# Machine Learning com Python

Henry Cagnini

# Henry Cagnini

Estudante de doutorado na PUCRS (2017 - presente)

Mestre pela PUCRS (2017)

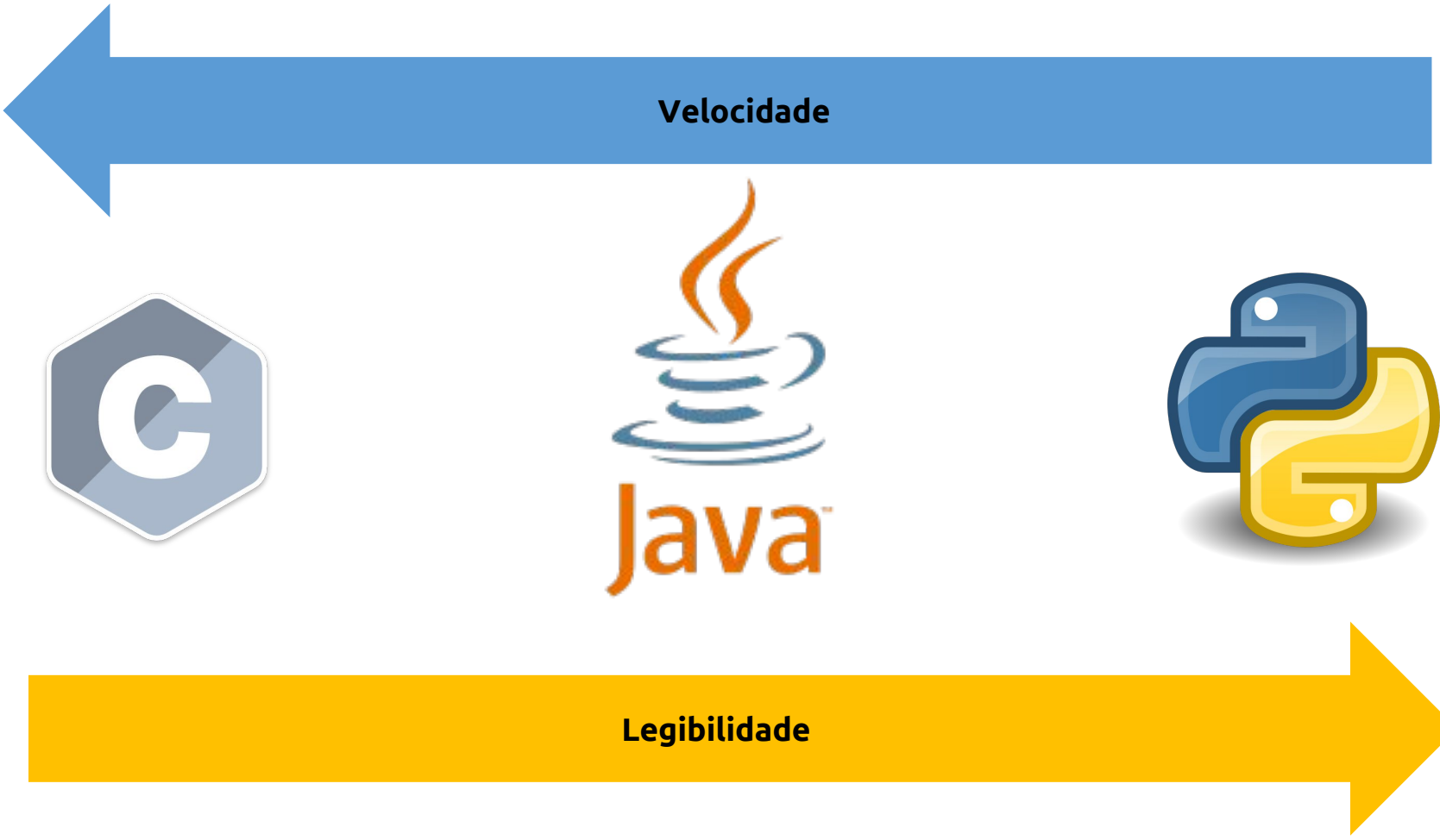
Bacharel pela UFSM (2014)

# 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



# 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. Creat a Python object c
   a. set c->PyObject_HEAD->typecode to integer
   b. set c->val to result

12 instruções

https://jakevdp.github.io/blog/2014/05/09/why-python-is-slow/

# 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**

```python
def predict(self, data, dt, inner=False):
    tree = dt.tree.node
    shape = data.shape
    data = data.values.ravel().tolist()

    preds = make_predictions(
        shape,
        data,
        tree,
        range(shape[0]),
        self.dataset_info.attribute_index
    )
    return preds
```

**C**

```c
static PyObject* make_predictions(PyObject *self, PyObject *args) {
    int n_objects, n_attributes;
    PyObject *predictions, *tree, *dataset, *attribute_index, *shape;

    if (!PyArg_ParseTuple(
            args, "O!O!O!O!O!",
            &PyTuple_Type, &shape,
            &PyList_Type, &dataset,
            &PyDict_Type, &tree,
            &PyList_Type, &predictions,
            &PyDict_Type, &attribute_index)) {
        return NULL;
    }

    n_objects = (int)PyInt_AsLong(PyTuple_GetItem(shape, 0));
    n_attributes = (int)PyInt_AsLong(PyTuple_GetItem(shape, 1));

    predict_dataset(n_objects, n_attributes, &dataset[0], tree, &predictions[0], attribute_index);

    return Py_BuildValue("O", predictions);
}
```

# Programando em C

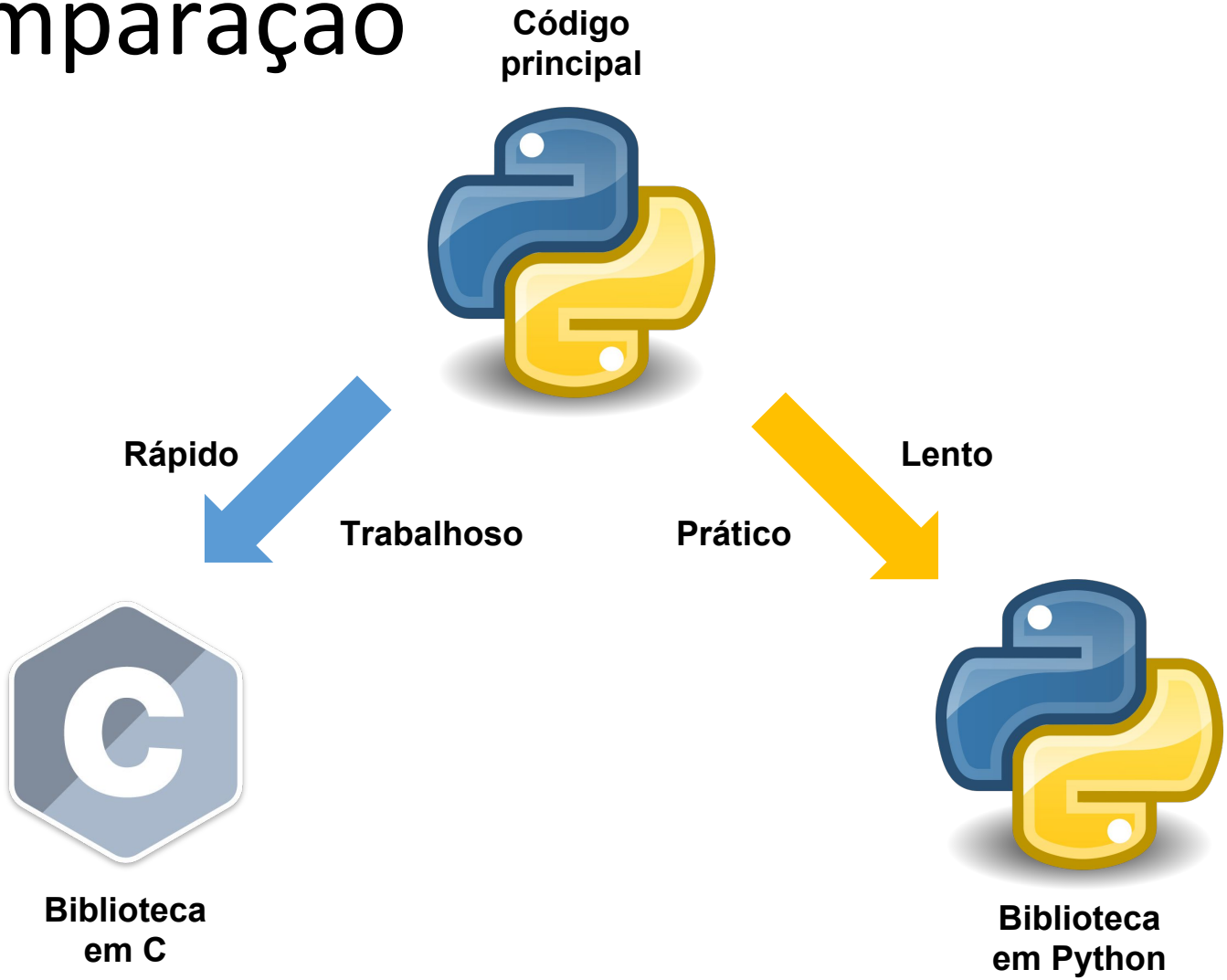
**Python**

```python
def predict(self, data, dt, inner=False):
    tree = dt.tree.node
    shape = data.shape
    data = data.values.ravel().tolist()

    preds = make_predictions(
        shape,
        data,
        tree,
        range(shape[0]),
        self.dataset_info.attribute_index
    )
    return preds
```

**C**

```c
static PyObject* make_predictions(PyObject *self, PyObject *args) {
    int n_objects, n_attributes;
    PyObject *predictions, *tree, *dataset, *attribute_index, *shape;

    if (!PyArg_ParseTuple(
            args, "O!O!O!O!O!",
            &PyTuple_Type, &shape,
            &PyList_Type, &dataset,
            &PyDict_Type, &tree,
            &PyList_Type, &predictions,
            &PyDict_Type, &attribute_index)) {
        return NULL;
    }

    n_objects = (int)PyInt_AsLong(PyTuple_GetItem(shape, 0));
    n_attributes = (int)PyInt_AsLong(PyTuple_GetItem(shape, 1));

    predict_dataset(n_objects, n_attributes, &dataset[0], tree, &predictions[0], attribute_index);

    return Py_BuildValue("O", predictions);
}
```

# Comparação


Código principal
Rápido
Trabalhoso
Lento
Prático
Biblioteca em C
Biblioteca em Python

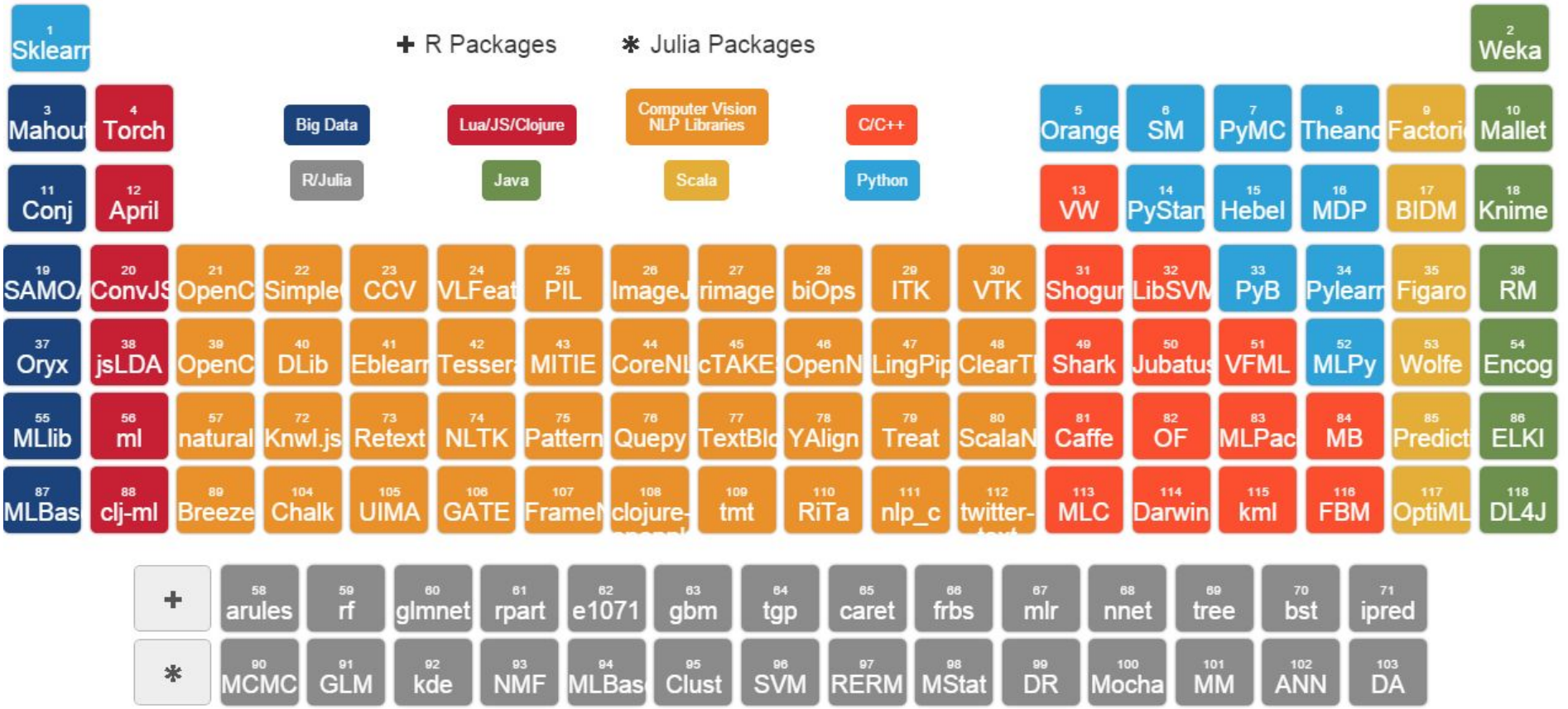
# Python para Aprendizado de máquina

# 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



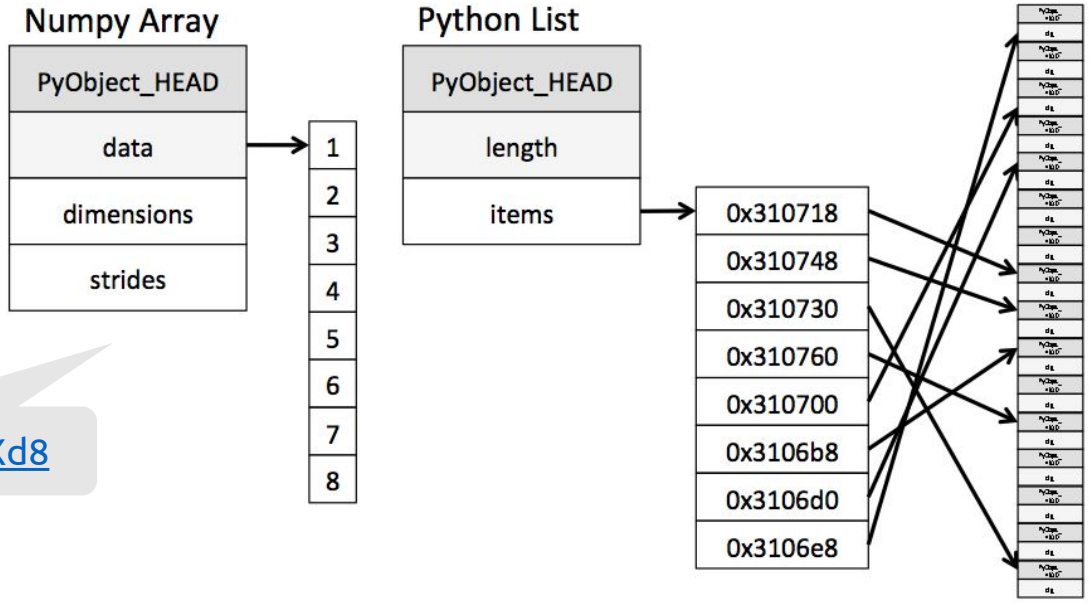

http://www.mln.io/resources/periodic-table/

# NumPy

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

# Numpy arrays

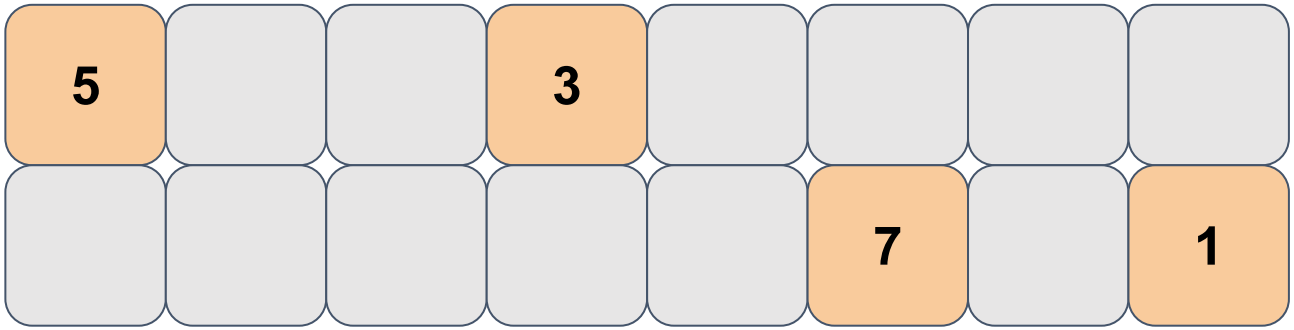
- São como **arrays** em C++/Java
- Alocam espaços **contíguos** em memória
- Utilizados em funções **numpy**



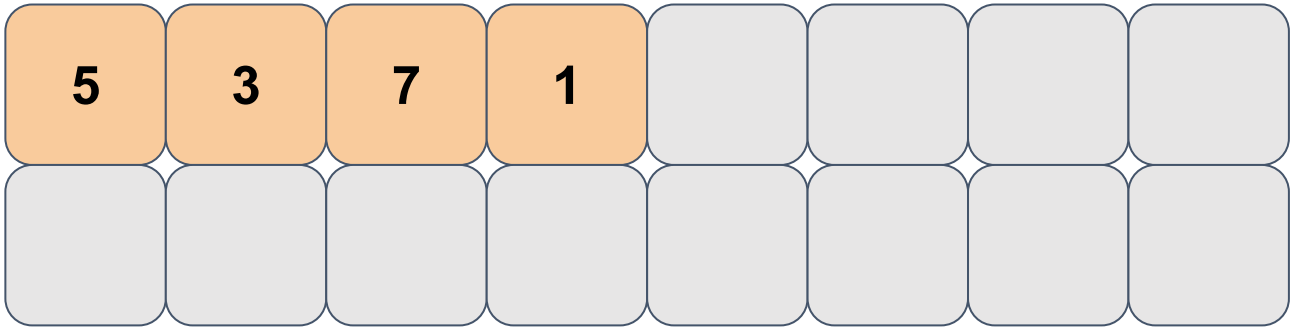

https://goo.gl/K2LXd8

# 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

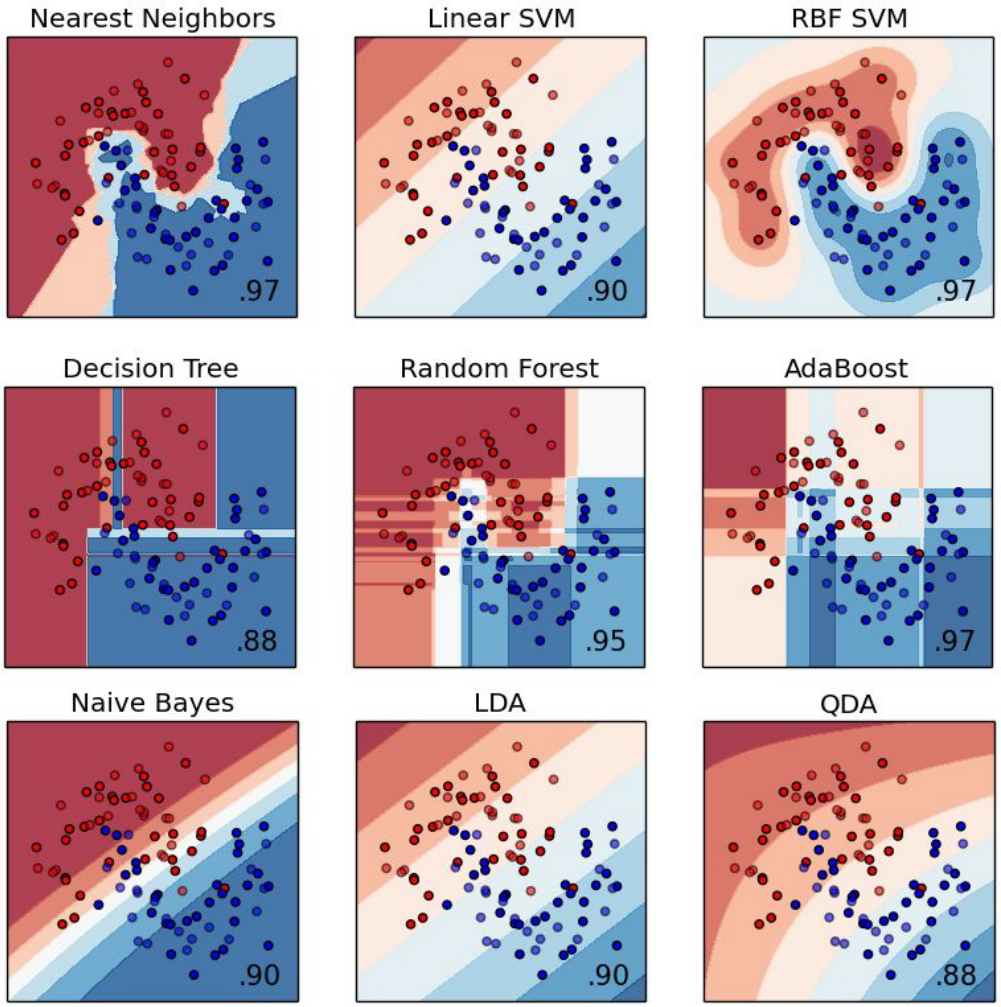
*“Ferramentas simples e eficientes para mineração e análise de dados”*

# scikit-learn

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

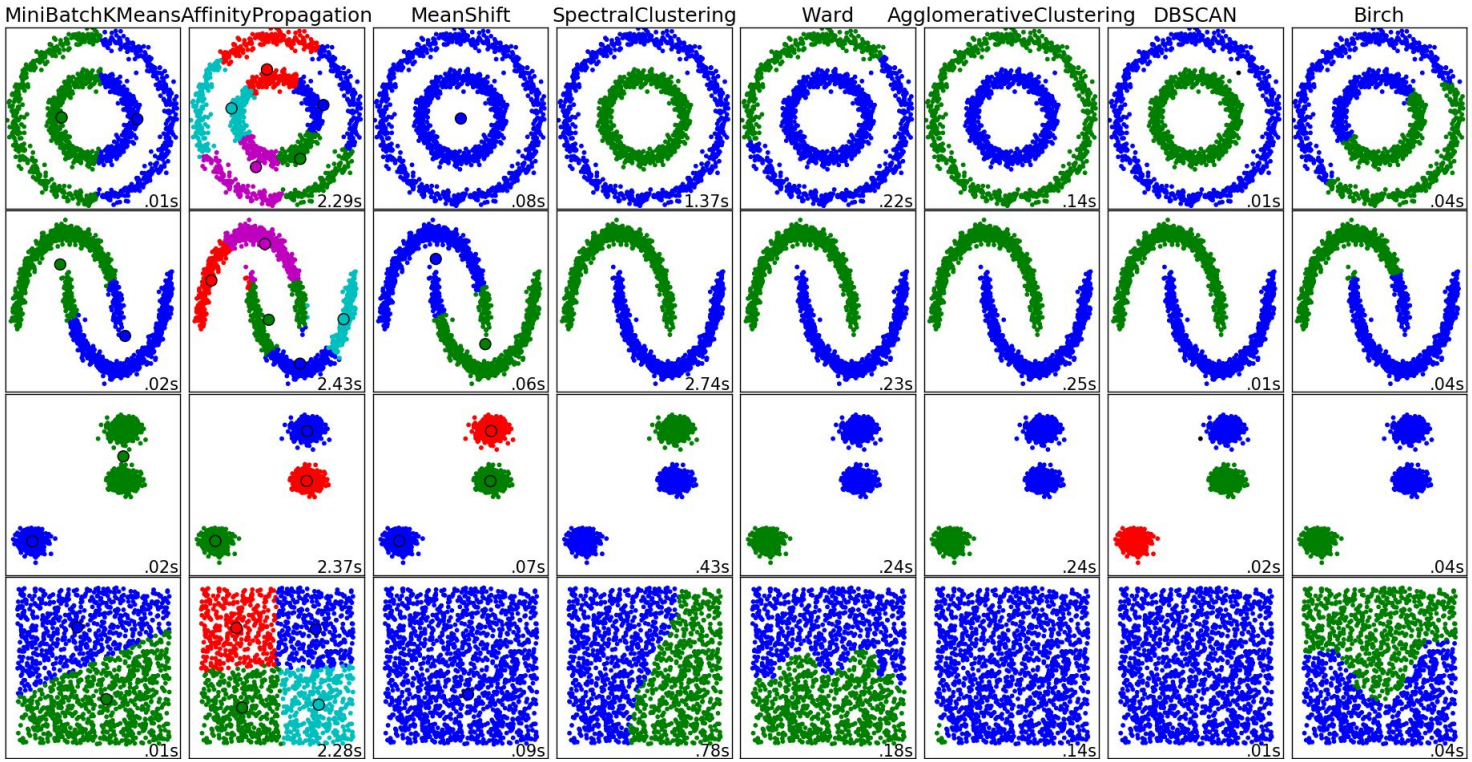
- Aprendizado supervisionado e não supervisionado
- Pré-processamento de dados
- Redução de dimensionalidade
- Validação de modelos

# scikit-learn: classificação




https://goo.gl/4kftaY

# scikit-learn: agrupamento




https://goo.gl/ZoIhJb

# Muito obrigado!

Dúvidas?