória
- Utilizados em funções **numpy**

Numpy Array



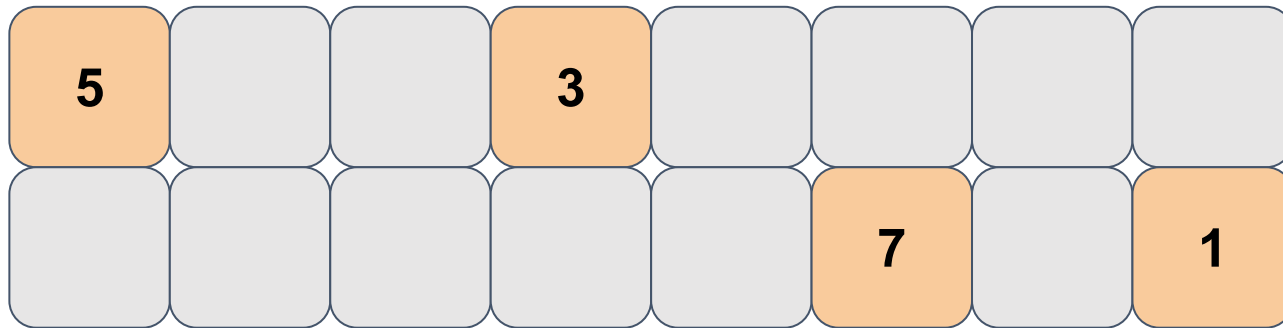
Python List



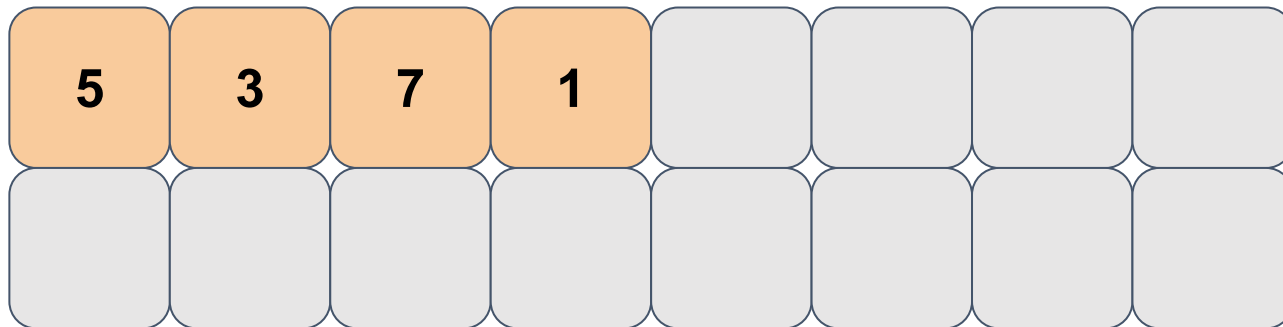
<https://goo.gl/K2LXd8>

Numpy arrays

List



Numpy
Array



Numpy arrays

In: `import timeit as ti`

In: `ti.timeit(stmt="[x**2 for x in xrange(10000)]",
number=100)`

Out: 0.0475

In: `ti.timeit(stmt="map(lambda x: x**2, xrange(10000))",
number=100)`

Out: 0.0973

In: `ti.timeit(setup="import numpy", stmt="a =
numpy.arange(10000); a**2", number=100)`

Out: 0.00113

Numpy arrays

Por quê?

- Além de usar dados **contíguos** em memória, as **funções numpy** usam a biblioteca BLAS (Basic Linear Algebra Subprograms):
 - São subrotinas para realizar cálculos matemáticos e matriciais, disponíveis para CPUs e GPUs
 - MATLAB e Octave também utilizam BLAS

pandas

pandas

- Biblioteca para manipulação de dados (tabelados) em memória principal
- Possui diversas funções de um banco de dados relacional
 - selecionar, agregar, ordenar, etc
- Permite dados faltantes, intervalos temporais, conversão de dados

scikit-learn

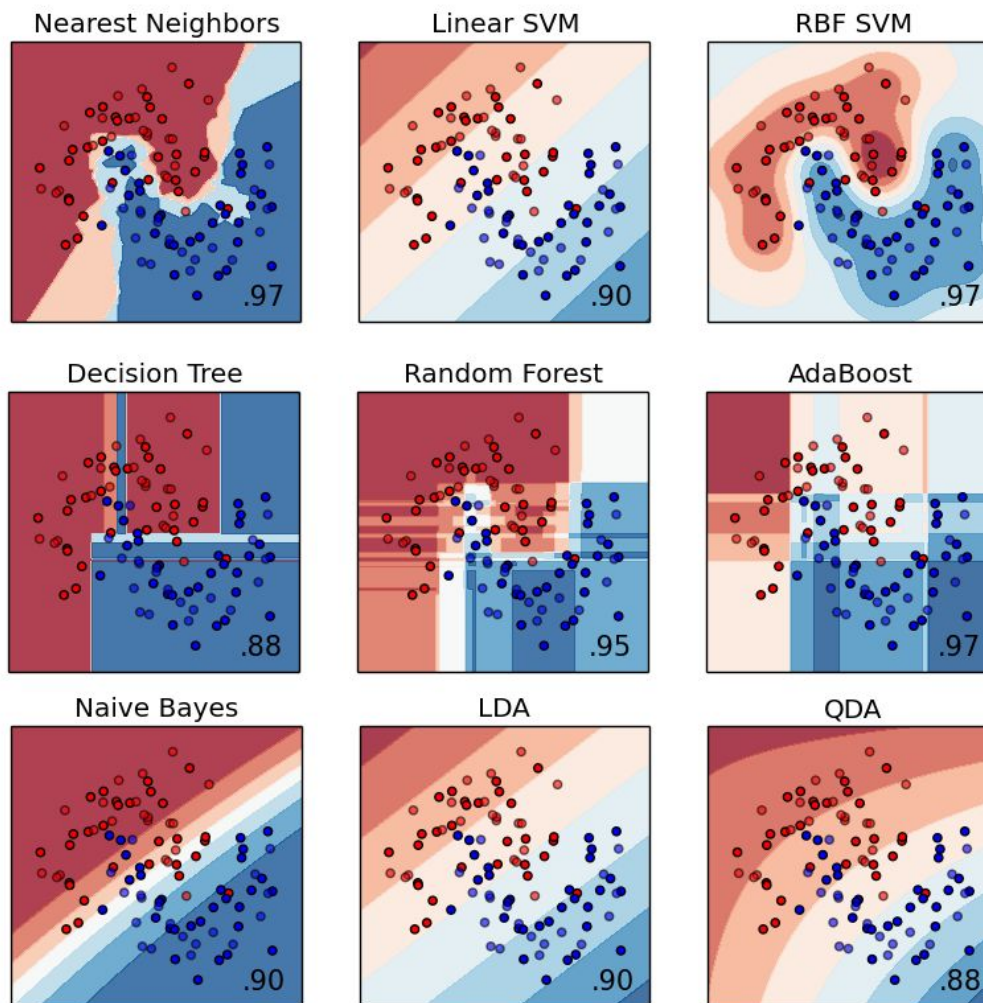
*“Ferramentas simples e eficientes
para mineração e análise de dados”*

scikit-learn

Biblioteca para realizar tarefas de mineração de dados e aprendizado de máquina:

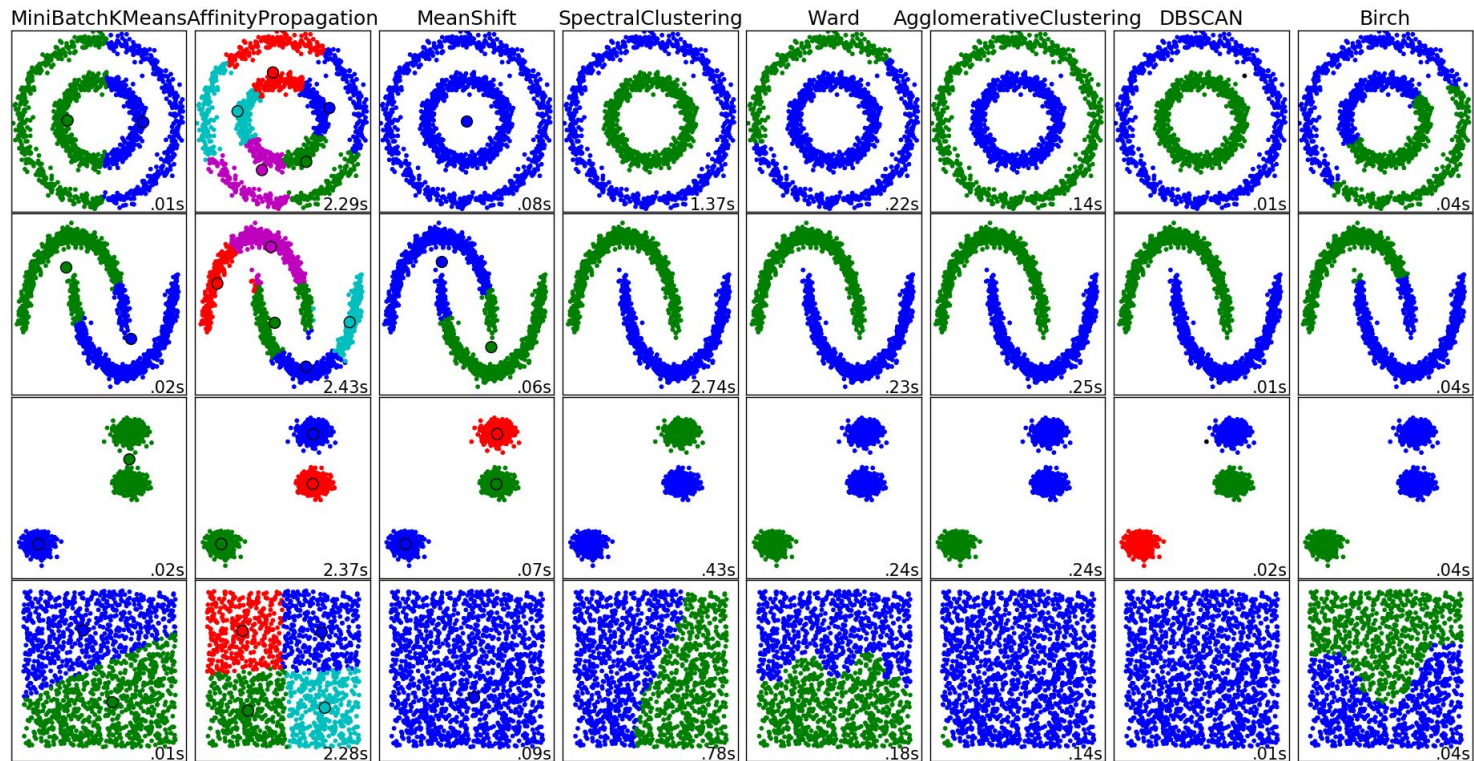
- Aprendizado supervisionado e não supervisionado
- Pré-processamento de dados
- Redução de dimensionalidade
- Validação de modelos

scikit-learn: classificação



<https://goo.gl/4kftaY>

scikit-learn: agrupamento



<https://goo.gl/ZolhJb>

Muito obrigado!

Dúvidas?