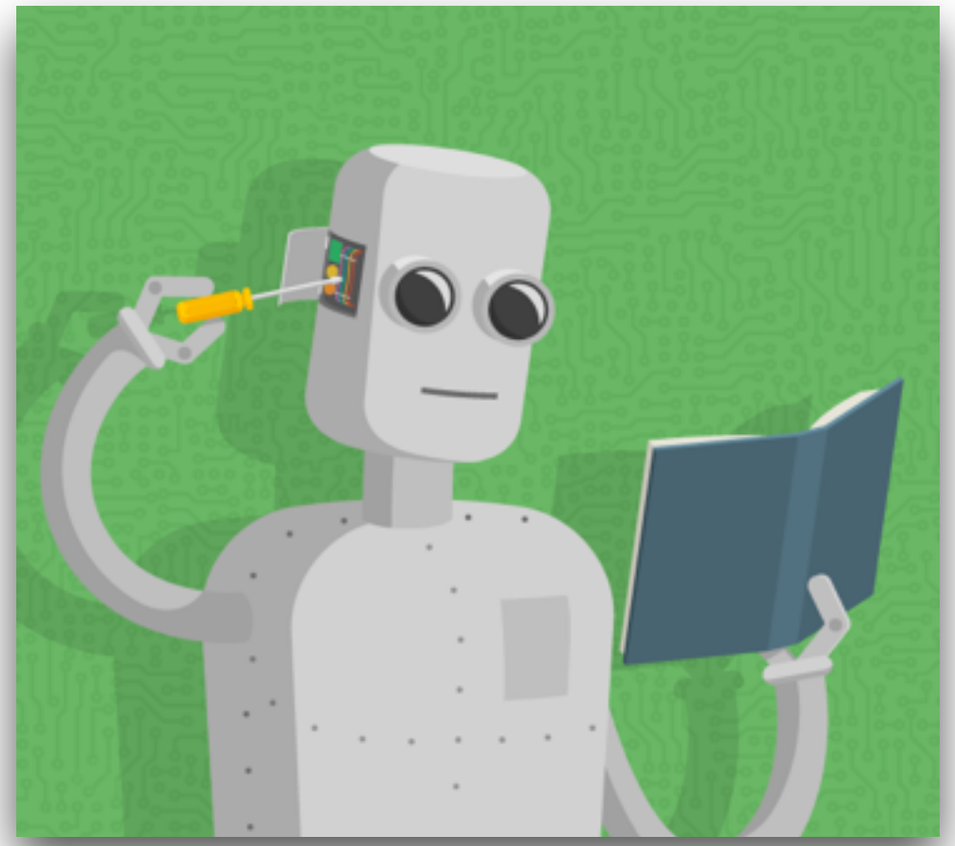
Disciplina de Inteligência Artificial (Aula 4)



Classificação com árvores de Decisão e Classificação Linear

Apresentação

Francisco Nauber Bernardo Gois

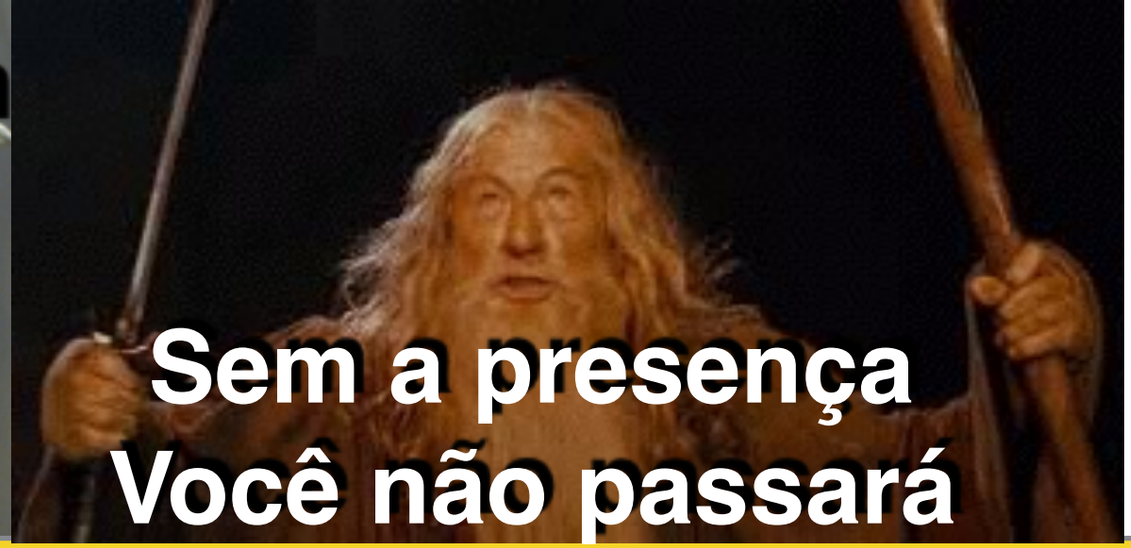
Analista aprendizado de máquina no Serviço Federal de Processamento de Dados

Doutorando em Informática Aplicada

Mestre em Informática Aplicada

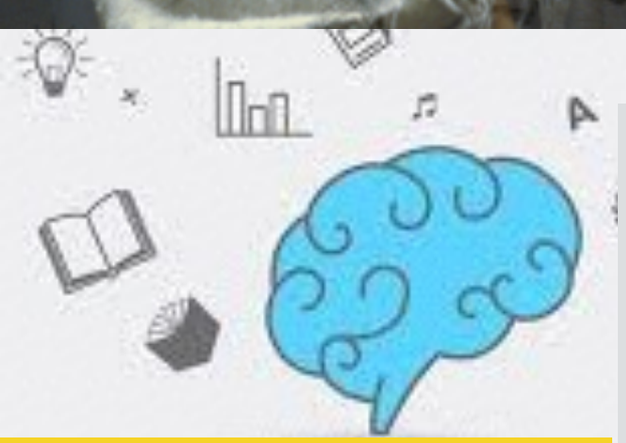
Especialista em desenvolvimento WEB

Dúvidas: naubergois@gmail.com



Para melhor desempenho na aula

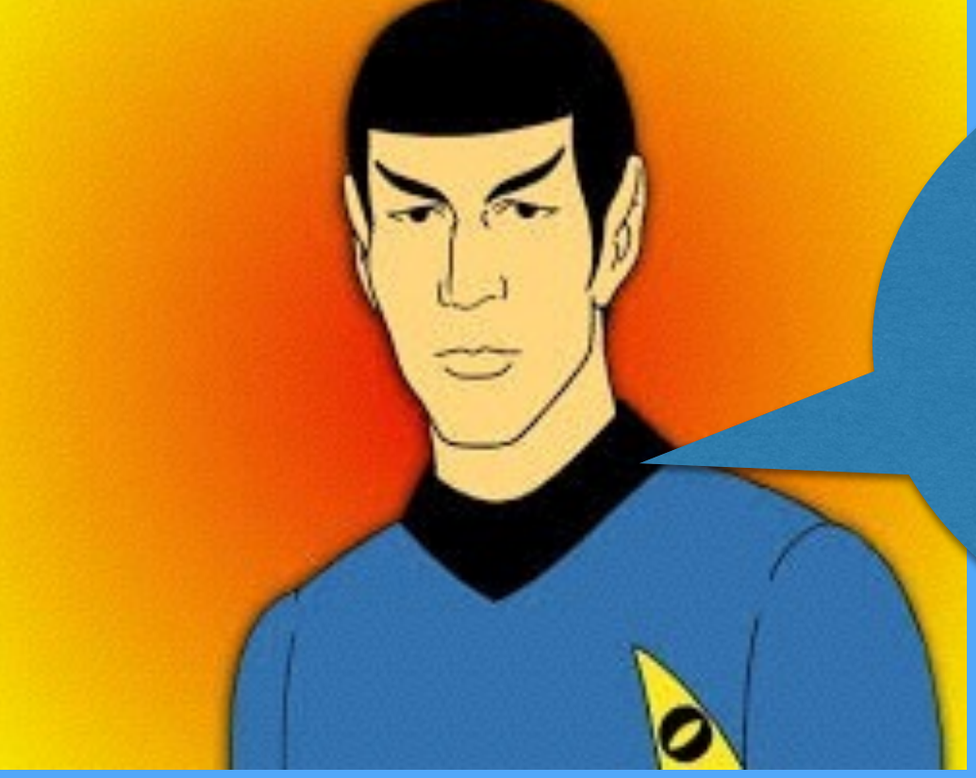
Cuidado com o Horário



Procure não conversar durante a aula

Buscar aprendizado ao invés de pontos

Não teremos pontuação fora dos trabalhos e provas da disciplina



OS trabalhos deveram ser entregues uma semana antes da prova

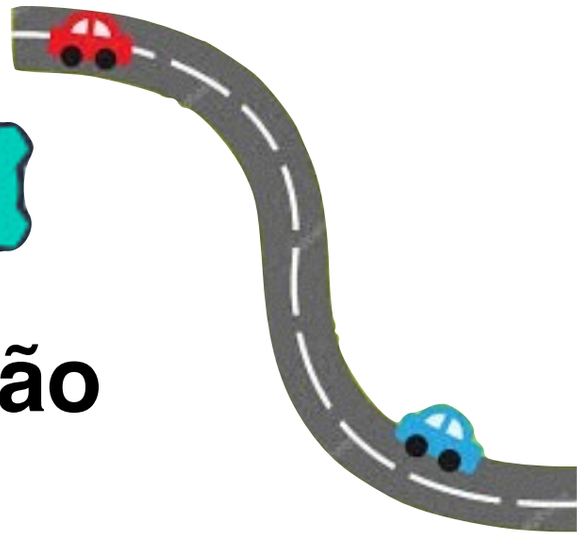
Um cadeira longa e prospera

Não teremos pontos após a prova não adianta pedir

Cronograma da Disciplina

Introdução ao Python

Classificação



Regressão

Dúvidas: naubergois@gmail.com

O que vimos na aula passada

- **Introdução a Classificação**

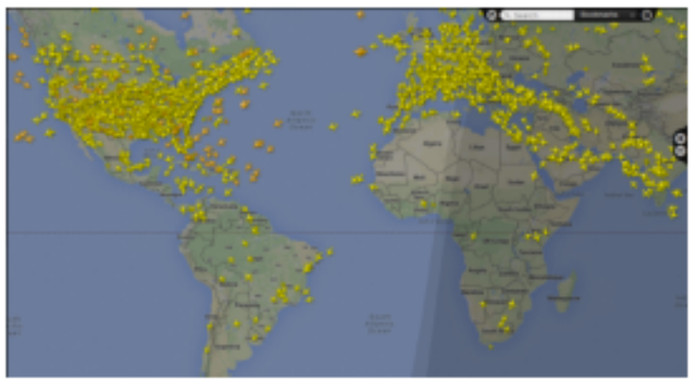
Objetivo da Aula

- **Classificação com Árvores de Decisão**

Porque aprendizado de máquina é importante

Explosão de dados

- Máquinas estão continuamente coletando dados
  - E consumidos
  - Por pessoas e máquinas



Porque aprendizado de máquina é importante

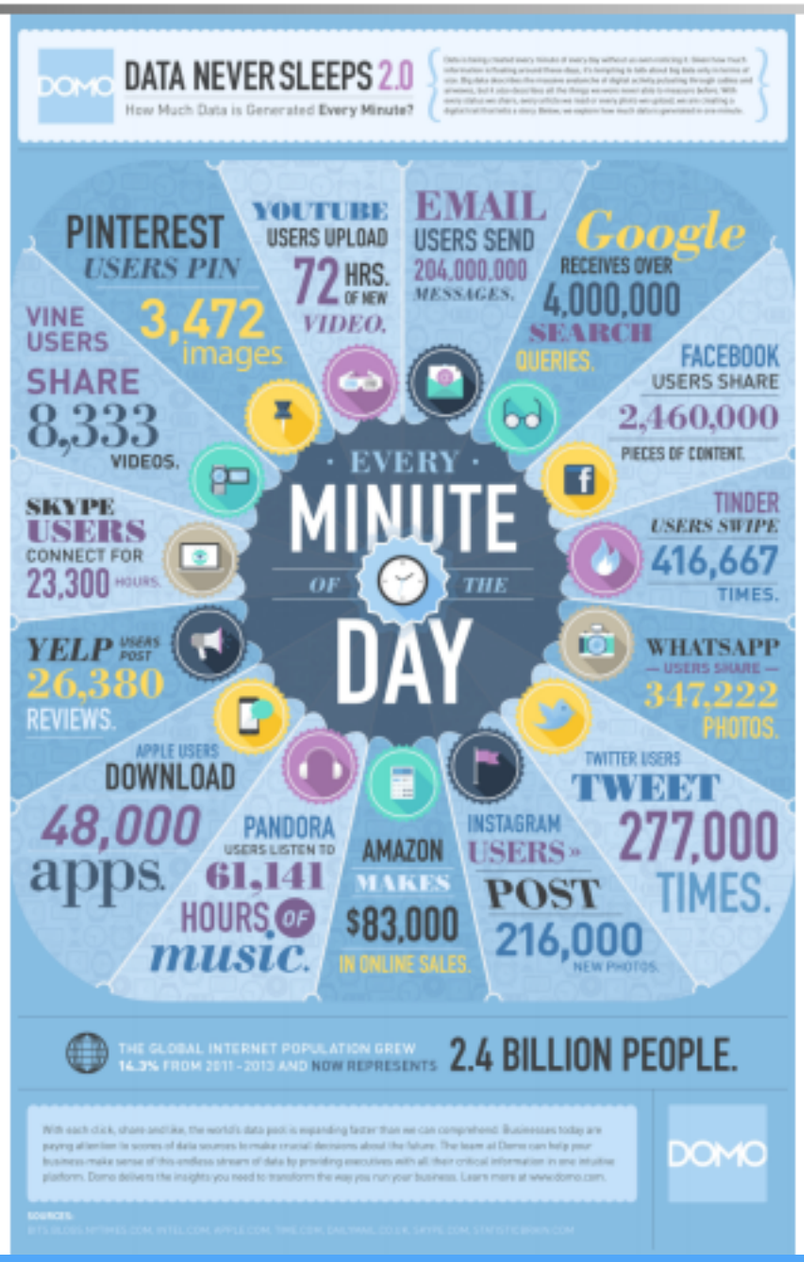
Explosão de dados

- Todo mundo é um cineasta
  - E quer uma grande audiência

- Todo mundo tem um ótimo gosto para vídeos e músicas
  - E quer que todos possam se beneficiar de seu bom gosto
- Todo mundo esta sendo visto e ouvido
  - A toda hora e em todo lugar

Porque aprendizado de máquina é importante

Dados nunca dormem



Quantos dados são gerados a cada minuto

Origem: *Domo business management platform*

https://www.domo.com

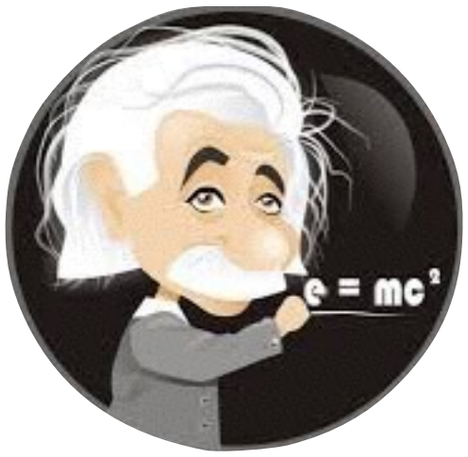
07/2013 e
05/2014

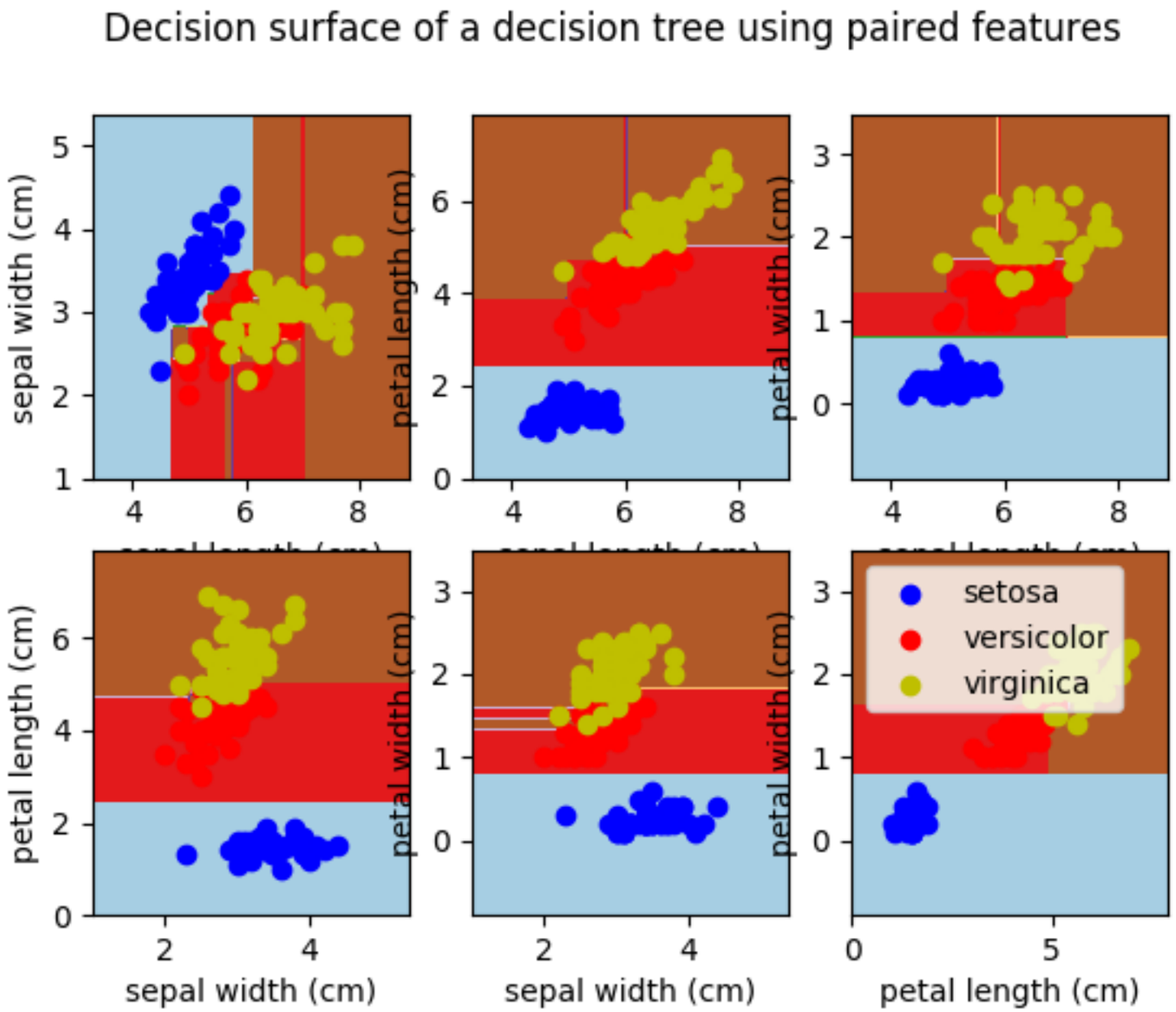
Porque aprendizado de máquina é importante

Carros da Google

http://www.theguardian.com/technology/2015/jun/28/google-self-driving-cars-accidents

Classificação





Classificação

```
# Split a dataset based on an attribute and an attribute value
def test_split(index, value, dataset):
    left, right = list(), list()
    for row in dataset:
        if row[index] < value:
            left.append(row)
        else:
            right.append(row)
    return left, right
```

Classificação

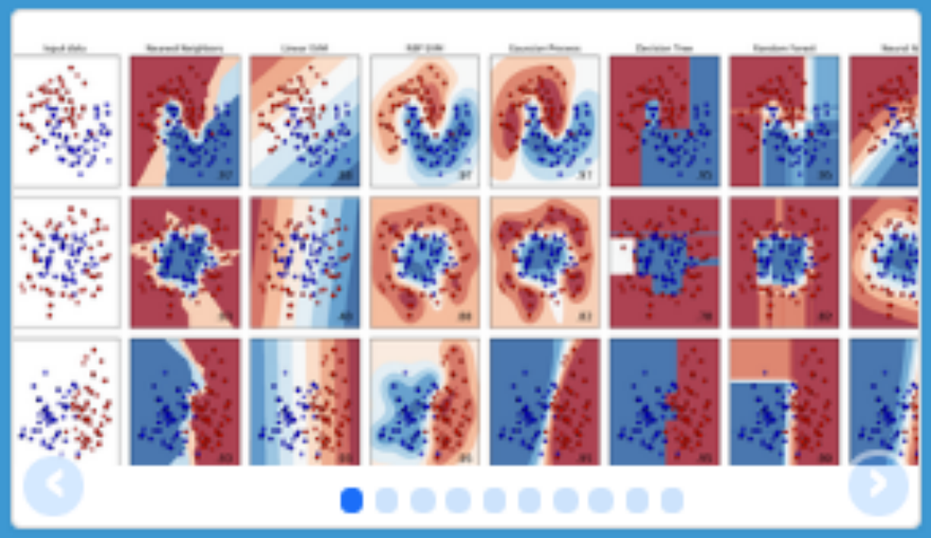
```
# Calculate the Gini index for a split dataset
def gini_index(groups, classes):
	# count all samples at split point
	n_instances = float(sum([len(group) for group in groups]))
	# sum weighted Gini index for each group
	gini = 0.0
	for group in groups:
		size = float(len(group))
		# avoid divide by zero
		if size == 0:
			continue
		score = 0.0
		# score the group based on the score for each class
		for class_val in classes:
			p = [row[-1] for row in group].count(class_val) / size
			score += p * p
		# weight the group score by its relative size
		gini += (1.0 - score) * (size / n_instances)
return gini
```

Classificação

```
# Select the best split point for a dataset
def get_split(dataset):
        class_values = list(set(row[-1] for row in dataset))
        b_index, b_value, b_score, b_groups = 999, 999, 999, None
        for index in range(len(dataset[0])-1):
                for row in dataset:
                        groups = test_split(index, row[index], dataset)
                        gini = gini_index(groups, class_values)
                        print('X%d < %.3f Gini=%.3f' % ((index+1), row[index], gini))
                        if gini < b_score:
                                b_index, b_value, b_score, b_groups = index, row[index], gini, groups
return {'index':b_index, 'value':b_value, 'groups':b_groups}
```

Classificação

```
dataset = [[2.771244718,1.784783929,0],
	[1.728571309,1.169761413,0],
	[3.678319846,2.81281357,0],
	[3.961043357,2.61995032,0],
	[2.999208922,2.209014212,0],
	[7.497545867,3.162953546,1],
	[9.00220326,3.339047188,1],
	[7.444542326,0.476683375,1],
	[10.12493903,3.234550982,1],
	[6.642287351,3.319983761,1]]
split = get_split(dataset)
print('Split: [X%d < %.3f]' % ((split['index']+1), split['value']))
```



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, … — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, … — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, … — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics. — Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction. — Examples

Vamos instalar o SKLearn

Classificação

Packaging and Distributing Projects

This section covers the basics of how to configure, package and distribute your own Python projects. It assumes that you are already

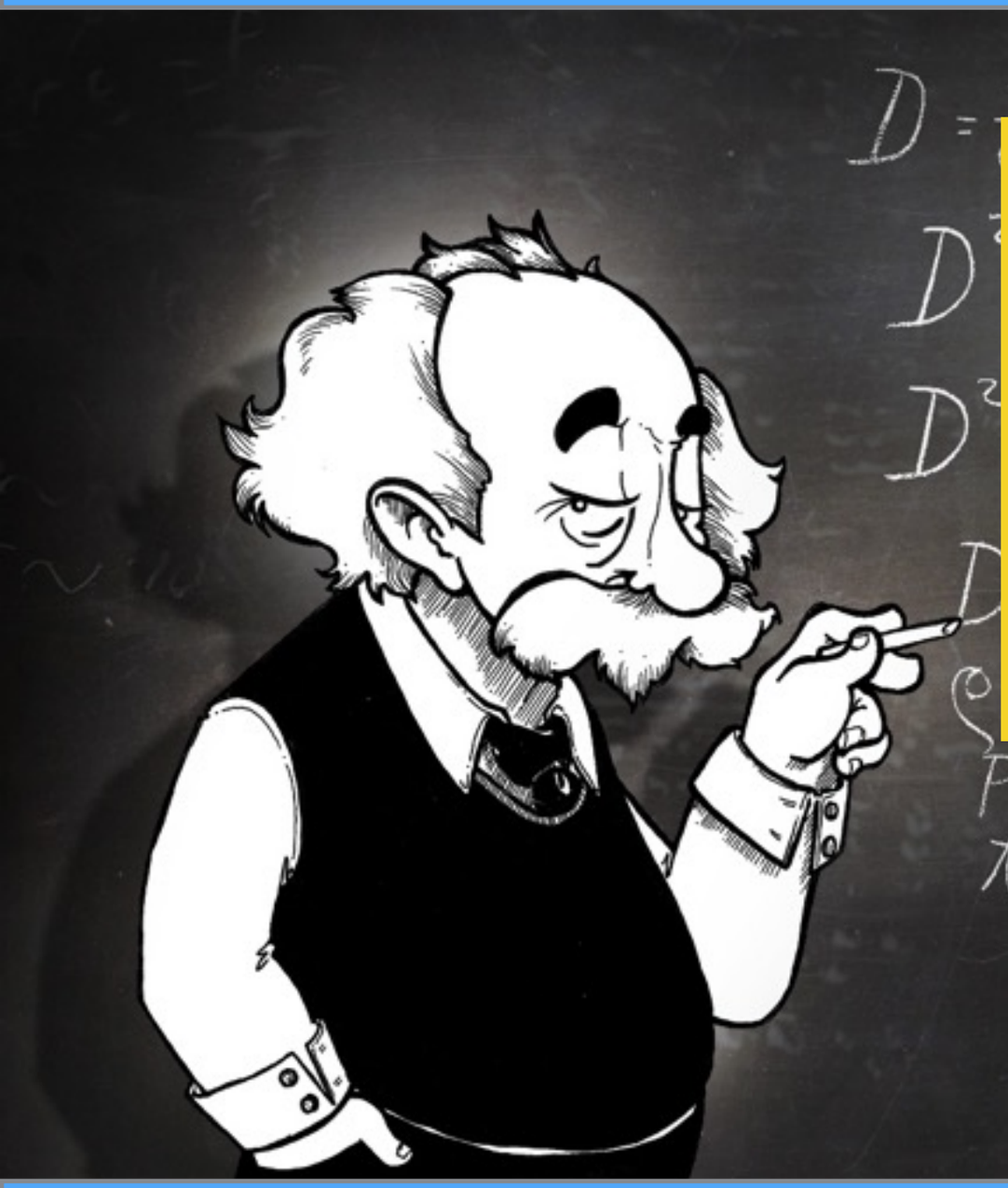
https://packaging.python.org/tutorials/distributing-packages/

The se ...provide guidan

For more reference material, see Building and Distributing Packages in the setuptools docs, but note that some advisory content there may be outdated. In the event of conflicts, prefer the advice in the Python Packaging User Guide.

Contents

Árvores de Decisão com SKLearn



```
>>> from sklearn import tree
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf =
        tree.DecisionTreeClassifier()
>>> clf = clf.fit(X, Y)
```

IRIS dataset

Iris Versicolor

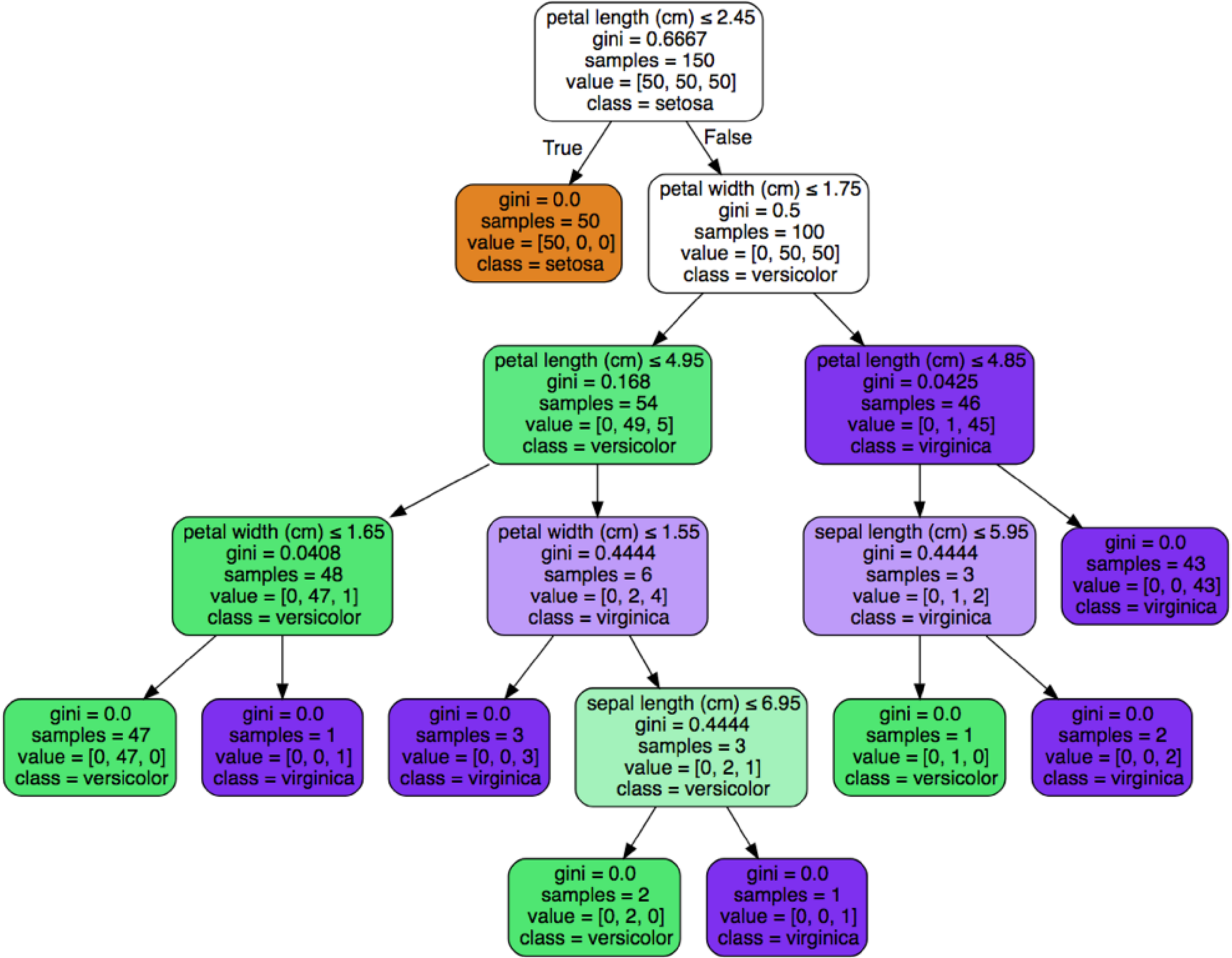
Iris Setosa

Iris Virginica

Árvore de decisão com SKLearn

```
>>> from sklearn.datasets import load_iris
>>> from sklearn import tree
>>> iris = load_iris()
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(iris.data, iris.target)
```


```
>>> import graphviz
>>> dot_data = tree.export_graphviz(clf, out_file=None)
>>> graph = graphviz.Source(dot_data)
>>> graph.render("iris")
```


petal length (cm) ≤ 2.45
gini = 0.6667
samples = 150
value = [50, 50, 50]
class = setosa
True
False
gini = 0.0
samples = 50
value = [50, 0, 0]
class = setosa
petal width (cm) ≤ 1.75
gini = 0.5
samples = 100
value = [0, 50, 50]
class = versicolor
petal length (cm) ≤ 4.95
gini = 0.168
samples = 54
value = [0, 49, 5]
class = versicolor
petal length (cm) ≤ 4.85
gini = 0.0425
samples = 46
value = [0, 1, 45]
class = virginica
petal width (cm) ≤ 1.65
gini = 0.0408
samples = 48
value = [0, 47, 1]
class = versicolor
petal width (cm) ≤ 1.55
gini = 0.4444
samples = 6
value = [0, 2, 4]
class = virginica
sepal length (cm) ≤ 5.95
gini = 0.4444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 43
value = [0, 0, 43]
class = virginica
gini = 0.0
samples = 47
value = [0, 47, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
gini = 0.0
samples = 3
value = [0, 0, 3]
class = virginica
sepal length (cm) ≤ 6.95
gini = 0.4444
samples = 3
value = [0, 2, 1]
class = versicolor
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica

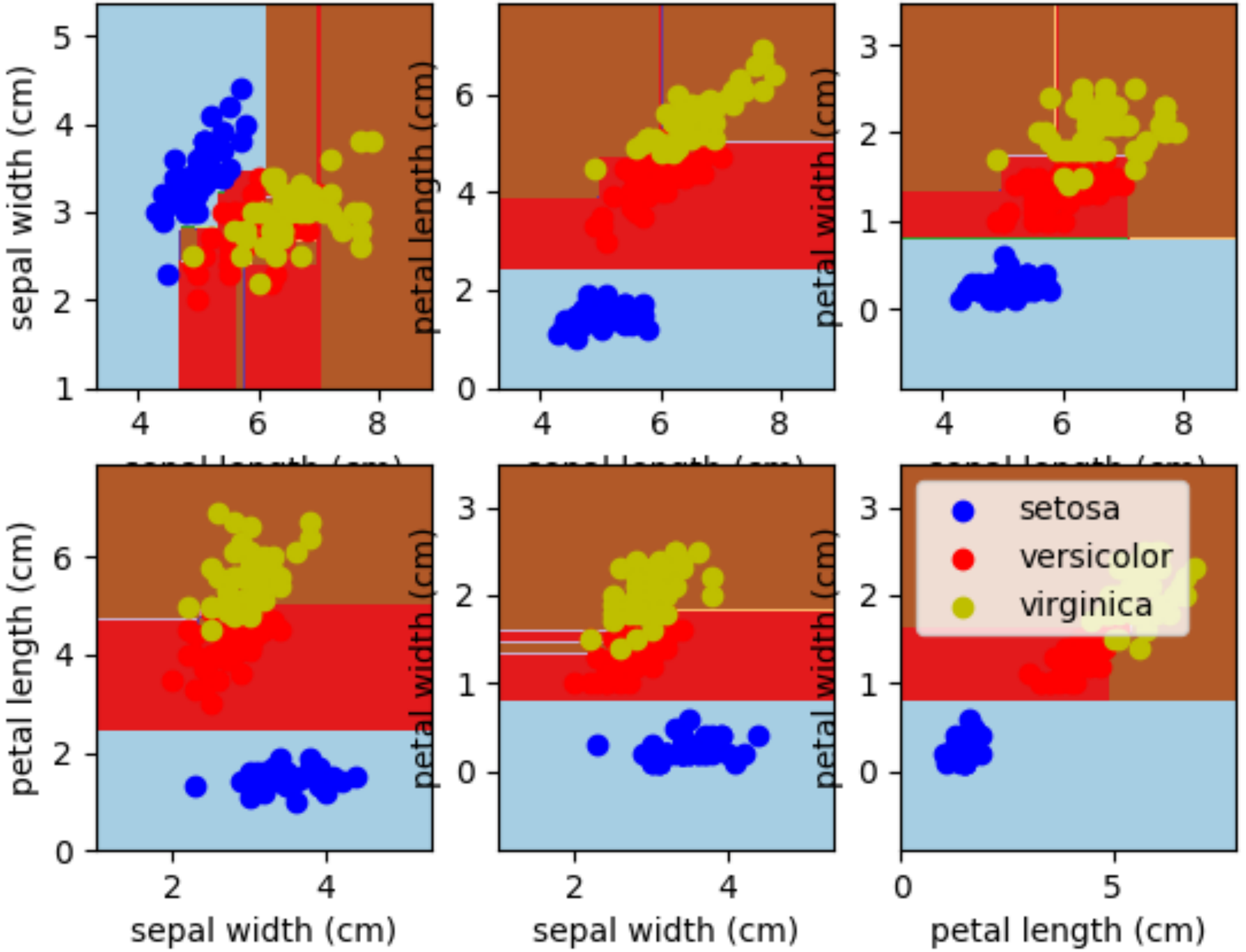
Árvores de Decisão

```
>>> clf.predict(iris.data[:1, :])
array([0])
```

```
>>> clf.predict_proba(iris.data[:1, :])
array([[ 1.,  0.,  0.]])
```

Dúvidas: naubergois@gmail.com

Decision surface of a decision tree using paired features



```
def split(node, max_depth, min_size, depth):
    left, right = node['groups']
    del(node['groups'])
    # check for a no split
    if not left or not right:
        node['left'] = node['right'] = to_terminal(left + right)
        return
    # check for max depth
    if depth >= max_depth:
        node['left'], node['right'] = to_terminal(left), to_terminal(right)
        return
# process left child
if len(left) <= min_size:
    node['left'] = to_terminal(left)
else:
    node['left'] = get_split(left)
    split(node['left'], max_depth, min_size, depth+1)
# process right child
if len(right) <= min_size:
    node['right'] = to_terminal(right)
else:
    node['right'] = get_split(right)
    split(node['right'], max_depth, min_size, depth+1)
```

```
# Build a decision tree
def build_tree(train, max_depth, min_size):
    root = get_split(train)
    split(root, max_depth, min_size, 1)
    return root
```

```
dataset = [[2.771244718,1.784783929,0],
    [1.728571309,1.169761413,0],
    [3.678319846,2.81281357,0],
    [3.961043357,2.61995032,0],
    [2.999208922,2.209014212,0],
    [7.497545867,3.162953546,1],
    [9.00220326,3.339047188,1],
    [7.444542326,0.476683375,1],
    [10.12493903,3.234550982,1],
    [6.642287351,3.319983761,1]]
tree = build_tree(dataset, 1, 1)
print  tree(tree)
```

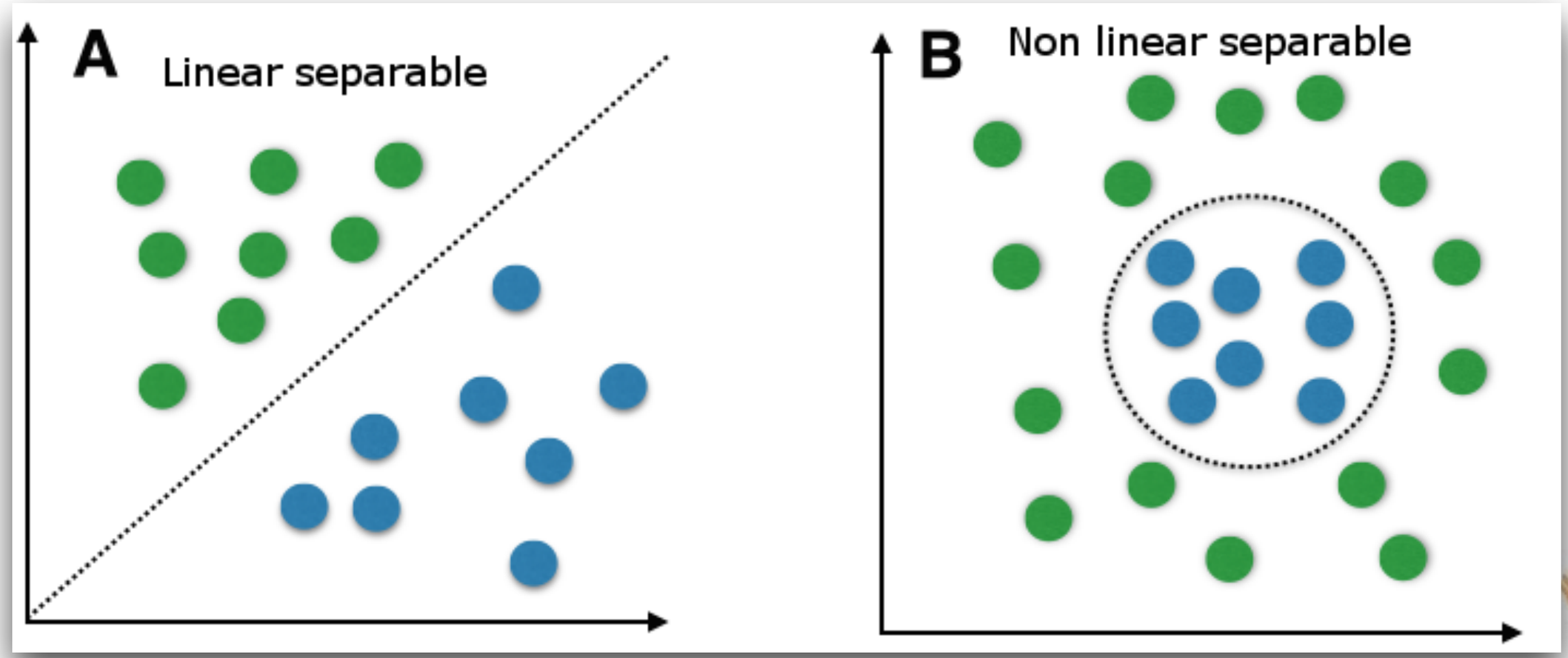
```
# Make a prediction with a decision tree
def predict(node, row):
    if row[node['index']] < node['value']:
        if isinstance(node['left'], dict):
            return predict(node['left'], row)
        else:
            return node['left']
    else:
        if isinstance(node['right'], dict):
            return predict(node['right'], row)
        else:
            return node['right']
```

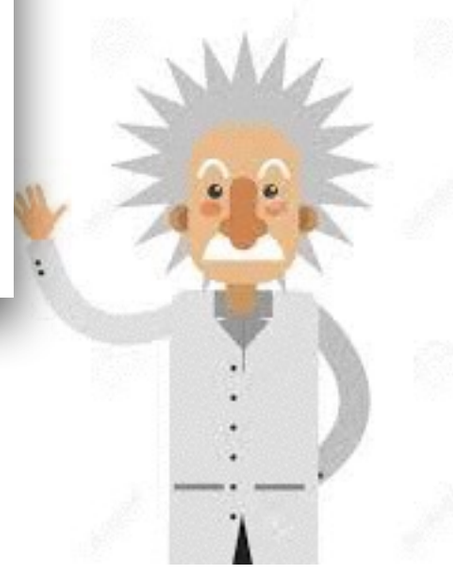
```
dataset = [[2.771244718,1.784783929,0],
	[1.728571309,1.169761413,0],
	[3.678319846,2.81281357,0],
	[3.961043357,2.61995032,0],
	[2.999208922,2.209014212,0],
	[7.497545867,3.162953546,1],
	[9.00220326,3.339047188,1],
	[7.444542326,0.476683375,1],
	[10.12493903,3.234550982,1],
	[6.642287351,3.319983761,1]]

#  predict with a stump
stump = {'index': 0, 'right': 1, 'value': 6.642287351, 'left': 0}
for row in dataset:
	prediction = predict(stump, row)
	print('Expected=%d, Got=%d' % (row[-1], prediction))
```

http://machinelearningmastery.com/implement-decision-tree-algorithm-scratch-python/

Veja o exemplo completo na URL Acima

Classificação Linear





Equação linear

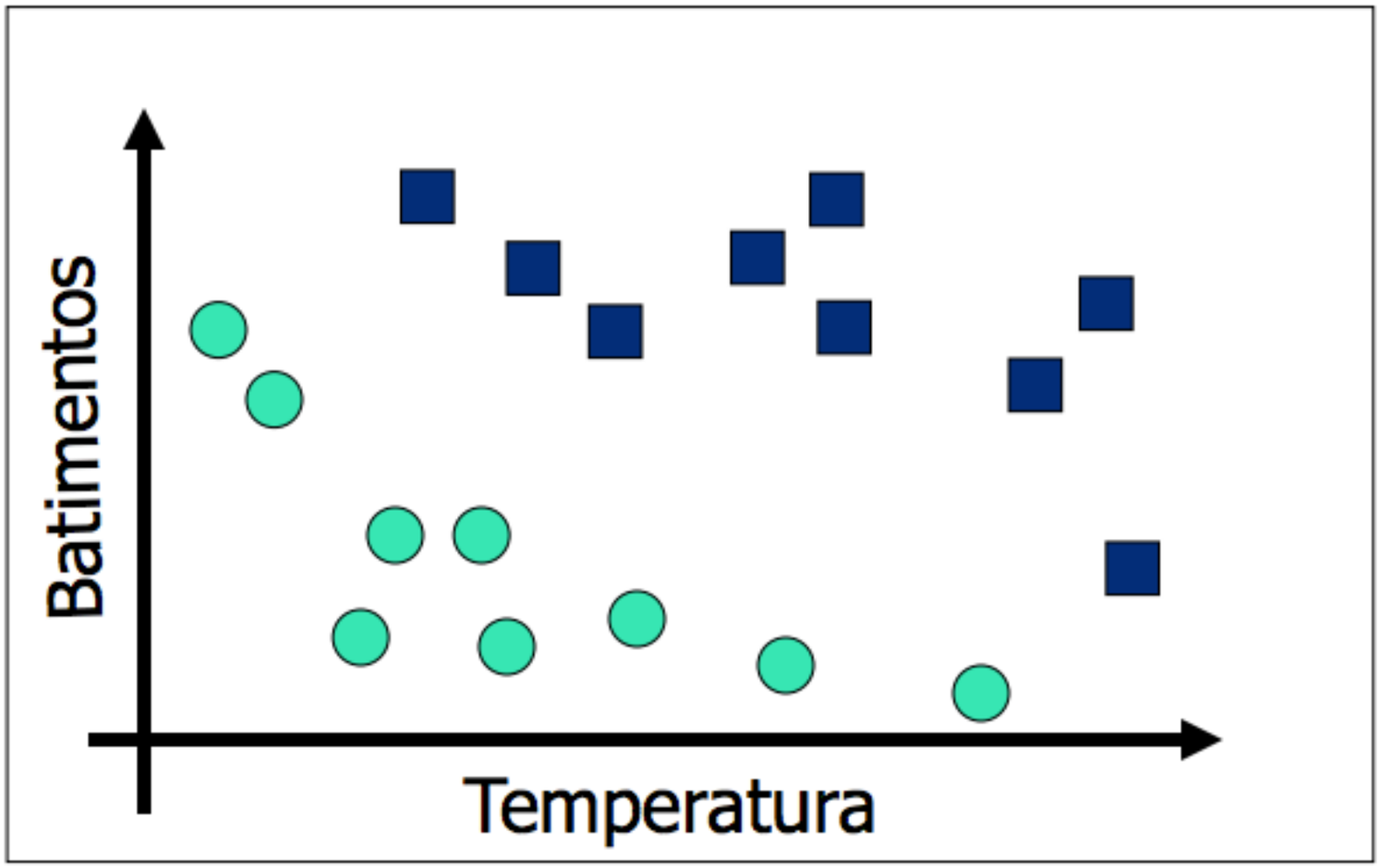
Equação linear é toda equação da forma:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1$$

em que $\mathbf{a_{11}, a_{12}, a_{13}, \ldots, a_{1n}}$ são números reais, q
recebem o nome de *coeficientes* das *incógnitas*; x
$\mathbf{x_2, x_3, \ldots, x_n}$, são as incógnitas; e $\mathbf{b_1}$ é um núme
real chamado *termo independente* (quando b=0,
equação recebe o nome de *linear homogênea*).

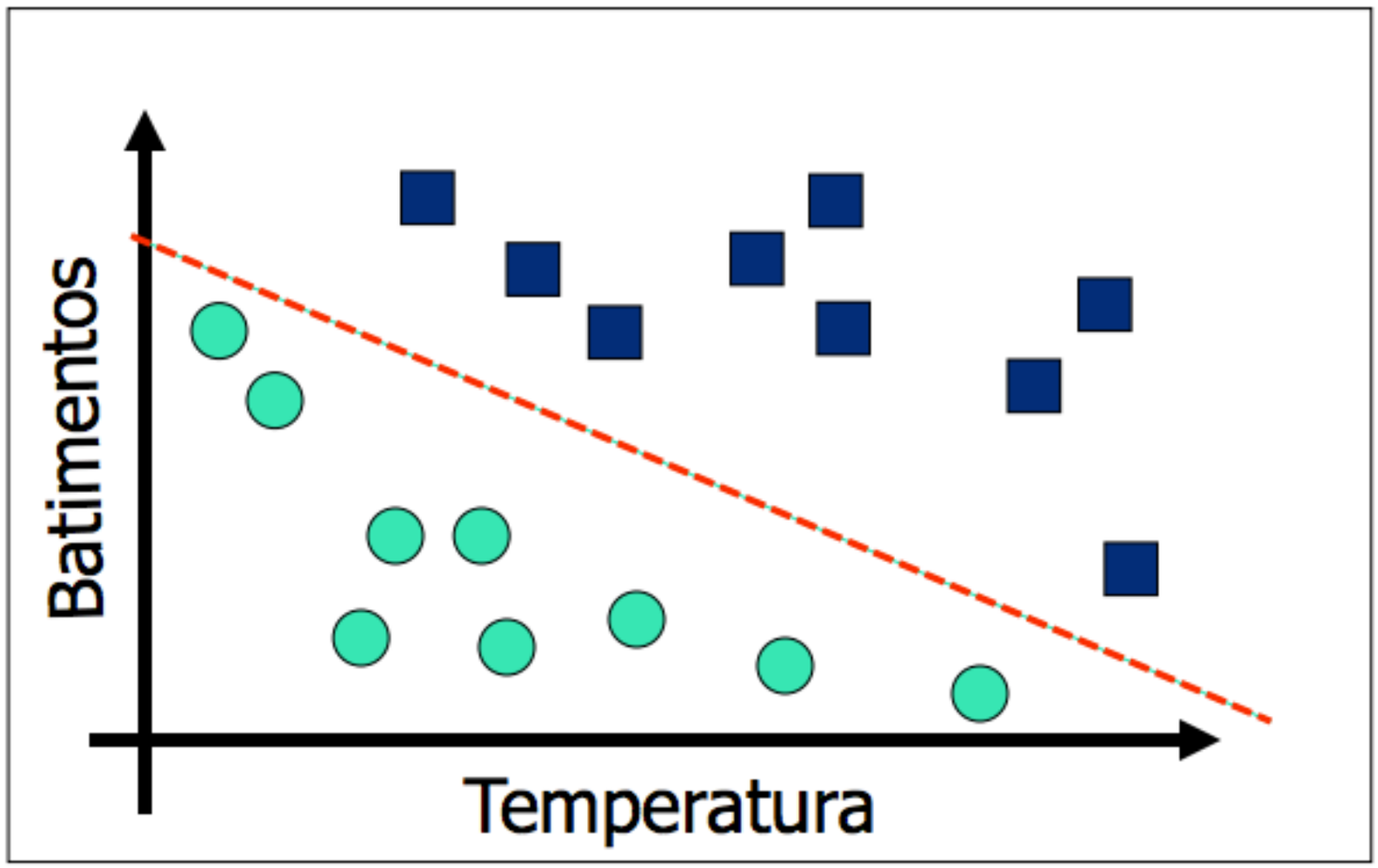
Classificação Linear

- Incluir número de batimentos



Classificação Linear

- Função linear permite diagnóstico

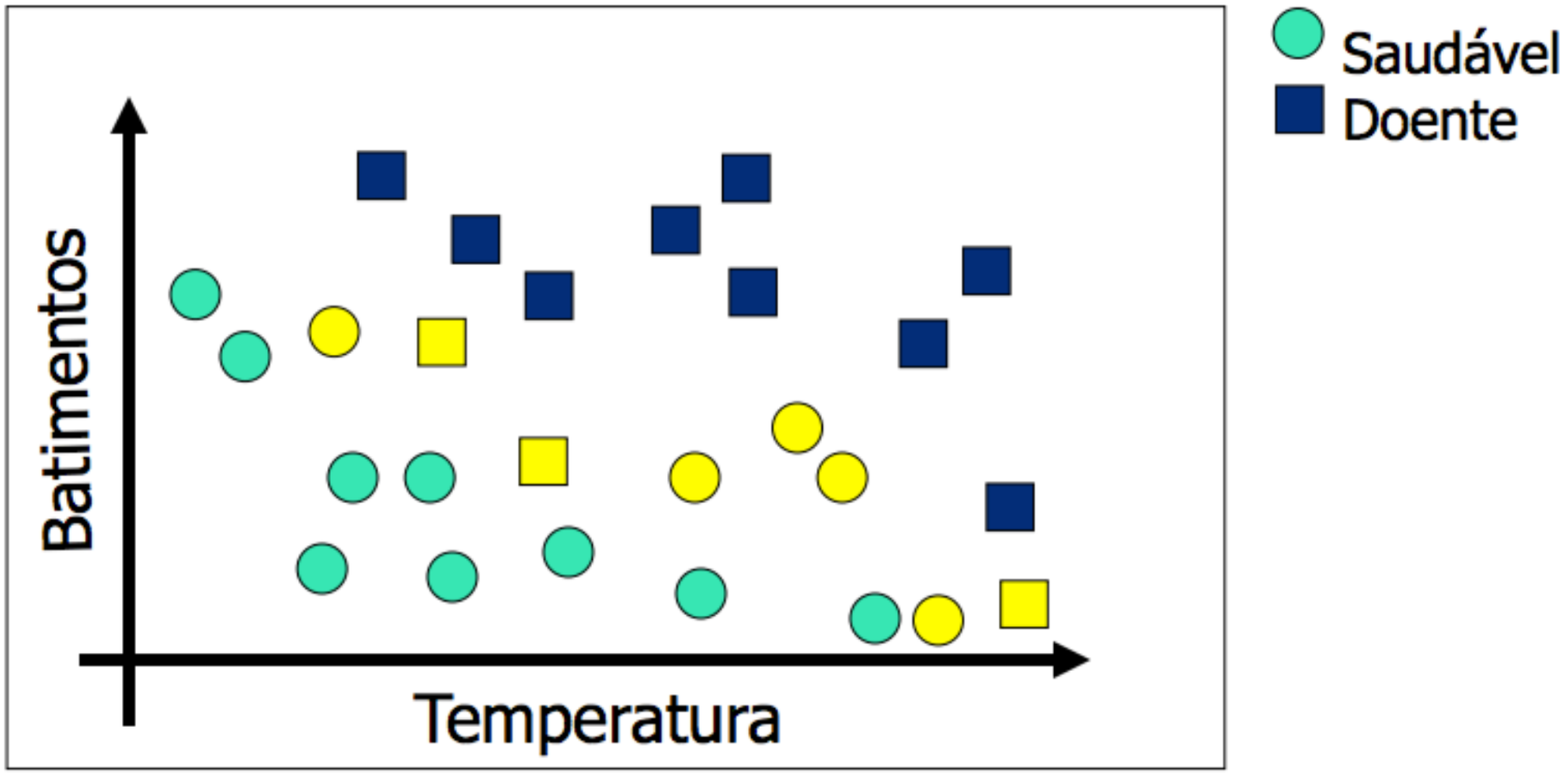


Saudável
Doente

Nova função:
Se a.t + b > 0
Então doente
Senão saudável

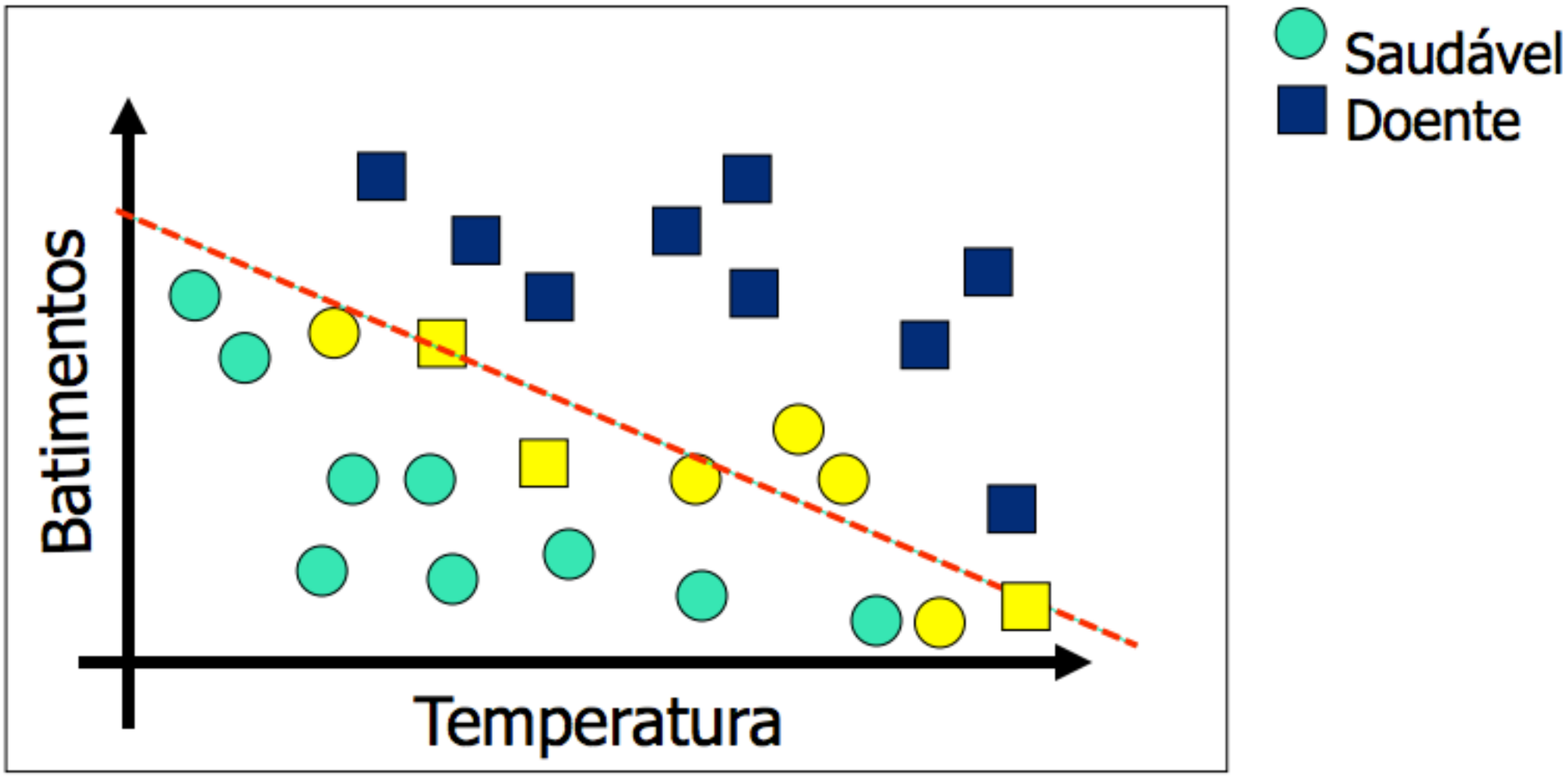
Classificação Linear

- Supor inclusão de outros pacientes



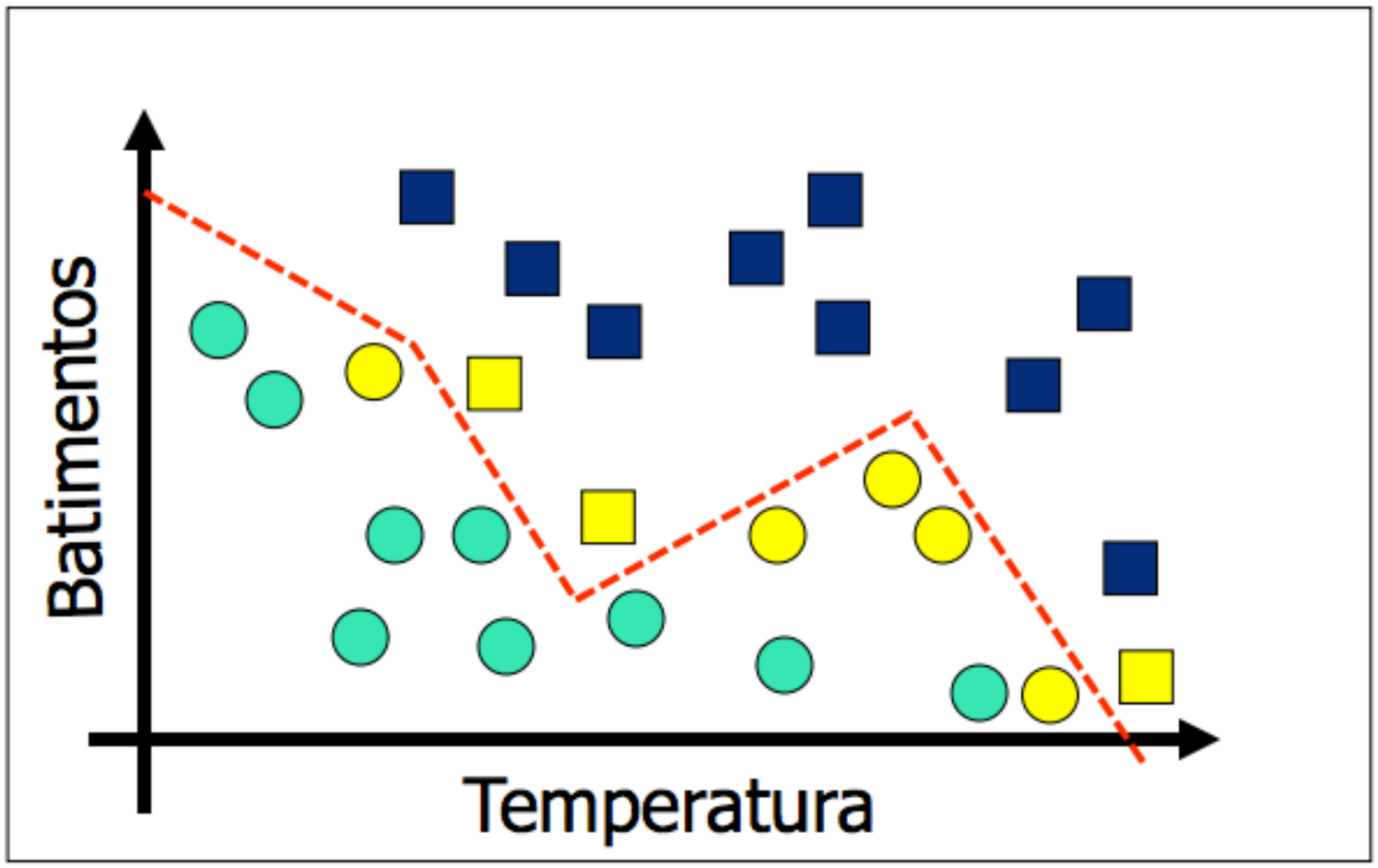
Classificação Linear

- Supor inclusão de outros pacientes



Classificação Linear

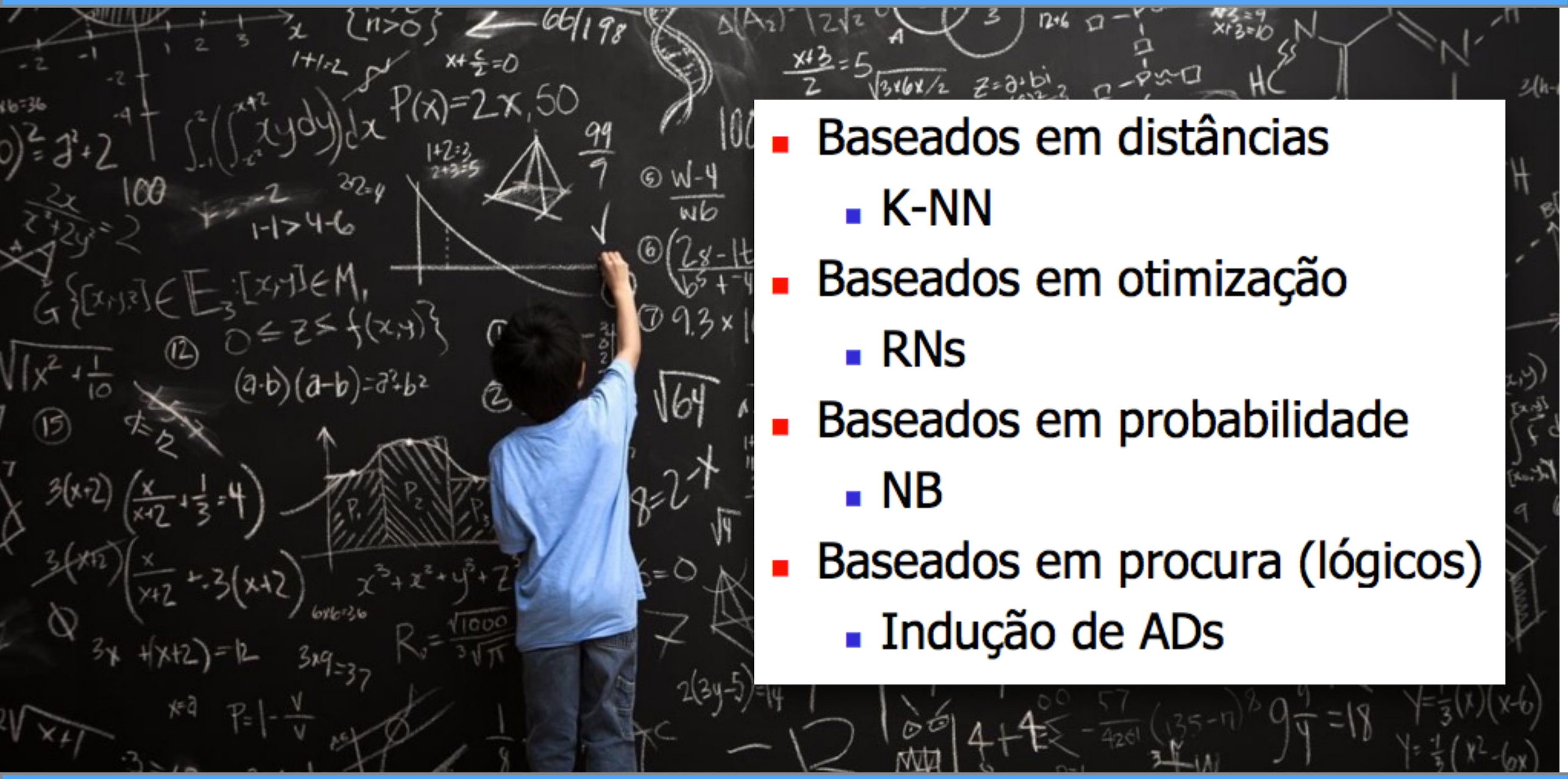
- Supor inclusão de outros pacientes



Saudável
Doente

Nova função:
Muito complexa
Para por aqui

Critérios difrentes para classificação



Critérios difrentes para classificação

- Baseados em distâncias
  - K-NN
- Baseados em otimização
  - RNs

Geométricos

- Baseados em probabilidade
  - NB
- Baseados em procura (lógicos)
  - Indução de ADs

Regressão Linar do Zero

The line for a simple linear regression model can be written as:

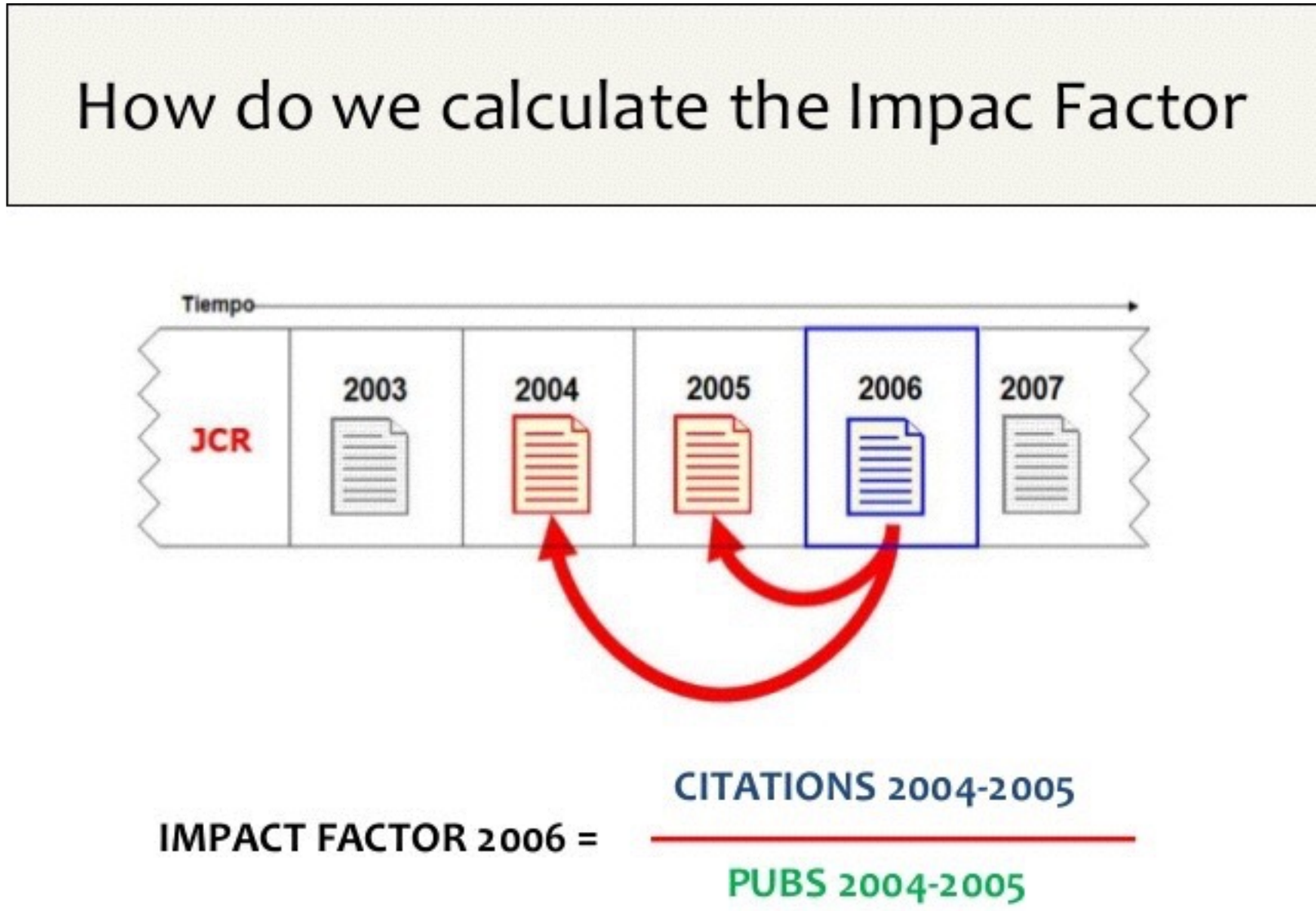
```
y = b0 + b1 * x
```

```
# Calculate the mean value of a list of numbers
def mean(values):
	return sum(values) / float(len(values))
```

Pesquisa

Momento Pesquisa

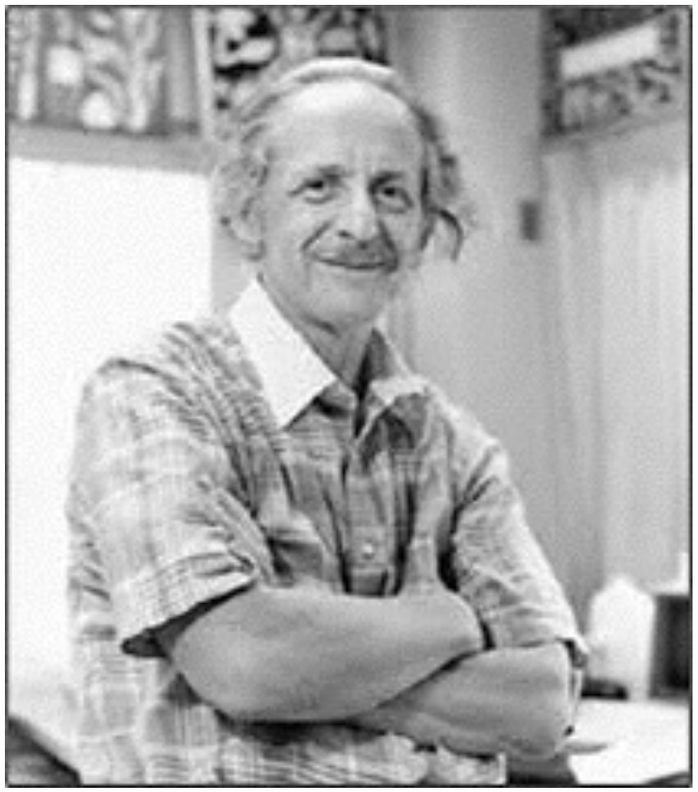
O Fator de Impacto, abreviado como FI, é uma medida que reflete o número médio de citações de artigos científicos publicados em determinado periódico. É empregado frequentemente para avaliar a importância de um dado periódico em sua área, sendo que aqueles com um maior FI são considerados mais importantes do que aqueles com um menor FI



Pesquisa

Momento Pesquisa

O FI foi criado por Eugene Garfield,[1] o fundador do Institute for Scientific Information (ISI), hoje parte da Thomson Reuters. Desde 1972 os FI são calculados anualmente para os periódicos indexados ao ISI e depois publicados no Journal Citation Reports (JCR), também da Thomson Reuters.



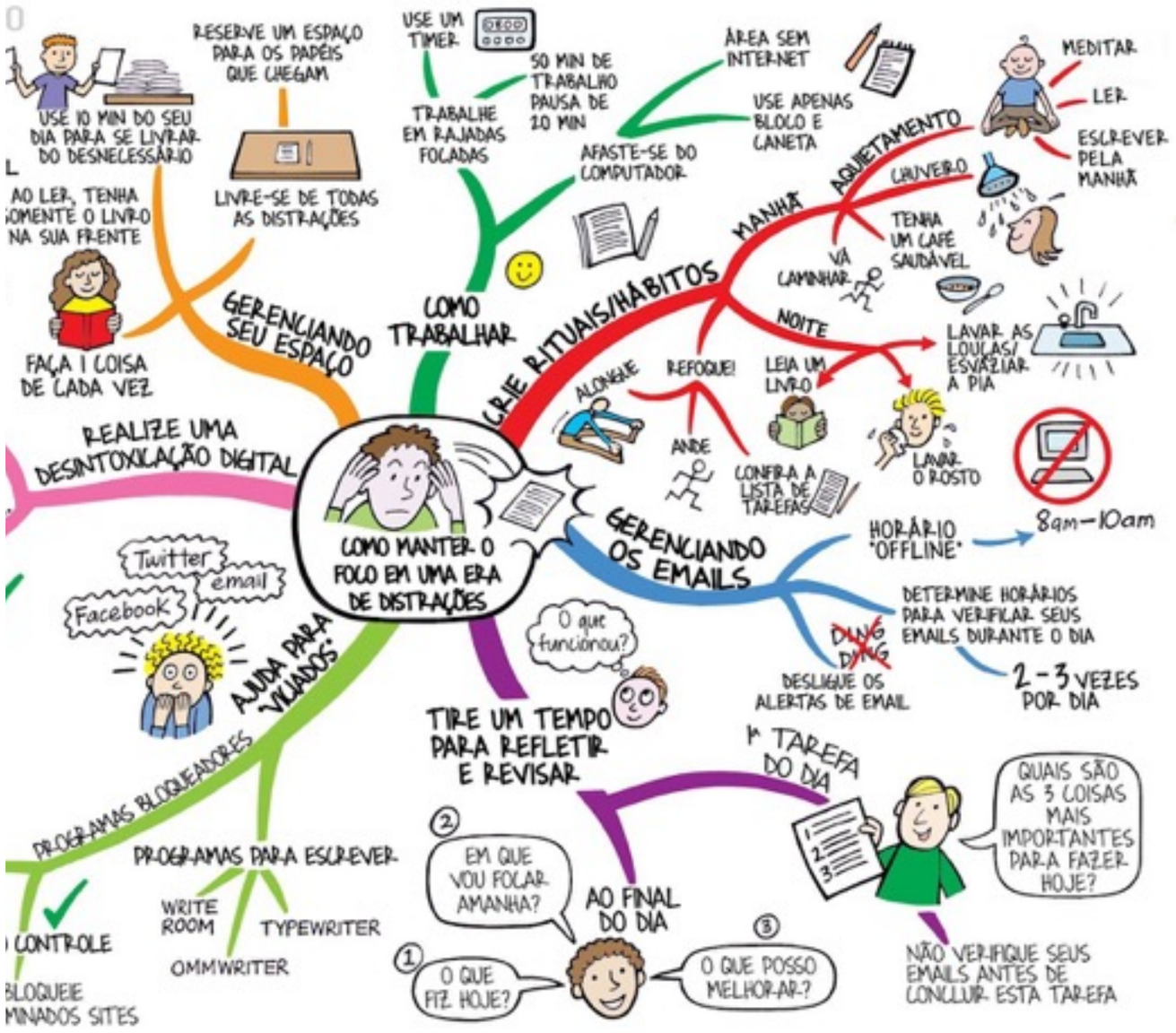
Pesquisa

Momento Pesquisa

O Prêmio Turing é concedido anualmente pela Associação para Maquinaria da Computação (em inglês: Association for Computing Machinery, ou ACM) para uma pessoa selecionada por contribuições à computação. As contribuições devem ser duradouras e fundamentais no campo computacional.

Mapa Mental

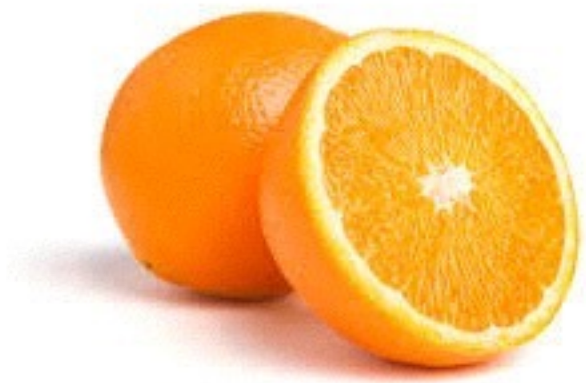
Mapa mental, ou mapa da mente é o nome dado para um tipo de diagrama, sistematizado pelo psicólogo inglês Tony Buzan, voltado para a gestão de informações, de conhecimento e de capital intelectual



Exercício

Exercício 1

Separem-se em equipe de até 5 pessoas e criem um mapa mental descrevendo o que foi feito na aula (10 minutos).



Ce sont les Batraciens, que le voyageur rencontre à chaque pas sur le sol japonais, ils fréquentent les lieux habités et peuplent les fertiles vallées, d'où une culture [illegible] à banni les oiseaux et les autres animaux sauvages. Fuyant la lumière et les lieux habités, la famille des Batraciens à queue généralement désignée par le nom de Salamandre, se retire dans les ombres des montagnes riches en sources et en ombrage. L'existence de différentes espèces de Salamandre au Japon, est d'une haute importance pour l'étude de la distribution géographique de ces reptiles, auxquels on a assigné jusqu'à ce jour l'Europe et l'Amérique du Nord pour patrie; mais ce qui est plus important encore c'est la découverte d'une espèce de Salamandre, qui par sa taille extraordinaire et sa forme spécifique, rappelle une création antédiluvienne, cette Salamandre le Géant des Batraciens est le représentant d'une race appartenant à cette longue période de notre globe qui sépare les formations houillères des terrains tertiaires, et qui vit apparaître au milieu de mers des reptiles gigantesques et d'organisations bizarres. Je veux parler de *l'homo diluvii testis*, la célèbre Salamandre fossile des carrières d'Oeningen qui depuis Scheuchzer jusqu'à Cuvier dont les écrits ont jeté une si vive lumière sur le monde primitif, a été l'objet des spéculations des naturalistes.

Notre grande Salamandre (Salamandra maxima) vit dans les profondes vallées des hautes montagnes de Nippon entre le 34° et 36° de lat. N.; elle séjourne dans les ruisseaux, dans les bassins et dans les lacs formés par les eaux pluviales au milieu des cratères des Volcans éteints à une hauteur de 4 à 5000 pieds au dessus du niveau de la mer. Quelquefois elle quitte pendant la nuit les eaux qui lui servent d'asile; mais son organisation et ses habitudes la rappellent bientôt dans cet élément, où elle trouve plus facilement que sur terre, une nourriture qui consiste en petits poissons, en grenouilles et en vers. C'est à [illegible] petit village situé aux pieds du mont Suzuka yama à 15 *Ri* environ à l'est de Miyako, que j'observai pour la première fois cette Salamandre. Un de mes disciples, le Docteur [illegible], avait chargé un herboriste qui habite cette montagne de faire la recherche de ce rare et curieux animal. D'après les renseignements des montagnards le *Sanshouuo* — c'est le nom indigène vulgaire de la grande Salamandre — se trouve le plus souvent dans les montagnes d'Okuda yama. J'ai eu le bonheur d'en rapporter une vivante en Europe. Elle existe encore au Musée des Pays-Bas, où elle a atteint une longueur d'environ trois pieds, taille extraordinaire que je n'ai jamais observée même au Japon. Cet individu a été depuis plusieurs années l'objet des observations de M. Schlegel, qui en a donné une description complète sous tous les rapports.

Dúvidas: naubergois@gmail.com

Francisco Nauber Bernardo Gois
Email: naubergois@gmail.com