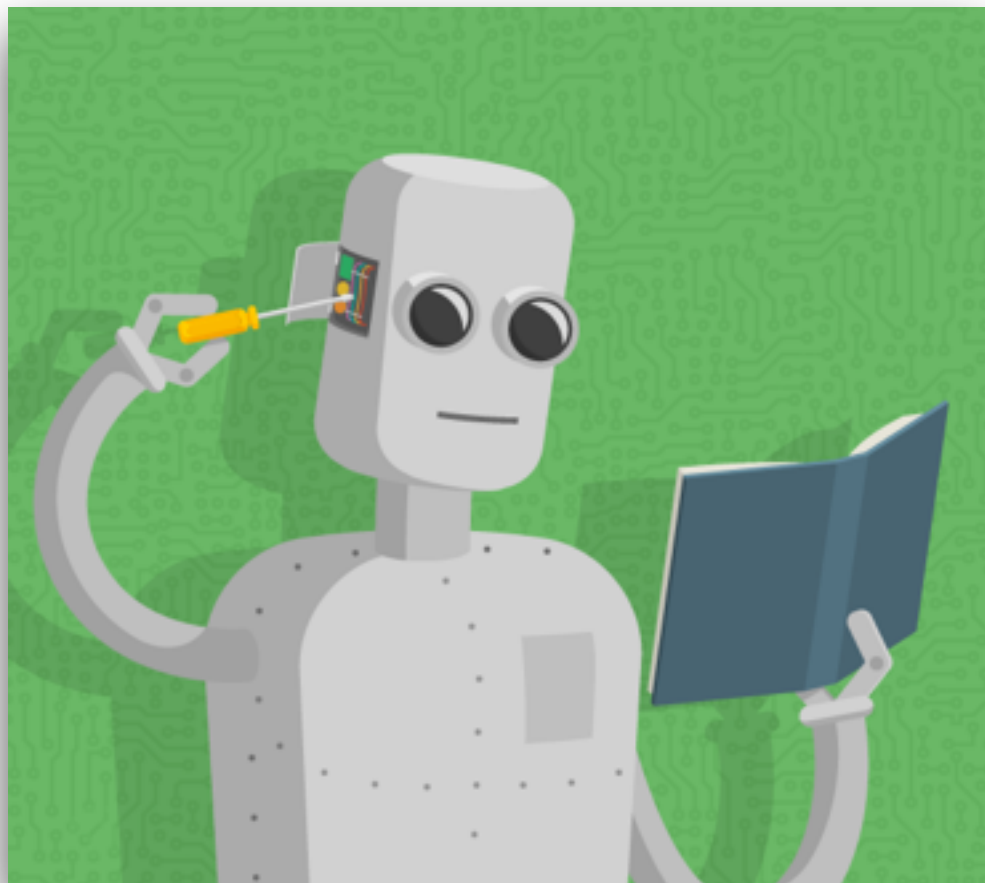




UNIFOR
ENSINANDO E APRENDENDO

Disciplina de Inteligência Artificial (Aula 5)



Classificação Baseada em Instâncias e TensorFlow

Apresentação

Francisco Nauber Bernardo Gois



Analista aprendizado de máquina
no Serviço Federal de Processamento
de Dados

Doutorando em Informática Aplicada

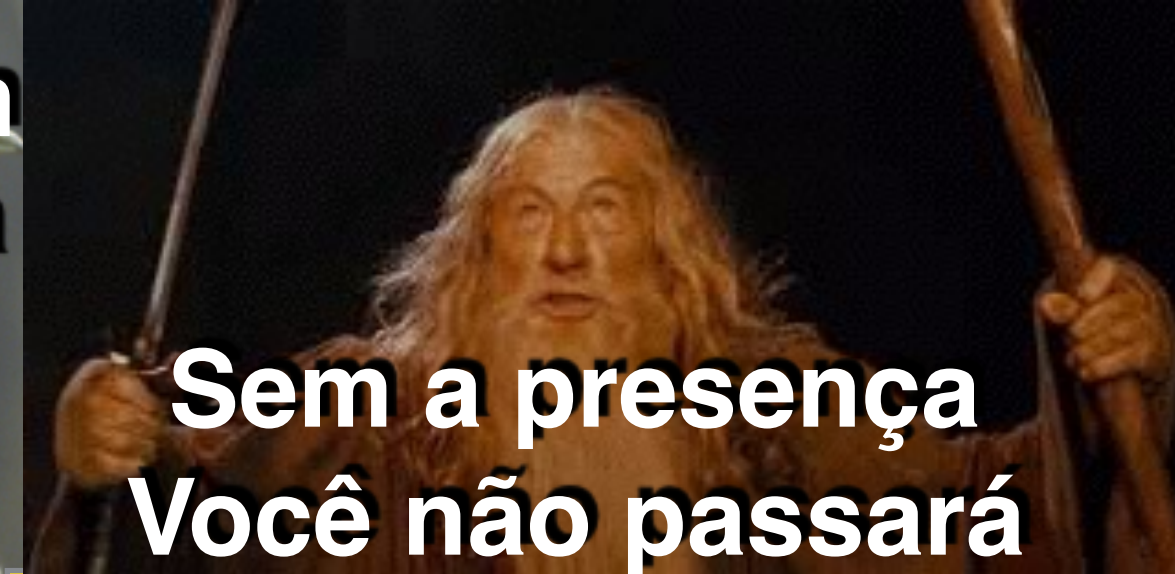
Mestre em Informática Aplicada

Especialista em desenvolvimento WEB

**Jovem Padawan
procure na aula**



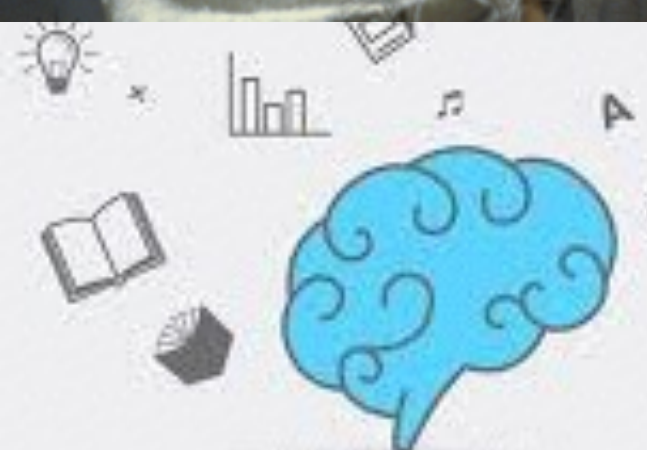
**Sem a presença
Você não passará**



**ao telefone
não falar**

**Para melhor
desempenho na
aula**

**Cuidado
com o
Horário**



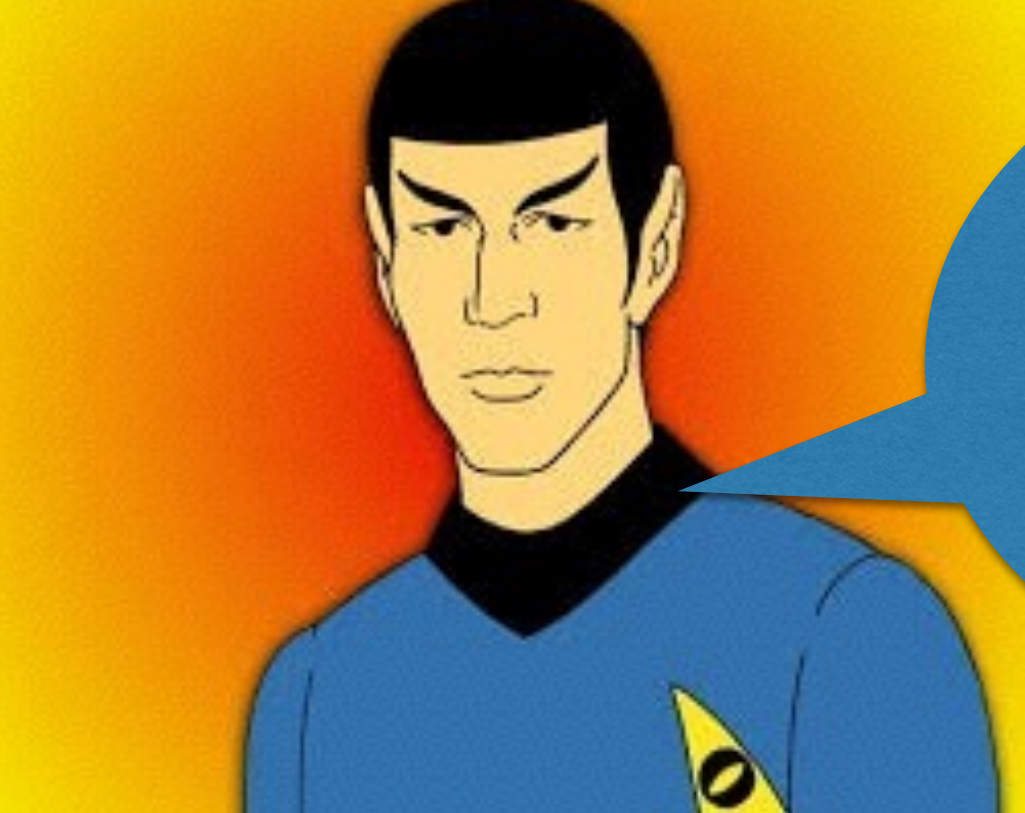
**Procure
não
conversar
durante a
aula**

**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



O que vimos na aula passada

- **Classificação com Árvores de Decisão**

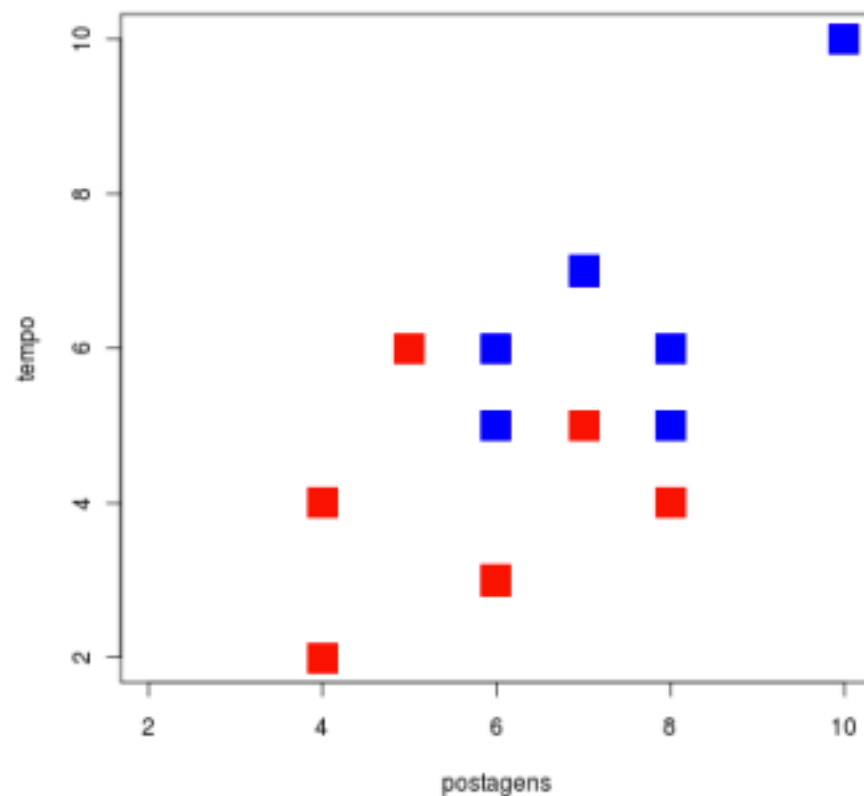
Objetivo da Aula

- **Classificação baseada em Instâncias**
- **Aprendendo a plotar gráficos no python**
- **Introdução ao TensorFlow**

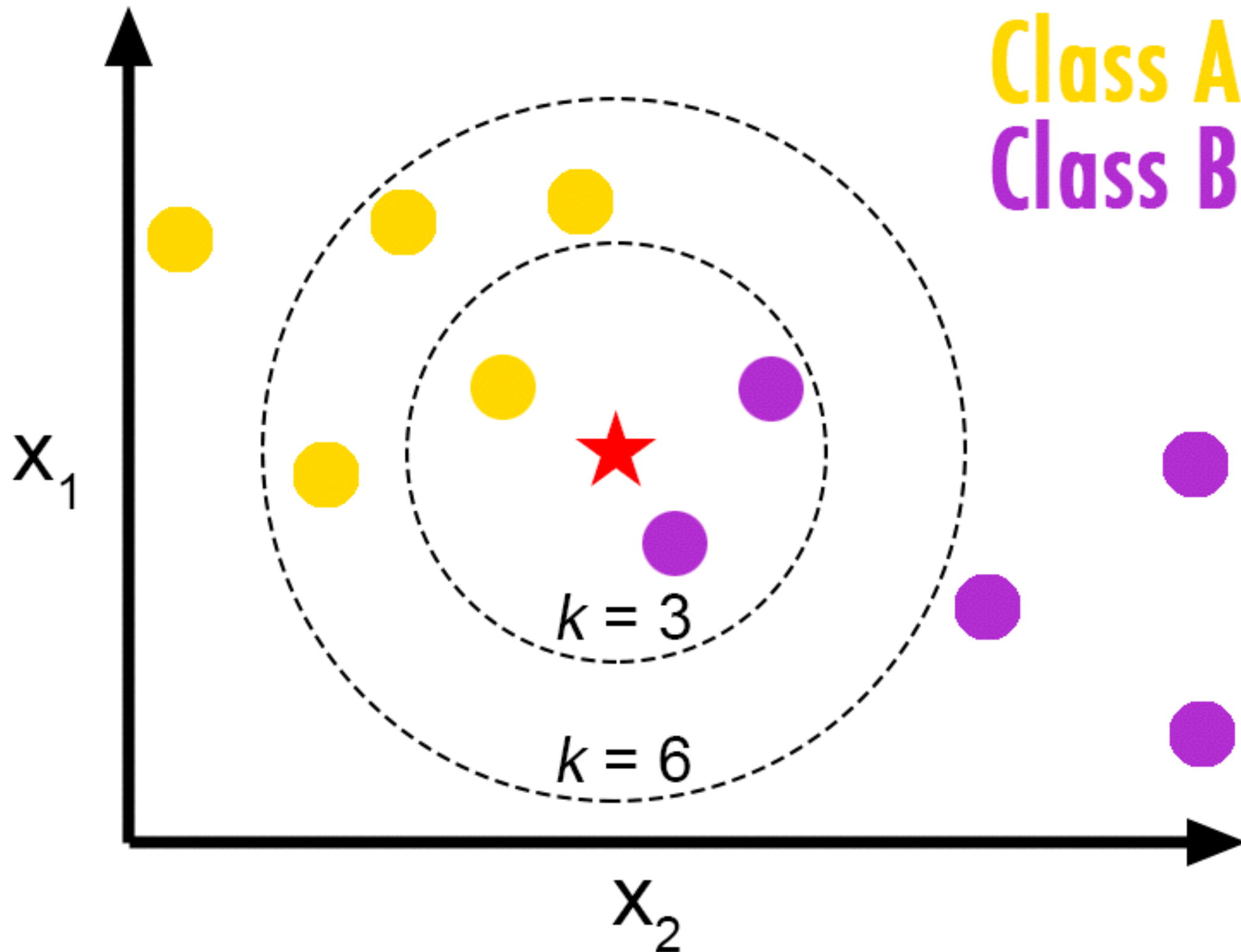
KNN

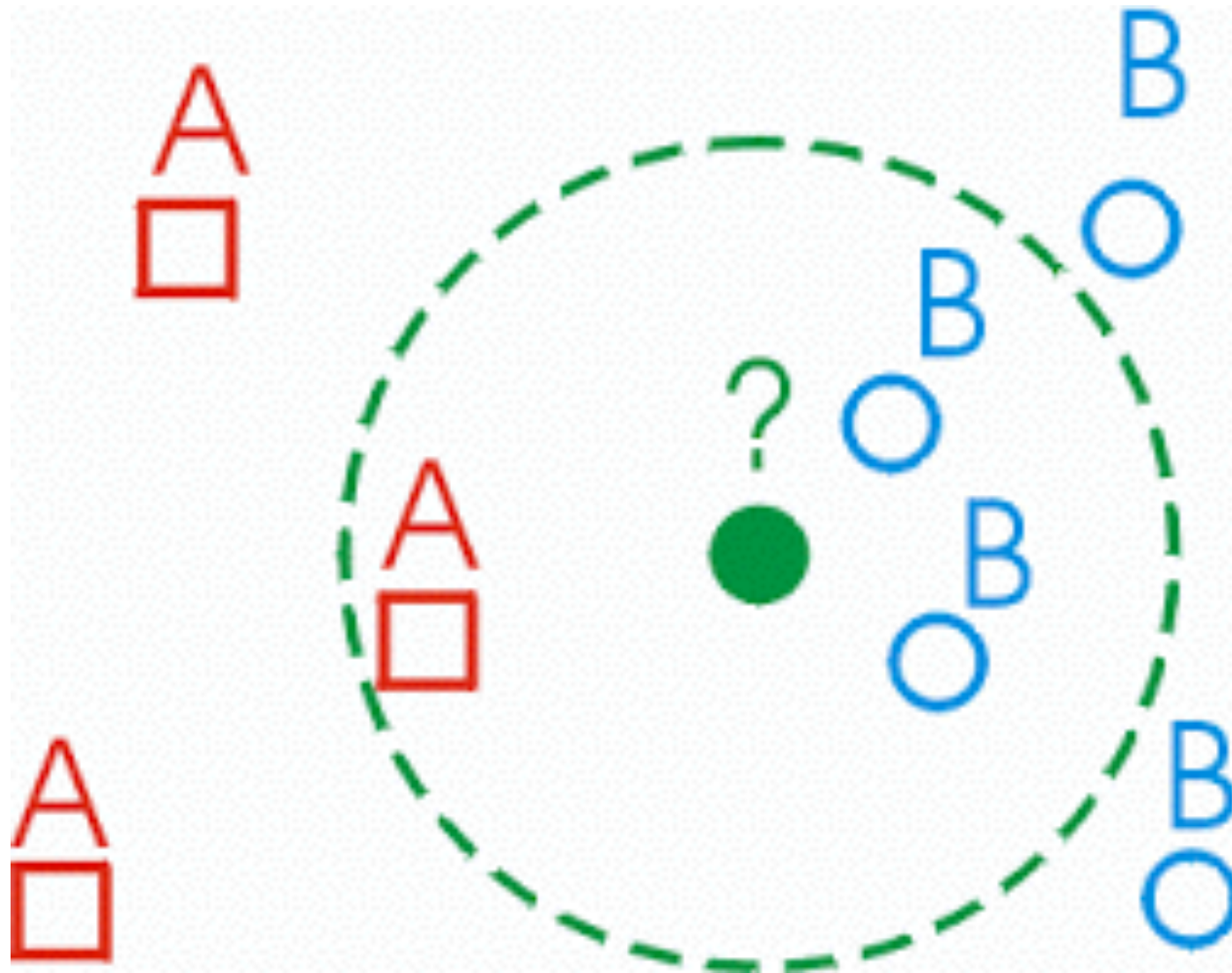
K-Vizinhos Mais Próximos

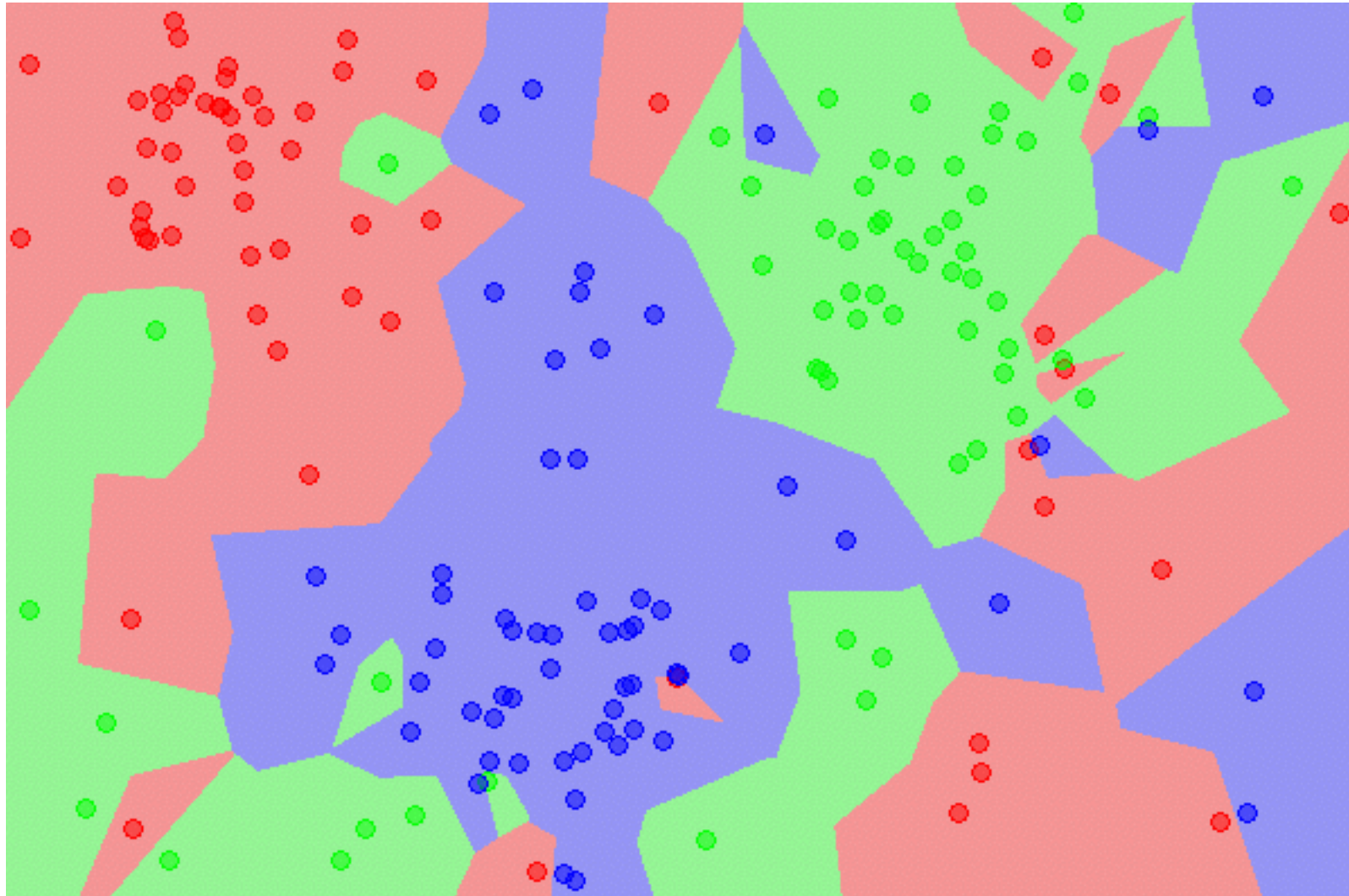
Dados dos alunos dispostos graficamente (em azul, os alunos que passaram, em vermelho, os alunos que não passaram).



KNN

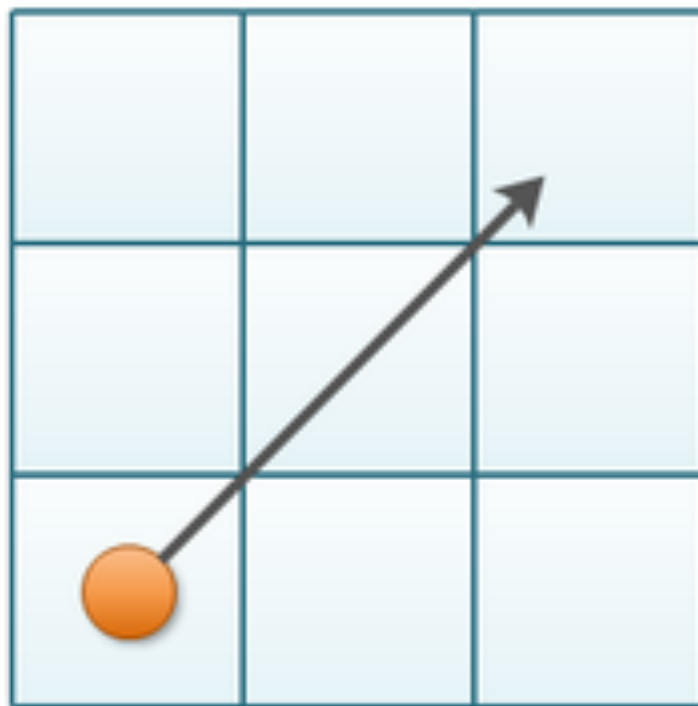


KNN

KNN classifica baseado em Distância

Distância Euclidiana

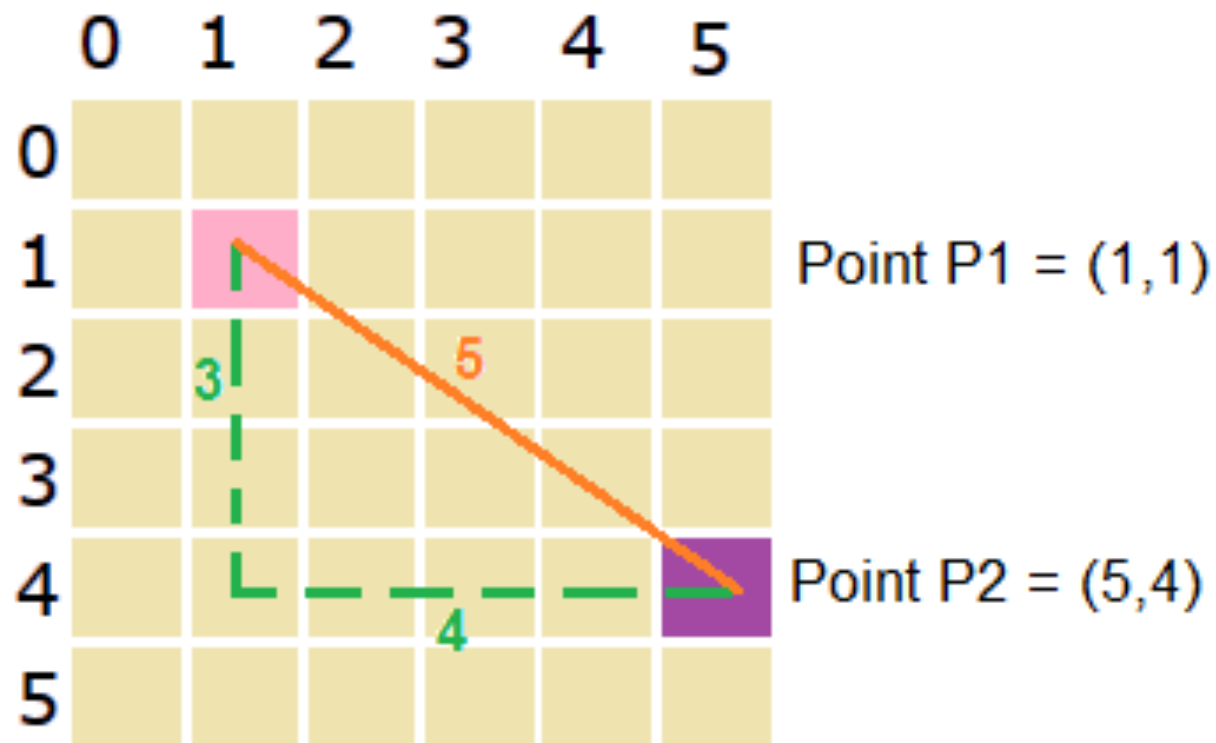
Euclidean Distance



$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$



Distância Manhattan



Euclidean distance = $\sqrt{(5-1)^2 + (4-1)^2} = 5$

Manhattan distance = $|5-1| + |4-1| = 7$

Distância Manhattan

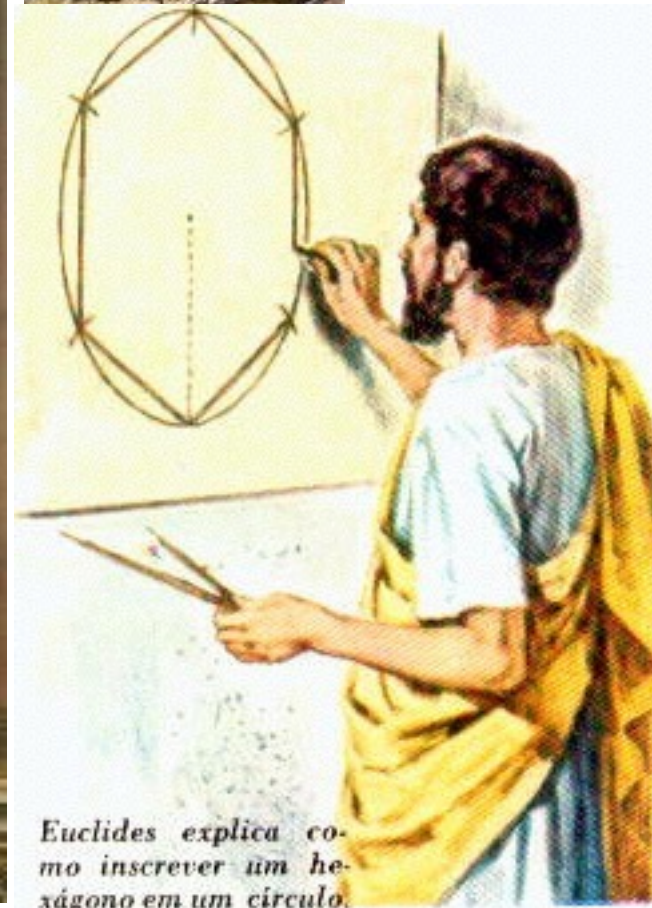
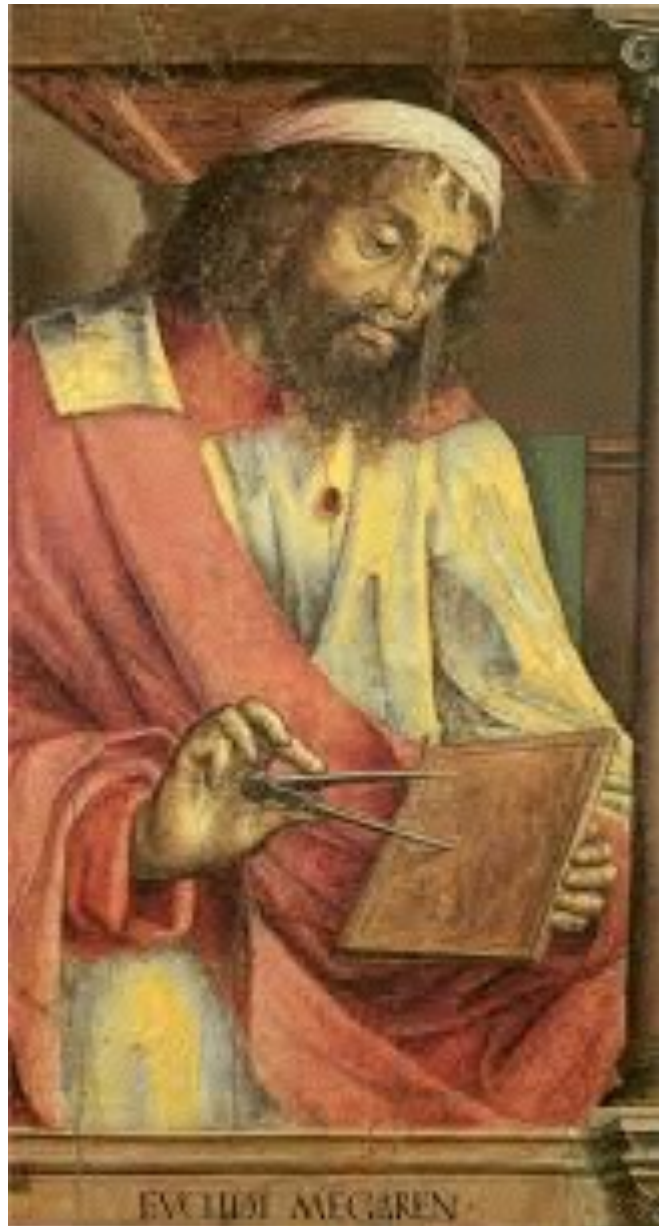


Euclidean



Manhattan

Euclides

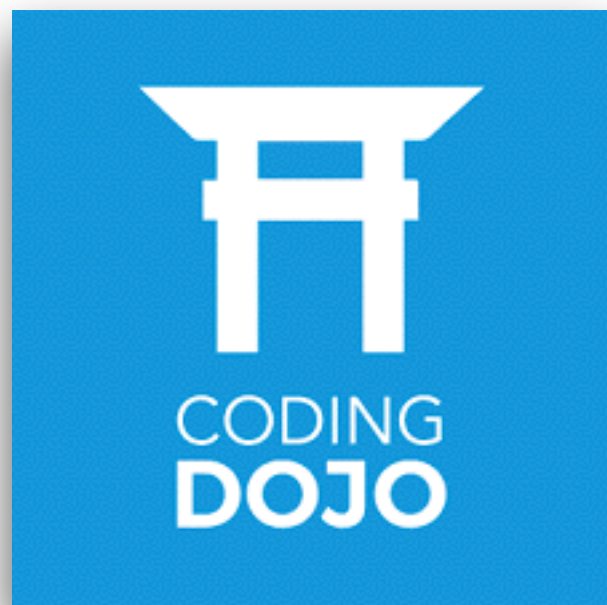


Euclides de Alexandria (em grego antigo: Εὐκλείδης Eukleidēs; fl. c. 300 AC) foi um professor, matemático platônico e escritor possivelmente grego, muitas vezes referido como o "Pai da Geometria". Além de sua principal obra, Os Elementos, Euclides também escreveu sobre perspectivas, seções cônicas, geometria esférica, teoria dos números e rigor.

Implementando KNN do Zero

Lendo os dados do arquivo

```
1 import csv
2 with open('iris.data', 'rb') as csvfile:
3     lines = csv.reader(csvfile)
4     for row in lines:
5         print ', '.join(row)
```



Implementando KNN do Zero



```
import csv
import random
def loadDataset(filename, split, trainingSet=[], testSet=[]):
    with open(filename, 'rb') as csvfile:
        lines = csv.reader(csvfile)
        dataset = list(lines)
        for x in range(len(dataset)-1):
            for y in range(4):
                dataset[x][y] = float(dataset[x][y])
            if random.random() < split:
                trainingSet.append(dataset[x])
            else:
                testSet.append(dataset[x])
```


Implementando KNN do Zero



```
trainingSet=[]  
testSet=[]  
loadDataset('iris.data', 0.66, trainingSet, testSet)  
print 'Train: ' + repr(len(trainingSet))  
print 'Test: ' + repr(len(testSet))
```

```
import math  
def euclideanDistance(instance1, instance2, length):  
    distance = 0  
    for x in range(length):  
        distance += pow((instance1[x] - instance2[x]), 2)  
    return math.sqrt(distance)
```

Implementando KNN do Zero



```
1 import operator
2 def getNeighbors(trainingSet, testInstance, k):
3     distances = []
4     length = len(testInstance)-1
5     for x in range(len(trainingSet)):
6         dist = euclideanDistance(testInstance, trainingSet[x], length)
7         distances.append((trainingSet[x], dist))
8     distances.sort(key=operator.itemgetter(1))
9     neighbors = []
10    for x in range(k):
11        neighbors.append(distances[x][0])
12    return neighbors
```

Implementando KNN do Zero



We can test out this function as follows:

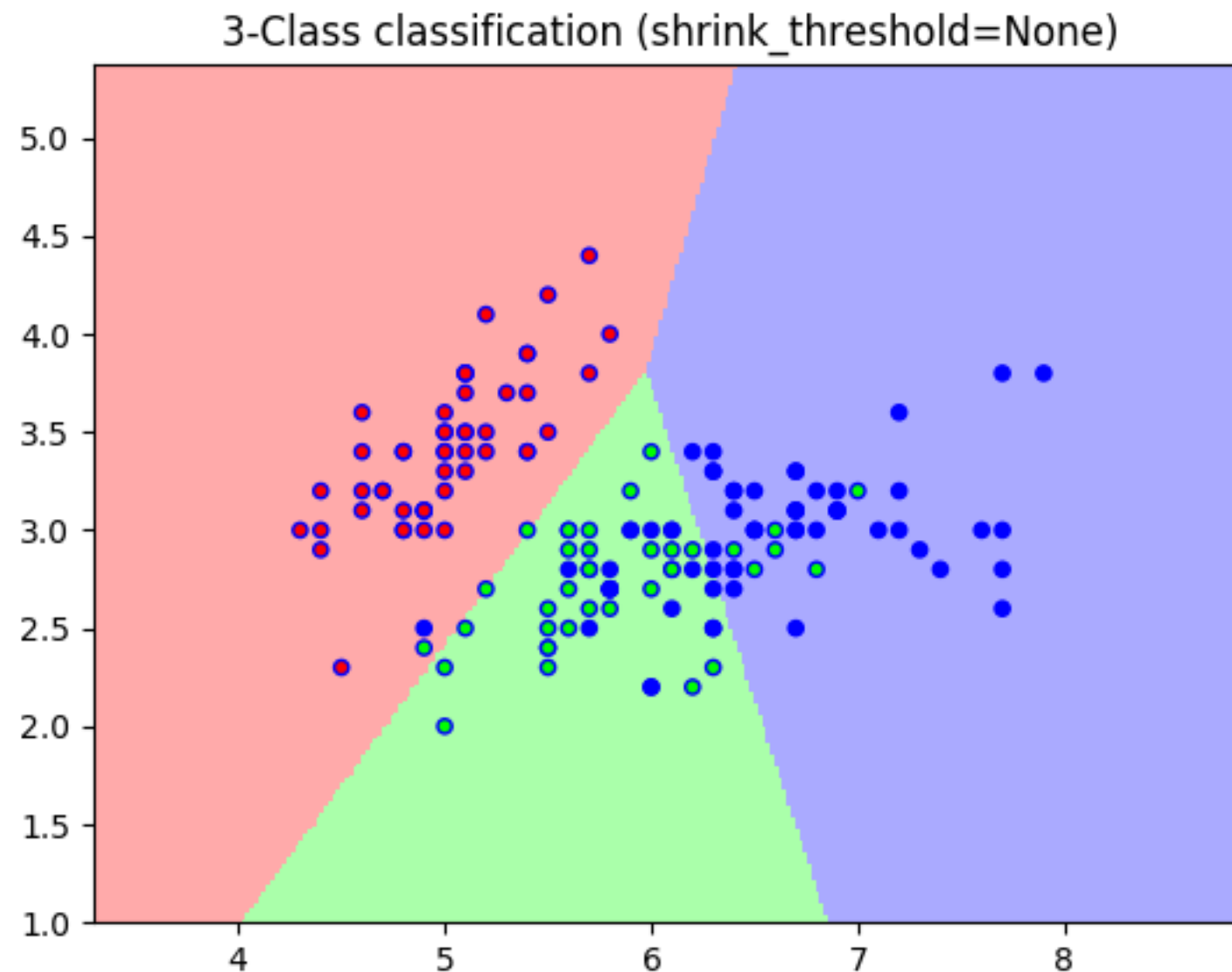
```
1 trainSet = [[2, 2, 2, 'a'], [4, 4, 4, 'b']]
2 testInstance = [5, 5, 5]
3 k = 1
4 neighbors = getNeighbors(trainSet, testInstance, 1)
5 print(neighbors)
```


Implementando KNN com SKLearn



```
>>> from  
sklearn.neighbors.nearest_centro  
id import NearestCentroid  
>>> import numpy as np  
>>> X = np.array([[-1, -1], [-2, -1],  
[-3, -2], [1, 1], [2, 1], [3, 2]])  
>>> y = np.array([1, 1, 1, 2, 2, 2])  
>>> clf = NearestCentroid()  
>>> clf.fit(X, y)  
NearestCentroid(metric='euclidean', shrink_threshold=None)  
>>> print(clf.predict([[-0.8, -1]]))  
[1]
```

Implementando KNN com SKLearn



Implementando KNN do Zero



```
import numpy as np
import pylab as pl
from sklearn import neighbors, datasets

# import some data to play with
iris = datasets.load_iris()
X = iris.data[:, :2] # we only take the first two features.
Y = iris.target

h = .02 # step size in the mesh

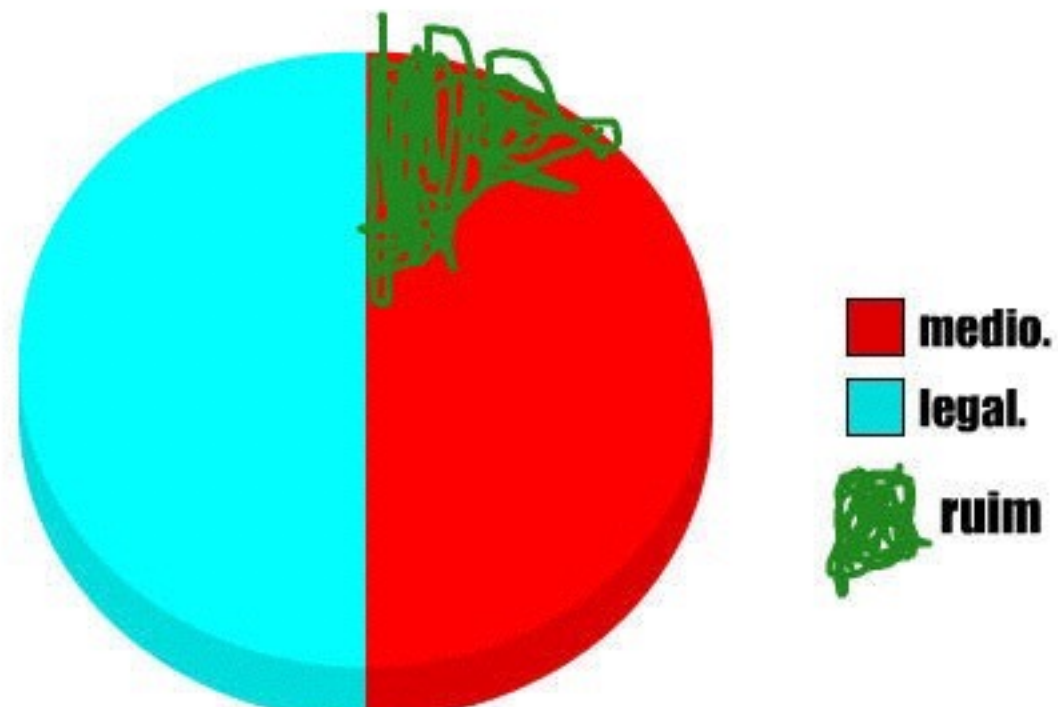
knn=neighbors.KNeighborsClassifier()
# we create an instance of Neighbours
Classifier and fit the data.
knn.fit(X, Y)
```


Alo Mundo



```
>>> plt.plot((1,2,3,4))  
[<matplotlib.lines.Line2D object at 0x8fd48ac>]  
>>> plt.ylabel(u'Alguns Números')  
>>> plt.show()
```

gráfico para os memes



Numpy Linespace

Numpy linspace - Retorna números em um intervalo especificado

```
>>> np.linspace(2.0, 3.0, num=5)
array([ 2. ,  2.25,  2.5 ,  2.75,  3. ])
>>> np.linspace(2.0, 3.0, num=5, endpoint=False)
array([ 2. ,  2.2,  2.4,  2.6,  2.8])
>>> np.linspace(2.0, 3.0, num=5, retstep=True)
(array([ 2. ,  2.25,  2.5 ,  2.75,  3. ]), 0.25)
```

Alo Mundo



```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> x = np.linspace(0,1,num=100)
>>> y = x
>>> plt.plot(x,y,'r--')
[<matplotlib.lines.Line2D object at 0xa6f1c8c>]
>>> z = [t**2 for t in x]
>>> plt.plot(x,z,'bo')
[<matplotlib.lines.Line2D object at 0xa6fac4c>]
>>> w = [t**3 for t in x]
>>> plt.plot(x,w,'g^')
[<matplotlib.lines.Line2D object at 0x9eac40c>]
>>> plt.show()
```


Configurando eixos de um gráfico



Por exemplo, para especificar que o eixo x varia de 0 a 1 enquanto o eixo y varia de 0 a 10, a instrução é:

```
plt.axis((0,1,0,10))
```

```
plt.show()
```

Configurando

```
>>> import matplotlib.pyplot as plt
>>> plt.plot((3,4,5),
c='#FFCC00', lw=3, marker='o',
ms=12,\
mfc='r', mec='b', mew=3,
drawstyle='steps-mid')
>>> plt.axis((-0.1, 2.1, 2.9, 5.1))
>>> plt.show()
```



Propriedade	Nome	Valores
Cor da linha	color ou c	Qualquer cor da <i>string</i> de formatação. Alternativamente pode-se expressar a cor como RGB na forma '#FFDD33'
Estilo da linha	linestyle ou ls	Qualquer valor de estilo de linha da string de formatação. Por exemplo: '-' para linhas traço-ponto.
Largura da linha	linewidth ou lw	Valor inteiro indicando o número de pontos de largura.
Estilo do desenho da linha	drawstyle	Os valores possíveis são: 'default' para pontos unidos por uma reta. 'steps-pre' para linha horizontal iniciando no ponto inicial. 'steps-mid' linha horizontal com um salto entre os pontos, e 'steps-post' para linha horizontal partindo do ponto final.
Formato da marcação de um ponto.	marker	Qualquer valor da <i>string</i> de formatação.
Cor da linha de contorno da marcação do ponto.	markeredgecolor ou mec	Qualquer cor da <i>string</i> de formatação. Alternativamente pode-se expressar a cor como RGB na forma '#FFDD33'
Largura da linha de contorno da marcação do ponto.	markeredgewidth ou mew	O número de pixels de largura.
Cor da marcação	markerfacecolor	Qualquer cor da <i>string</i> de formatação. Alternativamente pode-se expressar a cor como RGB na forma '#FFDD33'
Tamanho da marcação dos pontos.	markersize ou ms	Tamanho da marcação em <i>pixels</i> .
Preenchimento da marcação de pontos.	fillstyle	Os valores possíveis são 'full' para um marcador totalmente preenchido, ou 'left', 'right', 'bottom' e 'top' para marcadores preenchidos pela metade.

Criando dois gráficos

```
1. # program graficos2.py
2. # coding: utf-8
3. import matplotlib.pyplot as plt
4. plt.figure(1) # a primeira janela
5. plt.subplot(2,1,1) # o primeiro gráfico na primeira
   janela
6. plt.plot((1,2,3))
7. plt.subplot(2,1,2) # o segundo gráfico na primeira
   janela
8. plt.plot((4,5,6))
9.
10.
11. plt.figure(2) # uma segunda janela
12. plt.plot((4,5,6)) # cria o gráfico em subplot(1,1,1
   ) por padrão
13.
14. plt.figure(1) # torna a janela 1 a janela
   corrente; subplot(2,1,2) ainda é o gráfico corrente
15. plt.subplot(2,1,1) # faz subplot(2,1,1) na janela 1
   o gráfico corrente
16. plt.title(u'fácil como 1,2,3') # Título do gráfico 2,1,1
17. plt.show()
```

Dúvidas: naubergois@gmail.com

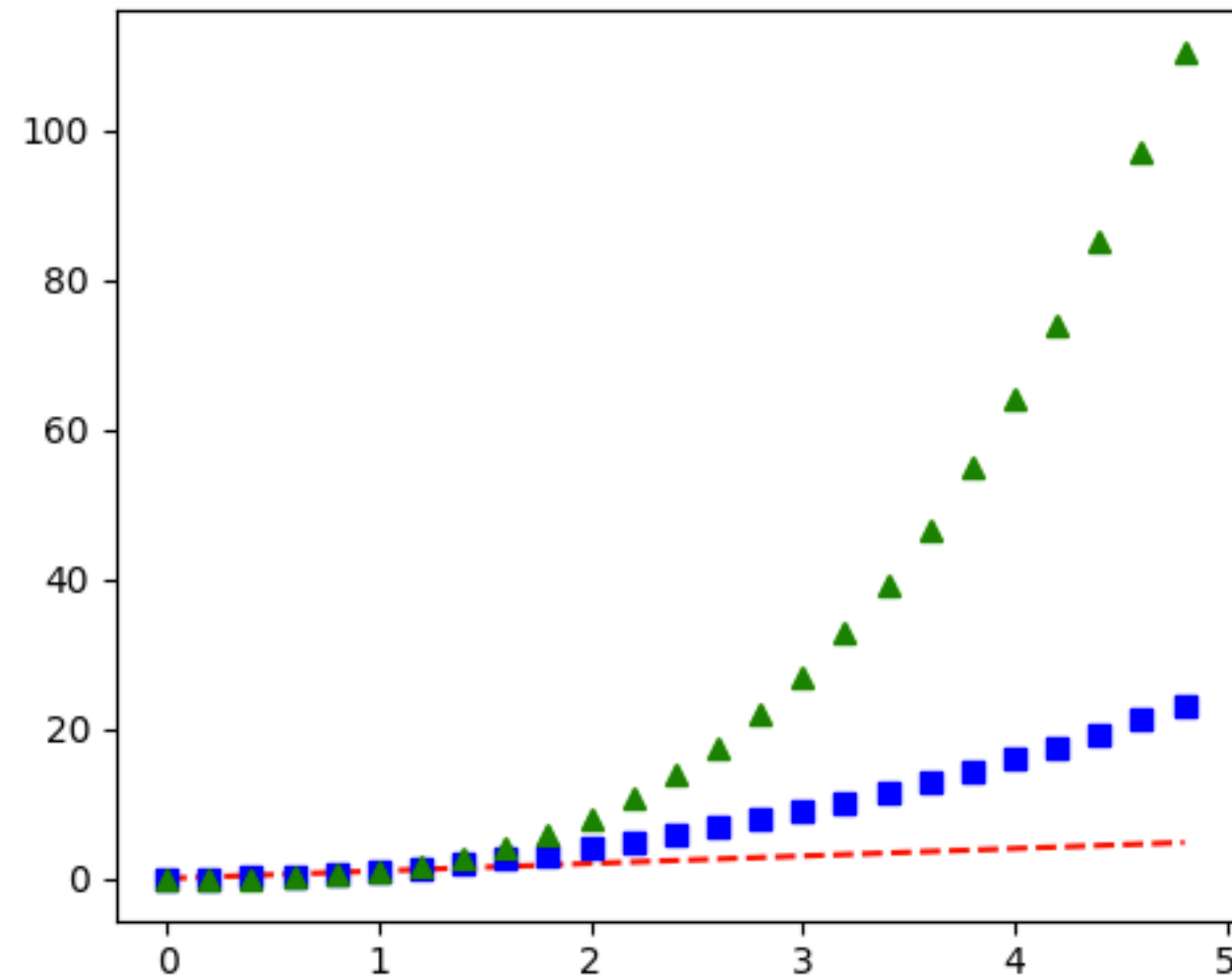


Gráfico com arange

```
import numpy as np
import matplotlib.pyplot as plt

# evenly sampled time at 200ms
# intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green
# triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```

CODING
DOJO

Múltiplas Figuras

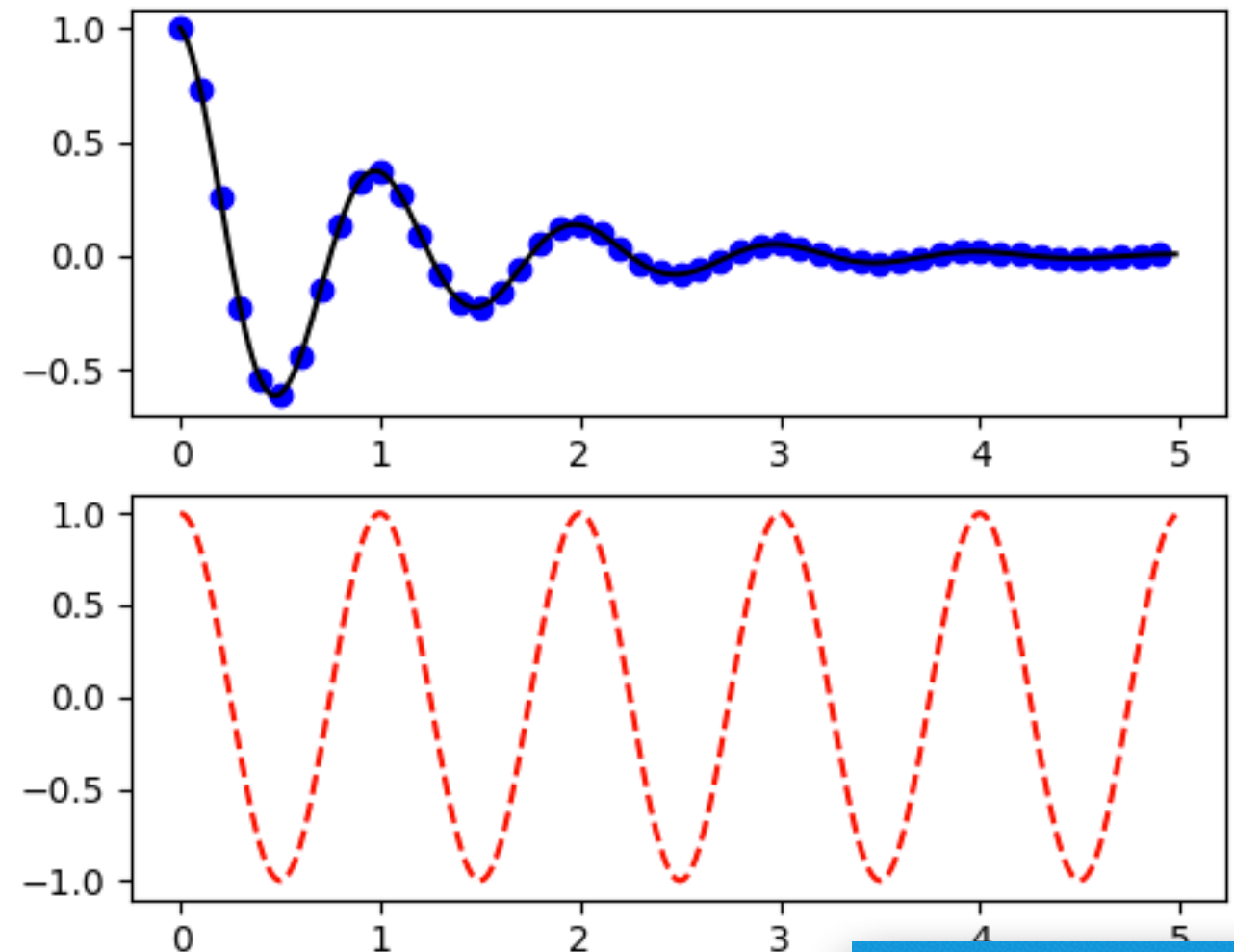
```
import numpy as np
import matplotlib.pyplot as plt

def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

plt.figure(1)
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')

plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()
```



Textos e histogramas

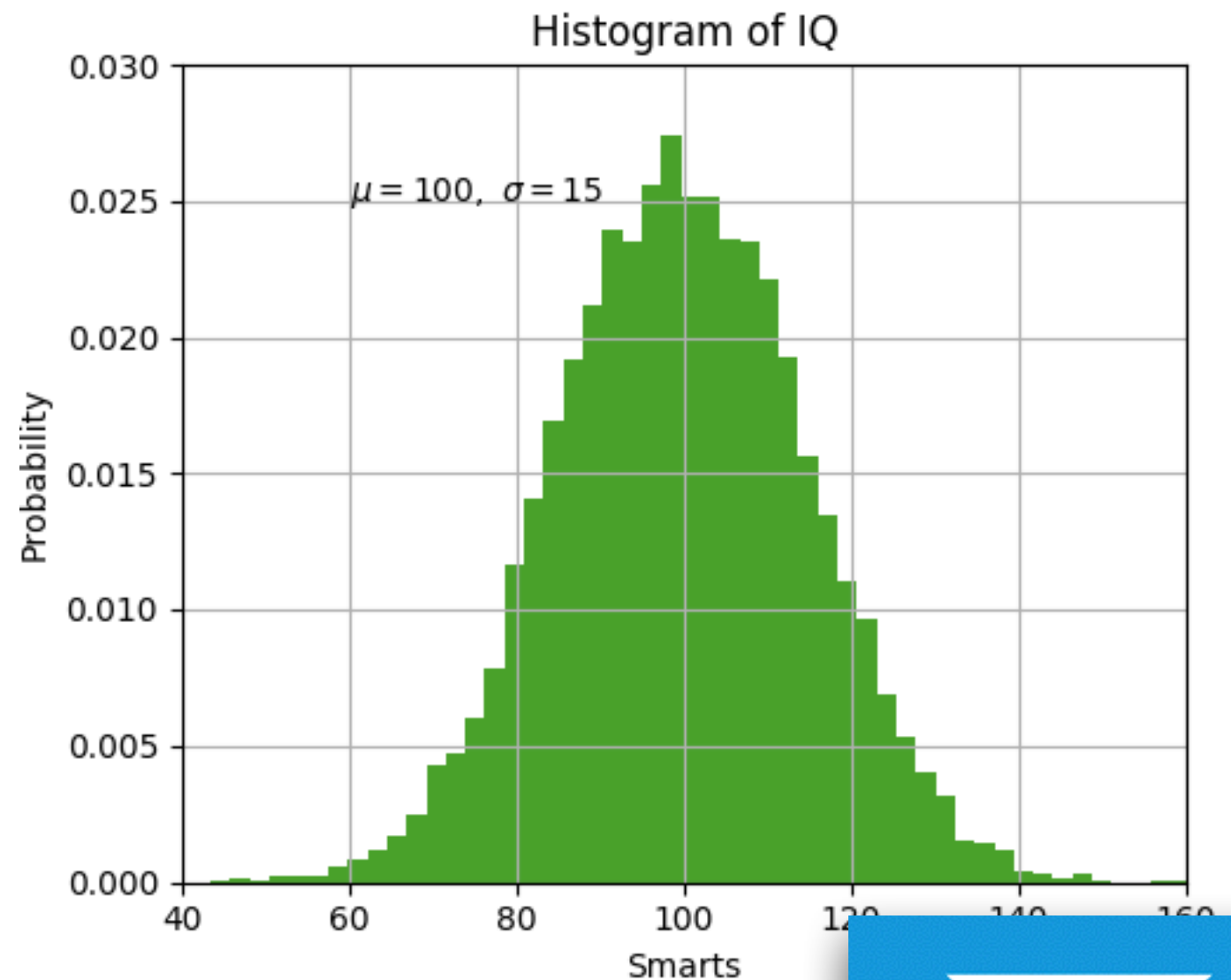
```
import numpy as np
import matplotlib.pyplot as plt

# Fixing random state for reproducibility
np.random.seed(19680801)

mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50, normed=1,
                             facecolor='g', alpha=0.75)

plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```



Anotando texto

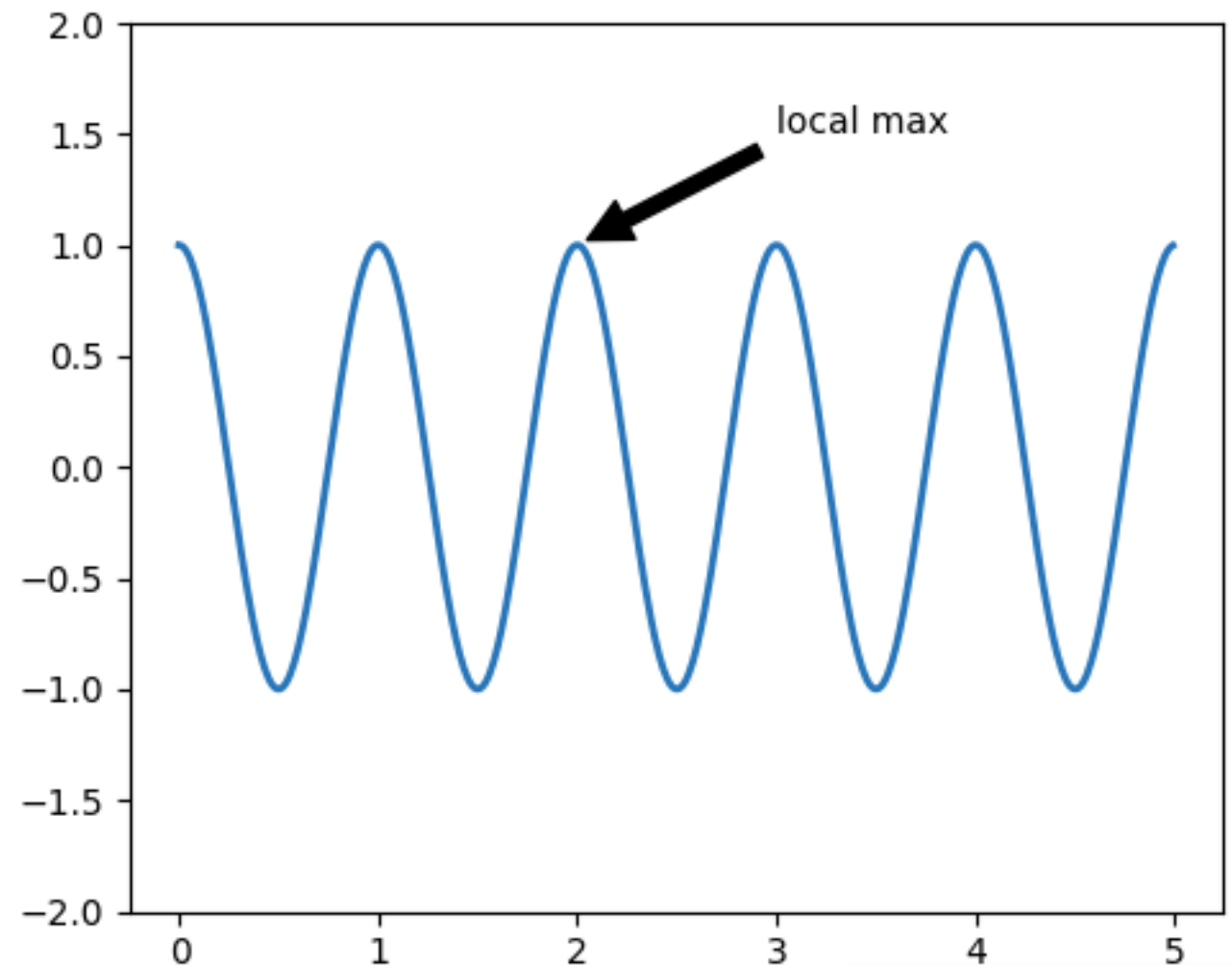
```
import numpy as np
import matplotlib.pyplot as plt

ax = plt.subplot(111)

t = np.arange(0.0, 5.0, 0.01)
s = np.cos(2*np.pi*t)
line, = plt.plot(t, s, lw=2)

plt.annotate('local max', xy=(2, 1),
            xytext=(3, 1.5),
            arrowprops=dict(facecolor='black',
                            shrink=0.05),
            )

plt.ylim(-2,2)
plt.show()
```



Introdução ao TensorFlow

- **Biblioteca para Deep Learning**
- **Biblioteca de código aberto disponibilizada pela Google**
- **Disponibiliza funções primitivas ou tensores para os mais diversos tipos de cálculos.**



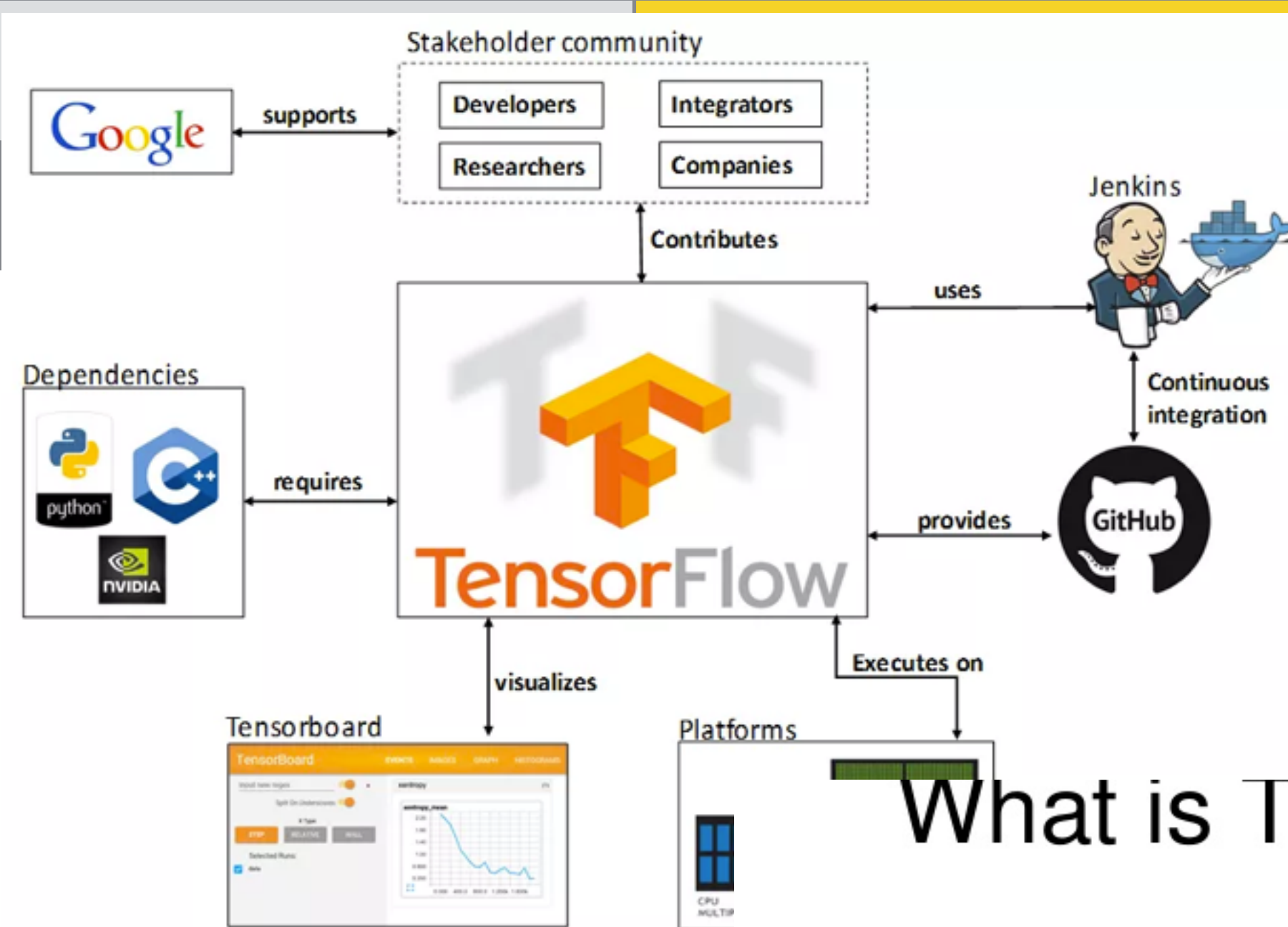
Introdução ao TensorFlow

- **Tensores são mapas multilineares:**
 - **Matrizes**
 - **Vetores**

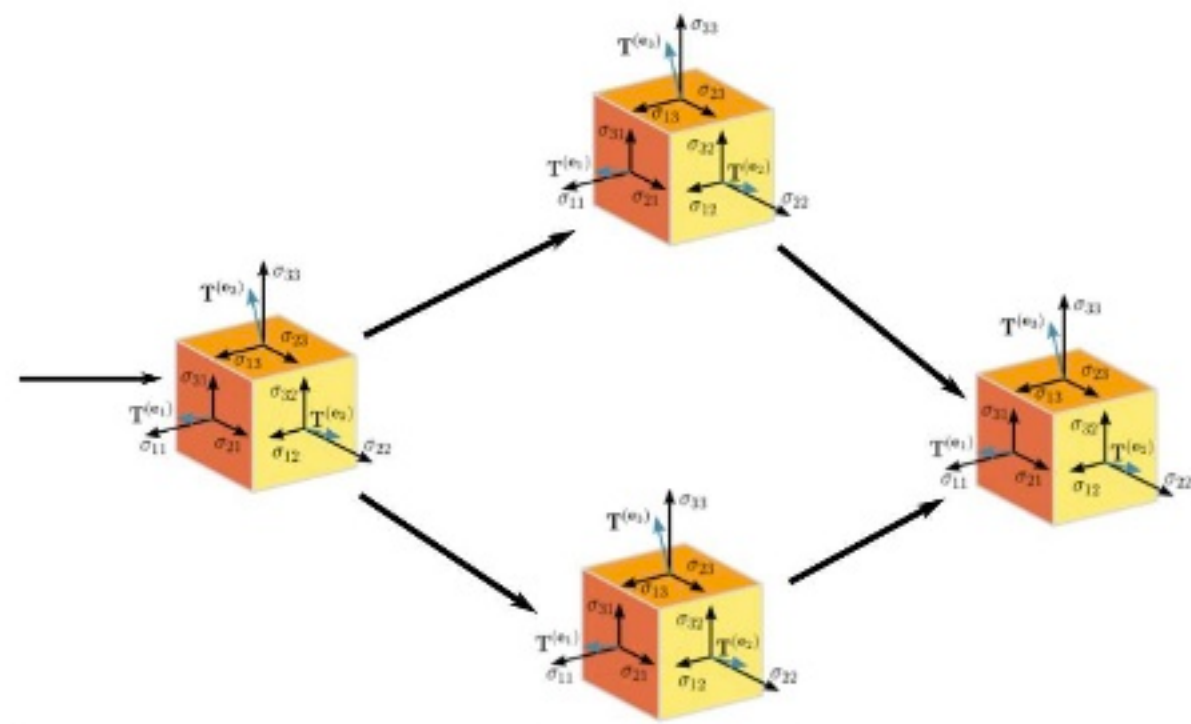


Regressão

ow



What is TensorFlow?



Dúvidas: naubergois@gmail.com

Comparando Tensorflow e Numpy

```
In [23]: import numpy as np
In [24]: a = np.zeros((2,2)); b =
np.ones((2,2))
In [25]: np.sum(b, axis=1)
Out[25]: array([ 2., 2.])
In [26]: a.shape
Out[26]: (2, 2)
In [27]: np.reshape(a, (1,4))
Out[27]: array([[ 0., 0., 0., 0.]])
```



Comparando Tensorflow e Numpy

Numpy to TensorFlow Dictionary

Numpy	TensorFlow
<code>a = np.zeros((2,2)); b = np.ones((2,2))</code>	<code>a = tf.zeros((2,2)), b = tf.ones((2,2))</code>
<code>np.sum(b, axis=1)</code>	<code>tf.reduce_sum(a, reduction_indices=[1])</code>
<code>a.shape</code>	<code>a.get_shape()</code>
<code>np.reshape(a, (1,4))</code>	<code>tf.reshape(a, (1,4))</code>
<code>b * 5 + 1</code>	<code>b * 5 + 1</code>
<code>np.dot(a,b)</code>	<code>tf.matmul(a, b)</code>
<code>a[0,0], a[:,0], a[0,:]</code>	<code>a[0,0], a[:,0], a[0,:]</code>

Recordando a multiplicação de matrizes

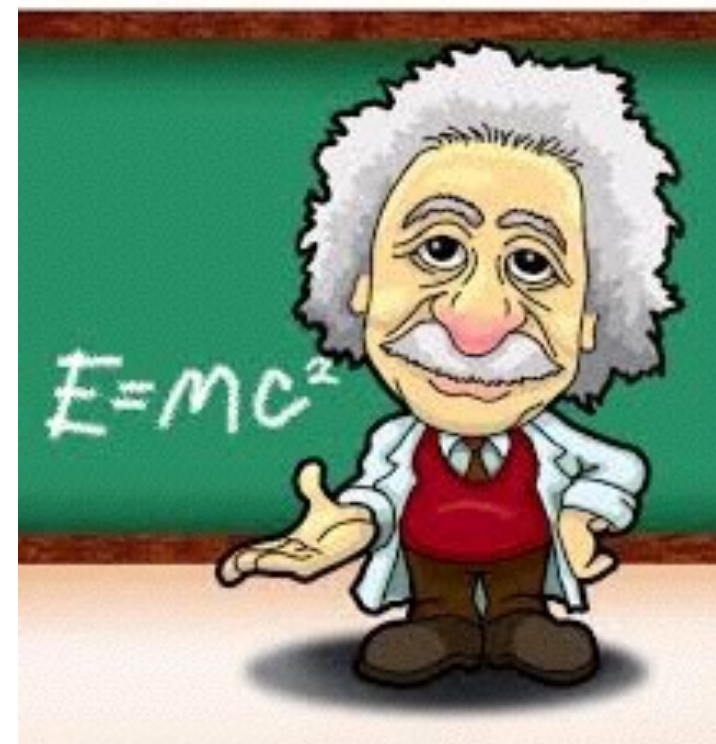
Matriz A Matriz B

$$\begin{bmatrix} 3 & 2 \\ 3 & 3 \\ 1 & 2 \end{bmatrix} \times \begin{bmatrix} 3 & 2 & 6 \\ 1 & 2 & 5 \end{bmatrix}$$

Realizando as operações de multiplicação e adição temos a Matriz C

$$\begin{bmatrix} (3 \times 3) + (2 \times 1) & (3 \times 2) + (2 \times 2) & (3 \times 6) + (2 \times 5) \\ (3 \times 3) + (3 \times 1) & (3 \times 2) + (3 \times 2) & (3 \times 6) + (3 \times 5) \\ (1 \times 3) + (2 \times 1) & (1 \times 2) + (2 \times 2) & (1 \times 6) + (2 \times 5) \end{bmatrix}$$

$$\begin{bmatrix} 11 & 10 & 28 \\ 12 & 12 & 33 \\ 5 & 6 & 16 \end{bmatrix}$$



Iniciando com Tensorflow

```
import tensorflow as tf
```

Importando o TensorFlow

Iniciando com Tensorflow

TensorFlow requires explicit evaluation!

```
In [37]: a = np.zeros((2,2))
```

```
In [38]: ta = tf.zeros((2,2))
```

```
In [39]: print(a)
```

```
[[ 0.  0.]  
 [ 0.  0.]]
```

```
In [40]: print(ta)
```

```
Tensor("zeros_1:0", shape=(2, 2), dtype=float32)
```

```
In [41]: print(ta.eval())
```

```
[[ 0.  0.]  
 [ 0.  0.]]
```

*TensorFlow computations define a **computation graph** that has no numerical value until evaluated!*

Iniciando com Tensorflow

TensorFlow Session Object (1)

- “A Session object encapsulates the environment in which Tensor objects are evaluated” - [TensorFlow Docs](#)


```
In [20]: a = tf.constant(5.0)
```

```
In [21]: b = tf.constant(6.0)
```

```
In [22]: c = a * b
```

```
In [23]: with tf.Session() as sess:  
.....:     print(sess.run(c))  
.....:     print(c.eval())  
.....:
```

c.eval() is just syntactic sugar for sess.run(c) in the currently active session!



```
30.0
```

```
30.0
```

Iniciando com Tensorflow

TensorFlow Variables (2)


```
In [32]: W1 = tf.ones((2,2))
```

```
In [33]: W2 = tf.Variable(tf.zeros((2,2)), name="weights")
```

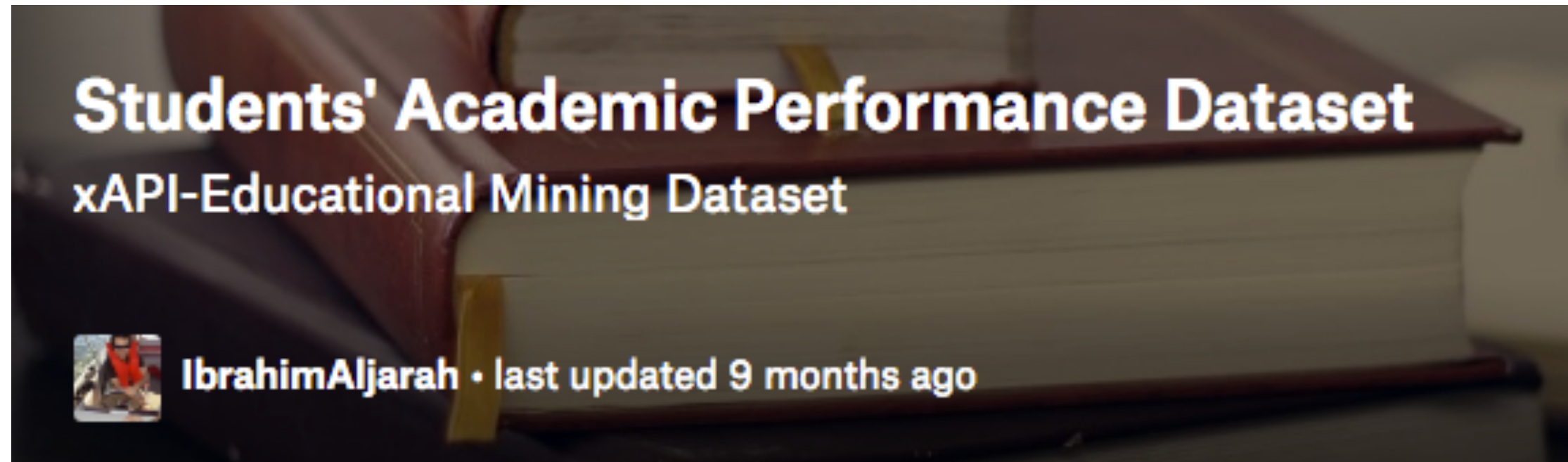
```
In [34]: with tf.Session() as sess:  
        print(sess.run(W1))  
        sess.run(tf.initialize_all_variables())  
        print(sess.run(W2))
```

```
.....:  
[[ 1.  1.]  
 [ 1.  1.]]  
[[ 0.  0.]  
 [ 0.  0.]]
```

Note the initialization step `tf.initialize_all_variables()`



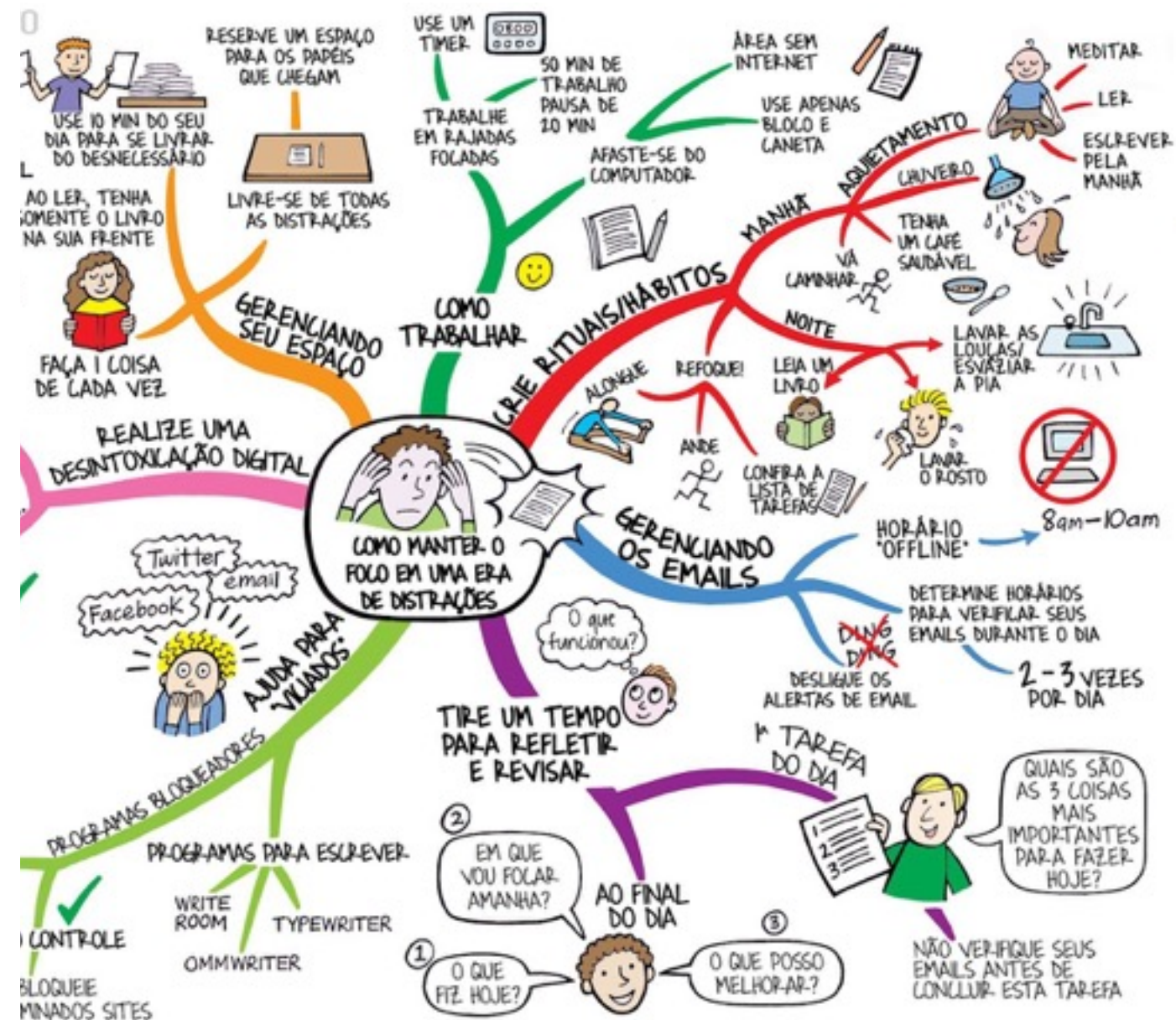
Revisando árvores de decisão

[Overview](#)[Kernels](#)[Discussion](#)[Activity](#)

<https://www.kaggle.com/miya0930/decisiontree-and-randomforest>

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 exubérante à basculer les vivants et les autres animaux sauvages. Fuyant la lumière et les lieux habités, la femelle des Batraciens à queue généralement dégingolée par le nom de Salamandre, se retire dans les profondeurs des montagnes riches en sources et en ombrage. L'existence de différentes espèces de Salamandres au Japon, est d'une haute importance pour l'étude de la distribution géographique de ces reptiles, mais ce qui est plus important encore c'est la découverte d'une espèce de Salamandre, qui par sa taille extraordinaire et sa forme spéciale, 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 fossilifères des terrains tertiaires, et qui vit apparues aux milieux de nos reptiles gigantesques et d'organisations bizarres. Je vais parler de l'époque débris, la célèbre Salamandre fœtale des cantons d'Osaka qui depuis Souchou jusqu'à Omi dont les dents ont été si vite livrées sur le monde primitif, a été l'objet des spéculations des naturalistes.

Notre grande Salamandre (Salamandre japonica) 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 lacs et dans les lieux humides par les eaux pluviales ou même 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'habitat, mais son organisation et ses habitudes la rappellent bientôt dans cet élément, car elle meurt plus facilement que sur terre, une salamandre qui consiste en petits poissons, en grenouilles et en vases. C'est à Sakurata le petit village situé aux pieds du mont Sennaga vers 15 Mi environ à l'est de Misaki, que j'ai observé pour la première fois cette Salamandre. En de mes disciples, le Docteur Enryu, avait chargé un herboriste qui habite cette montagne de faire la recherche de ce rare et curieux animal. Malgré les renseignements des montagnards le Souchouman — c'est le nom indigène vulgaire de la grande Salamandre — se trouve le plus souvent dans les montagnes d'Okudayama. Fui en le loupant des rapporter une vivante en Europe. Elle existe encore au Mont des Pays-Bas, on 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.

