

Bioinformatics
Multidisciplinary
Environment

Centro
Multiusuário
de Bioinformática



Aprendizado de Máquina Aplicada a dados de Câncer

Tetsu Sakamoto
Daniela Coelho B. G. Pereira

Material disponível em:

`git clone https://github.com/danielacbgp/Cancer`



Agenda

✓ Aprendizagem de Máquina

✓ Rede Neural

✓ Keras

✓ SVM

✓ Google Colaboratory

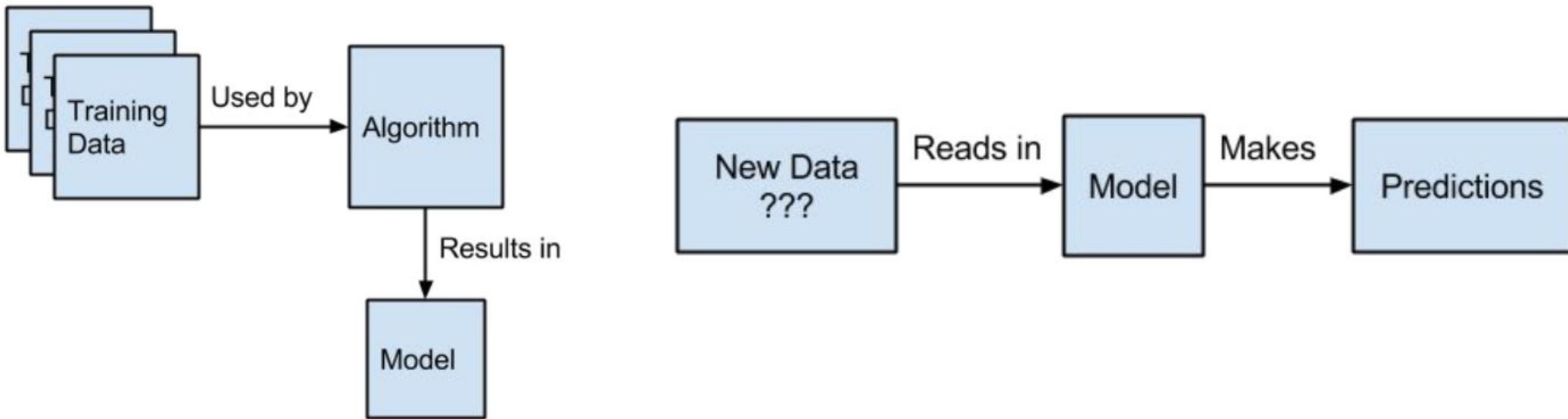
Aprendizagem de máquina



Aprendizagem de Máquina: Campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados

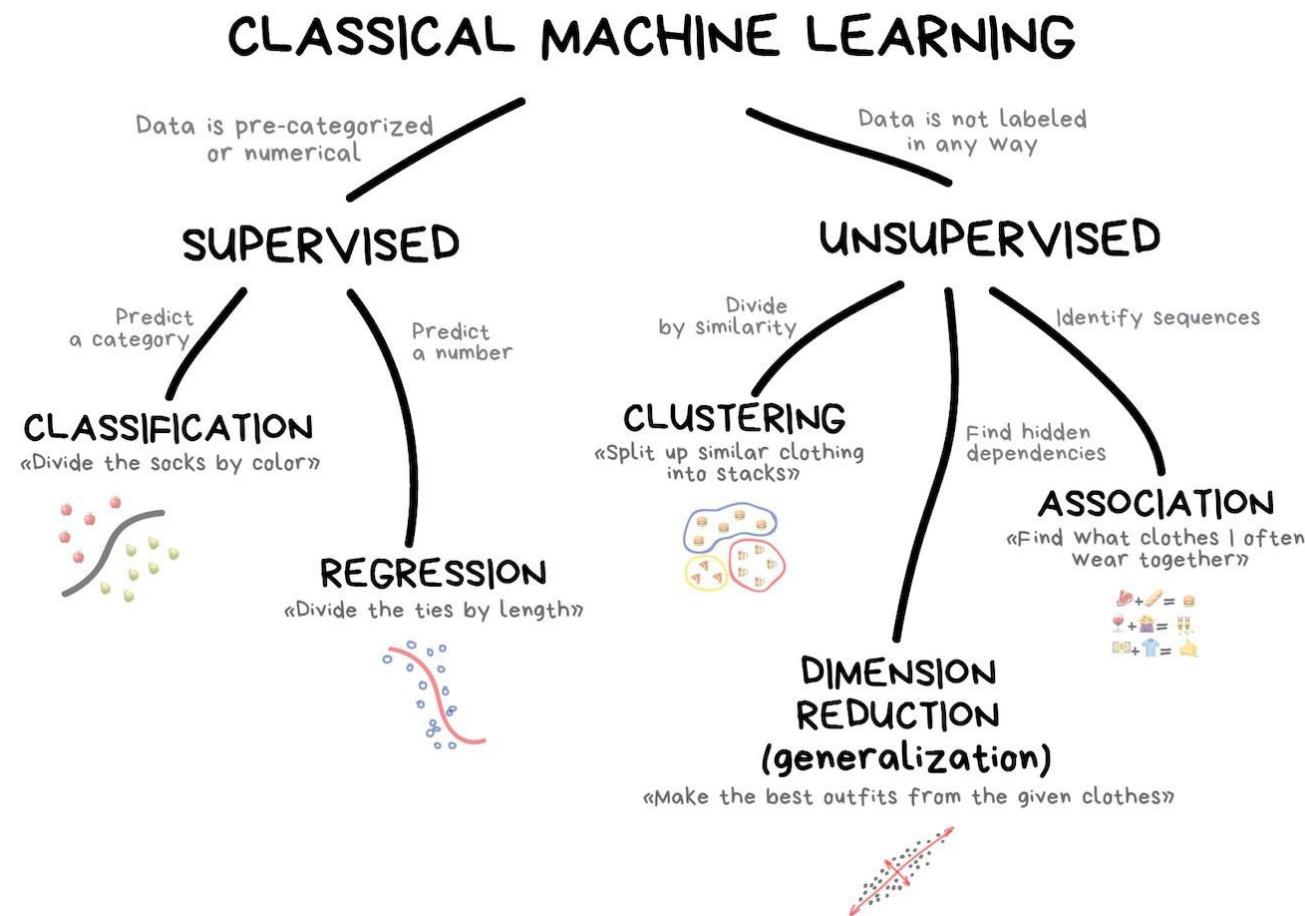
Arthur Samuel, 1959

Aprendizagem de Máquina

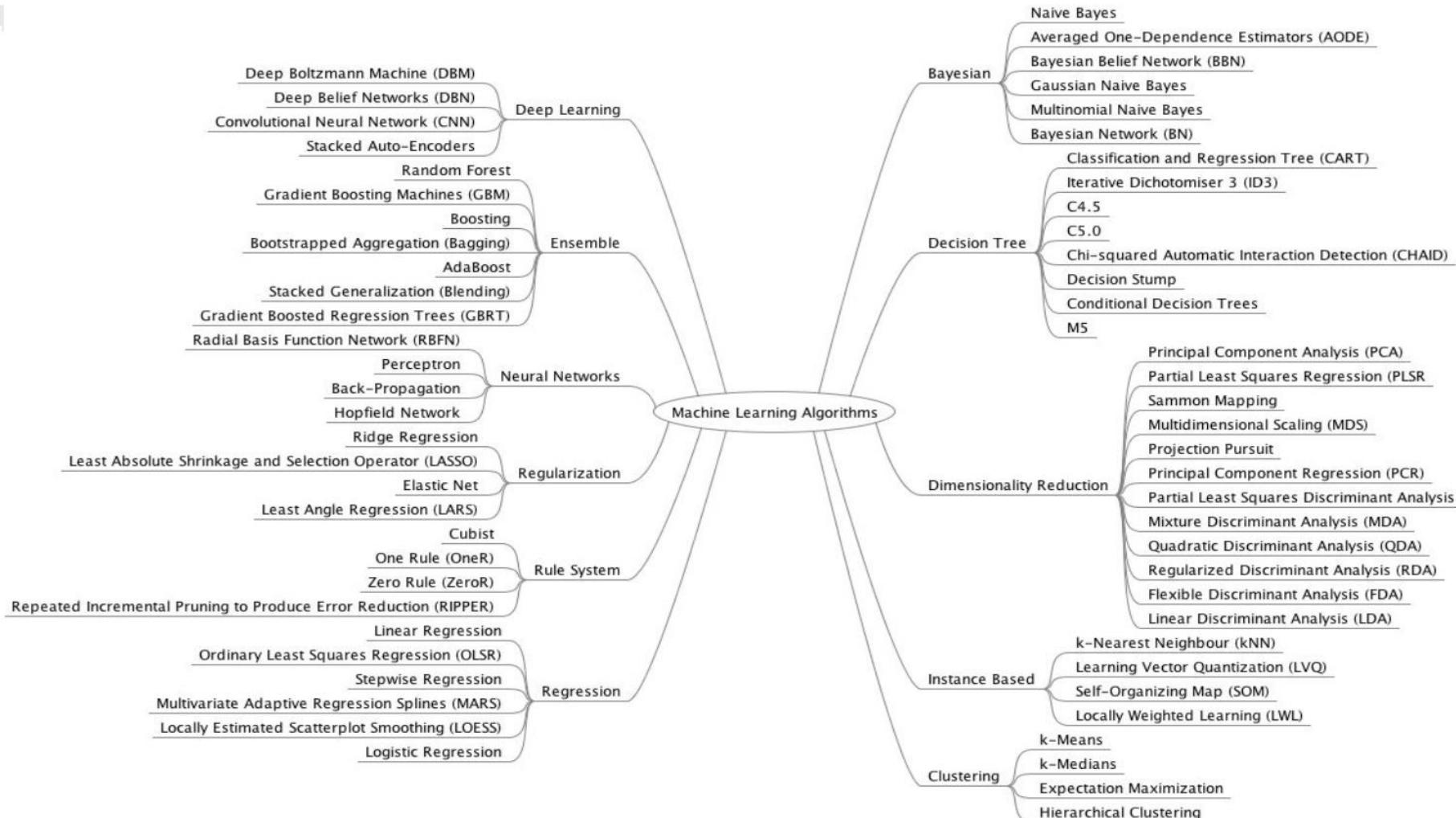


Tais algoritmos operam construindo um modelo a partir de inputs amostrais a fim de fazer previsões ou decisões guiadas pelos dados ao invés de simplesmente seguindo inflexíveis e estáticas instruções de programação tradicional

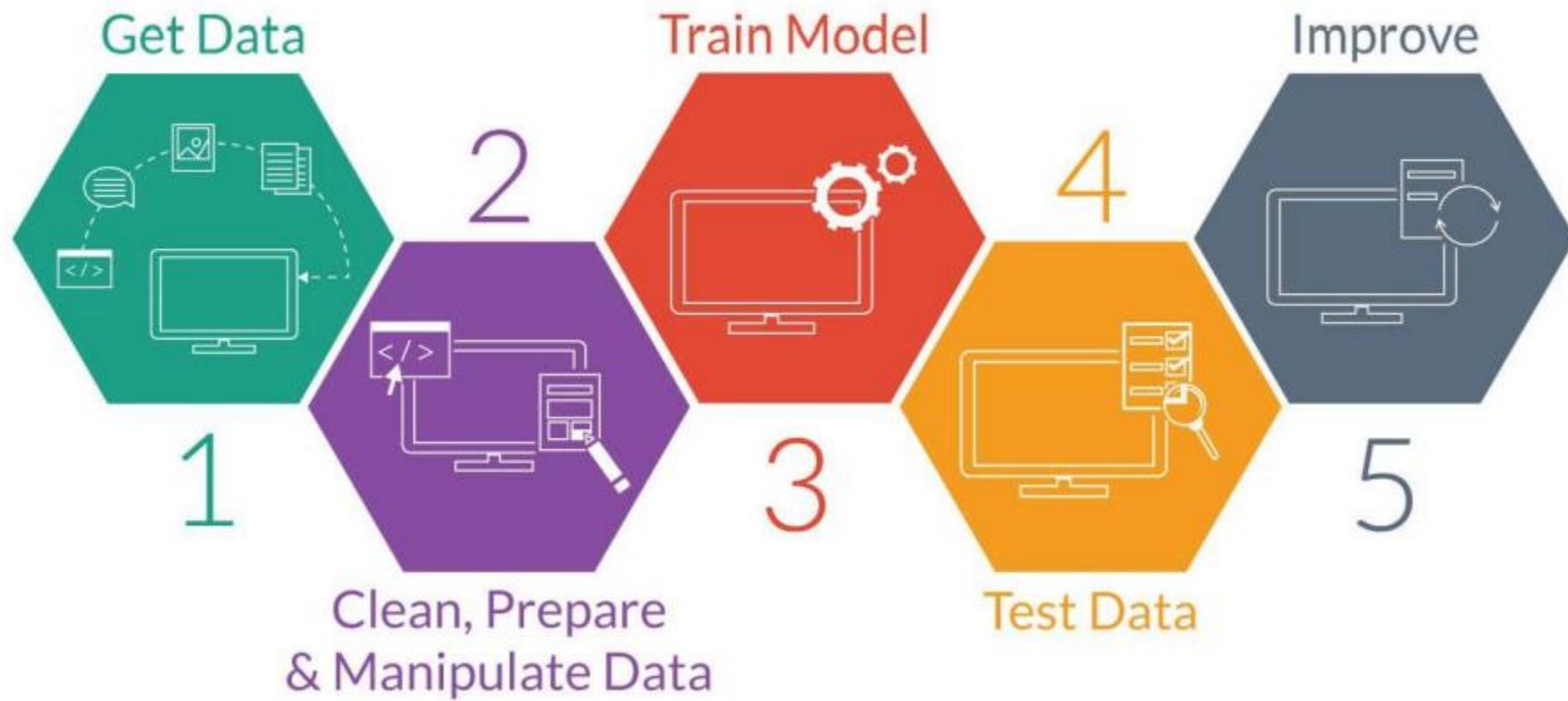
Tipos de Sistemas de Aprendizagem de Máquina



Algoritmos de Aprendizagem de Máquina



A general ML workflow

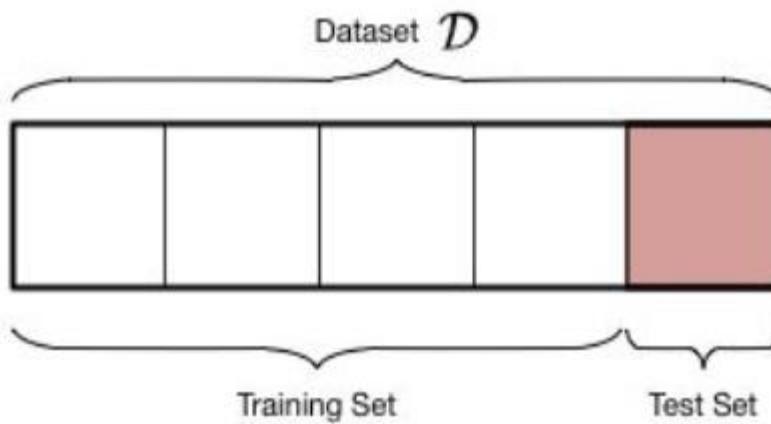


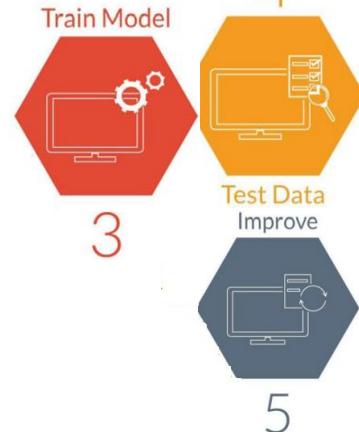


- A mineração dos dados é o processo que transforma os dados brutos (dados do mundo real) em um formato comprehensível pelos algoritmos de aprendizagem de máquina.
- Geralmente, é necessário realizar um pré-processamento de todos os dados do *dataset*, antes de submetê-los aos modelos de aprendizagem de máquina.
- Toda esta parte de mineração de dados pode ser implementada nas bibliotecas em Python:
 - Pandas (fornece ferramentas de análise de dados e estruturas de dados de alta performance)
 - Numpy (fornece ferramentas para trabalhar com arrays multidimensionais de alto desempenho).
 - Scikit-learn, uma biblioteca de aprendizagem de máquina que possui várias ferramentas para mineração e análise de dados.

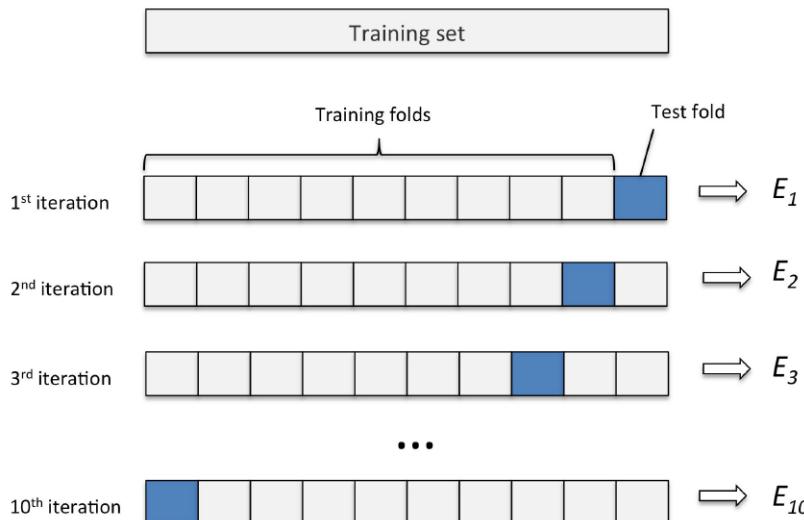


- Divisão do dataset em 2 sets: Treinamento e Teste





Grid Search com Cross Validation



```
from sklearn.model_selection import GridSearchCV
```

```
param_grid = {'n_estimators': np.arange(10, 100, 10),
              'max_depth': np.arange(2, 20),
              'max_features': ['auto', 'log2', None],
              'criterion': ['gini', 'entropy'],
              'random_state': [42],
              'class_weight': ['balanced']}
```

```
# create the grid
```

```
grid_forest = GridSearchCV(RandomForestClassifier(),
                            param_grid, cv = 5,
                            n_jobs=-1, scoring ='accuracy')
```

```
#training
```

```
%time grid_forest.fit(train_sem_X, train_sem_y)
```

```
#best estimator
```

```
best_forest = grid_forest.best_estimator_
print(best_forest)
```

```
#score
```

```
print("Score da aprendizagem",np.abs(grid_forest.best_score_))
```

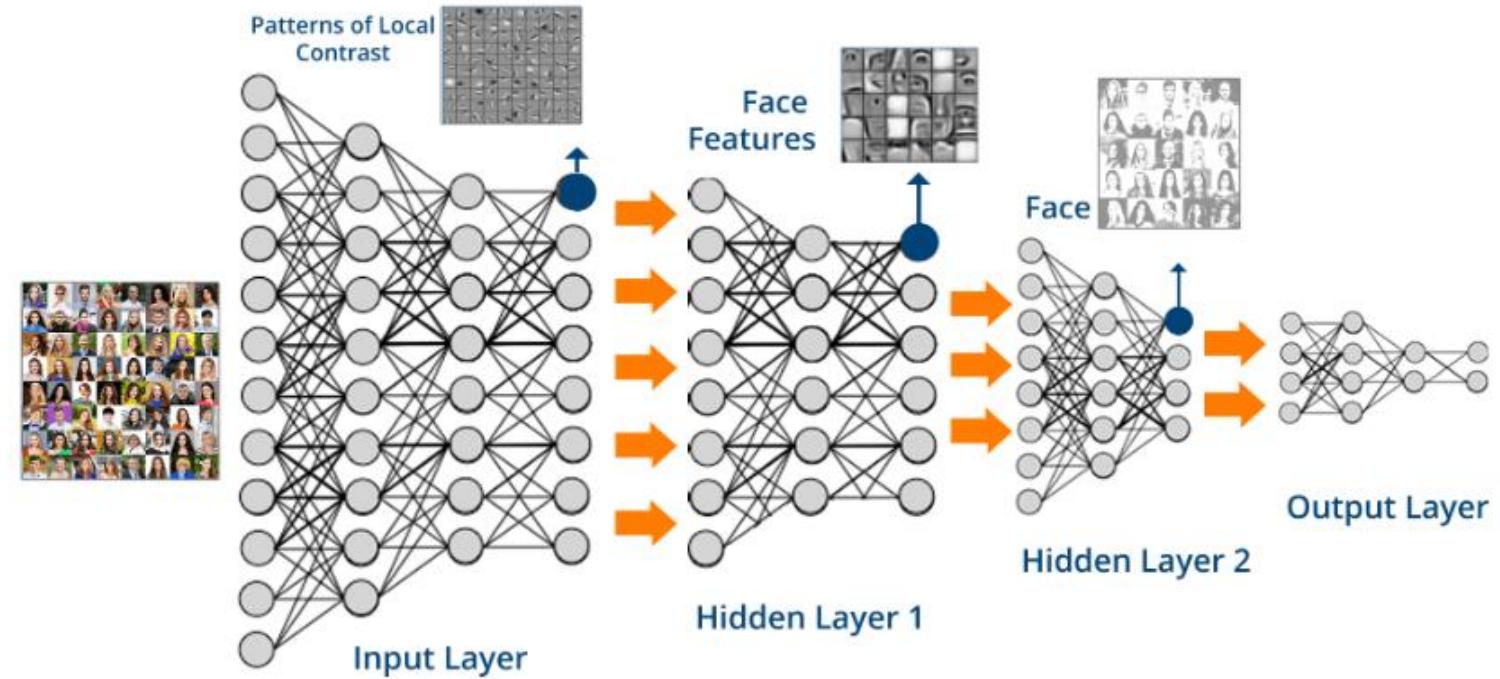
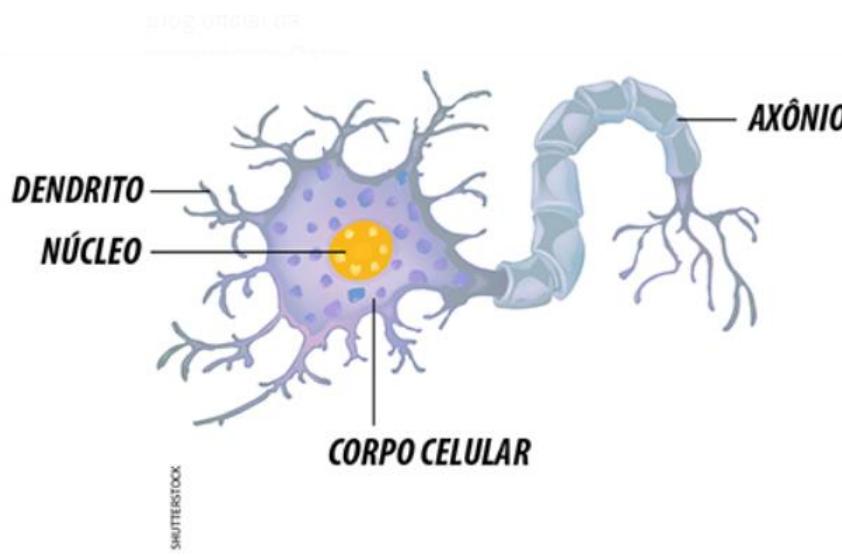
```
#accuracy sobre a base test para uma classificação ternária
```

```
predictions = best_forest.predict(test_sem_X)
```

```
accuracy = accuracy_score(y_true = test_sem_y, y_pred = predictions)
```

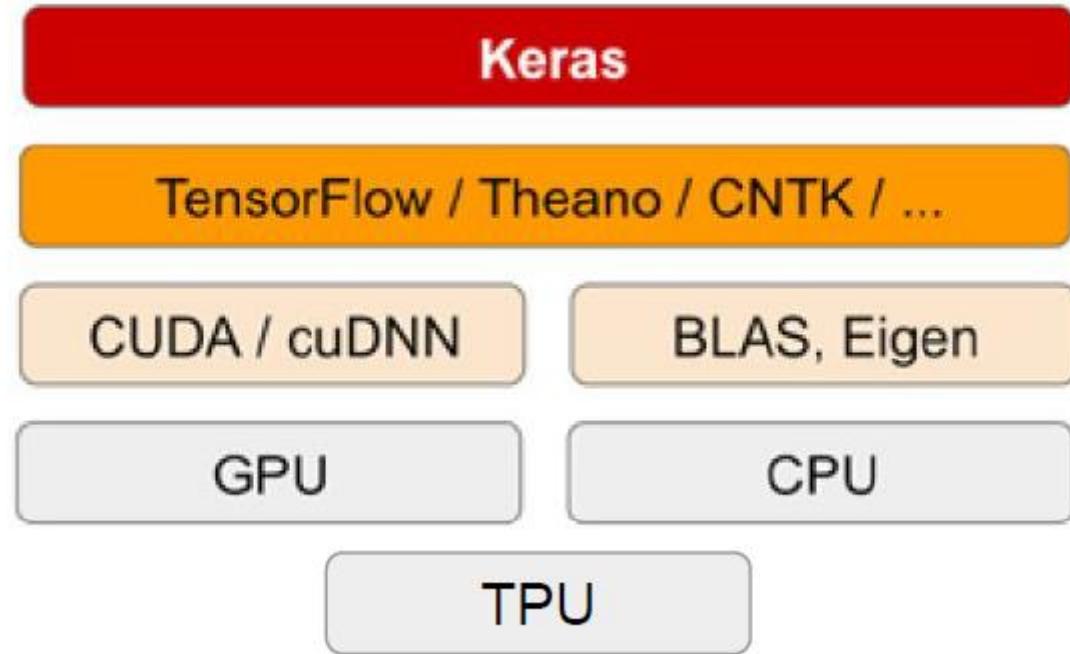
```
print("Acurácia do teste via accuracy_score:",accuracy)
```

Rede Neural



Os modelos de redes neurais foram inspirados na estrutura dos neurônios no nosso cérebro e nas mensagens passando entre neurônios

O que é Keras?



Características:

- Permite que o mesmo código seja executado sem problemas em CPU, GPU ou TPU.
- Suporta vários tipos de arquiteturas de rede.
- Compatível com qualquer versão do Python de 2.7 em diante.
- Promove o desenvolvimento rápido de modelos de rede e de fácil entendimento.

<https://keras.io/>

Keras - Workflow

□ Especificação da Arquitetura

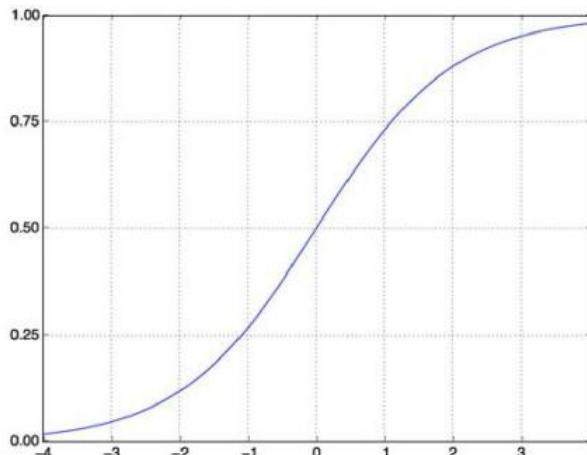
- Quantas camadas?
- Quantos nós em cada camada?
- Qual função de ativação será usado em cada camada?

```
In [1]: import numpy as np  
  
In [2]: from keras.layers import Dense  
  
In [3]: from keras.models import Sequential  
  
In [4]: predictors = np.loadtxt('predictors_data.csv', delimiter=',')  
  
In [5]: n_cols = predictors.shape[1]  
  
In [6]: model = Sequential()  
  
In [7]: model.add(Dense(100, activation='relu', input_shape = (n_cols,)))  
  
In [8]: model.add(Dense(100, activation='relu'))  
  
In [9]: model.add(Dense(1))
```

Keras - Workflow

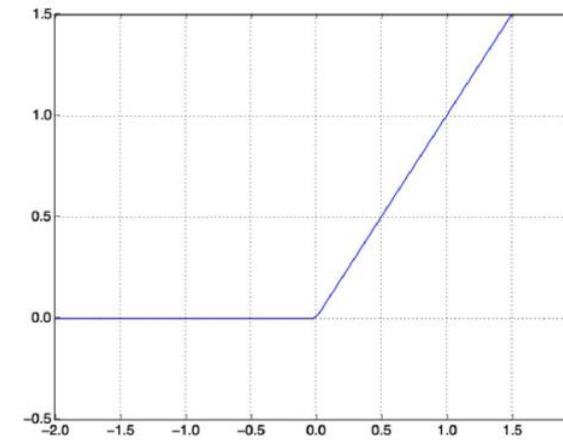
- As funções de ativação são um elemento extremamente importante das redes neurais artificiais. Elas basicamente decidem se um neurônio deve ser ativado ou não. Ou seja, se a informação que o neurônio está recebendo é relevante para a informação fornecida ou deve ser ignorada.
- As funções de ativação mais usadas em redes neurais são:

A função sigmoid



A função tenta empurrar os valores de Y para os extremos, qualidade desejável quando tentamos classificar os valores para uma classe específica.

A função ReLU

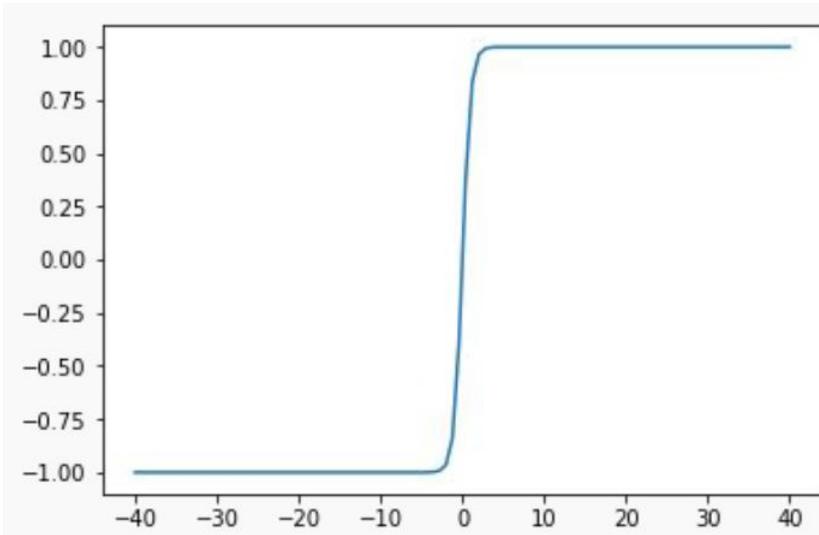


Se a entrada for negativa, ela será convertida em zero e o neurônio não será ativado. Apenas alguns neurônios são ativados, ao mesmo tempo, tornando a rede esparsa, eficiente e fácil para a computação.

Keras - Workflow

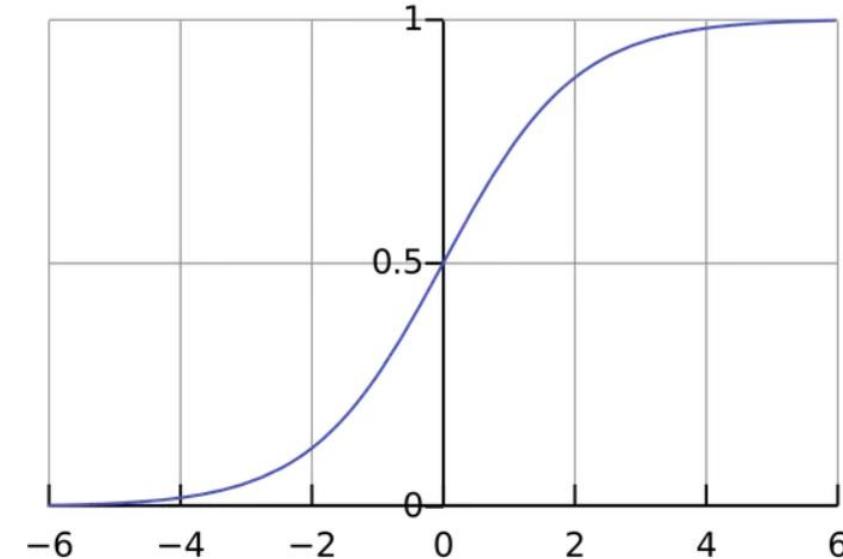
- As funções de ativação mais usadas em redes neurais são:

A função tanh



Funciona de forma semelhante à função sigmóide, mas simétrico em relação à origem. É muito utilizada em problemas de classificação

A função Softmax



É um tipo de função sigmoide. Útil quando lidamos com problemas de classificação multiclasse.

Keras - Workflow

□ Compilação do Modelo

- Qual Loss function?
- Qual otimizador? Adagrad, AdaDelta, Adam, Adamax, Nadam
- Os otimizadores atualizam os parâmetros de peso para minimizar a função de perda.
A função de perda informa ao otimizador se ele está se movendo na direção certa

Problem type	Last-layer activation	Loss function
Binary classification	sigmoid	binary_crossentropy
Multiclass, single-label classification	softmax	categorical_crossentropy
Multiclass, multilabel classification	sigmoid	binary_crossentropy
Regression to arbitrary values	None	mse
Regression to values between 0 and 1	sigmoid	mse or binary_crossentropy

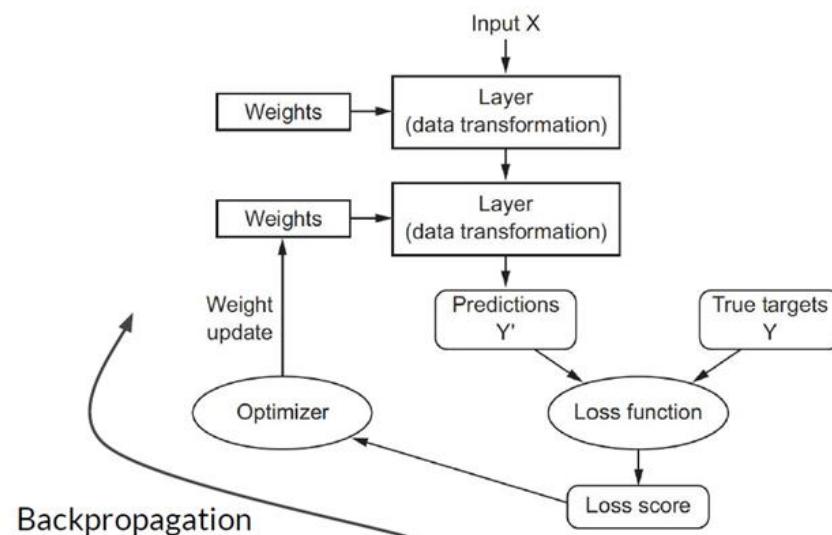
□ Fit

- Ciclo de *back-propagation*. O objetivo do backpropagation é otimizar os pesos para que a rede neural possa aprender a mapear corretamente as entradas para as saídas.

```
In [1]: model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics=['accuracy'])

In [2]: model.fit(predictors, target, validation_split=0.3)

Epoch 1/10
89648/89648 [=====] - 3s - loss: 0.7552 - acc: 0.5775 - val_loss: 0.6969 -
val_acc: 0.5561
Epoch 2/10
89648/89648 [=====] - 4s - loss: 0.6670 - acc: 0.6004 - val_loss: 0.6580 -
val_acc: 0.6102
...
Epoch 8/10
89648/89648 [=====] - 5s - loss: 0.6578 - acc: 0.6125 - val_loss: 0.6594 -
val_acc: 0.6037
Epoch 9/10
89648/89648 [=====] - 5s - loss: 0.6564 - acc: 0.6147 - val_loss: 0.6568 -
val_acc: 0.6110
Epoch 10/10
89648/89648 [=====] - 5s - loss: 0.6555 - acc: 0.6158 - val_loss: 0.6557 -
val_acc: 0.6126
```



Keras - Workflow

□ Utilização do modelo:

- Salvar o modelo
- Utilizar o modelo para realizar previsões

Os modelos devem ser salvos
No formato hdf5 (**Hierarchical Data Format**),
extensão .h5
Padrão usado para armazenamento de
grandes quantidades de dados numéricos

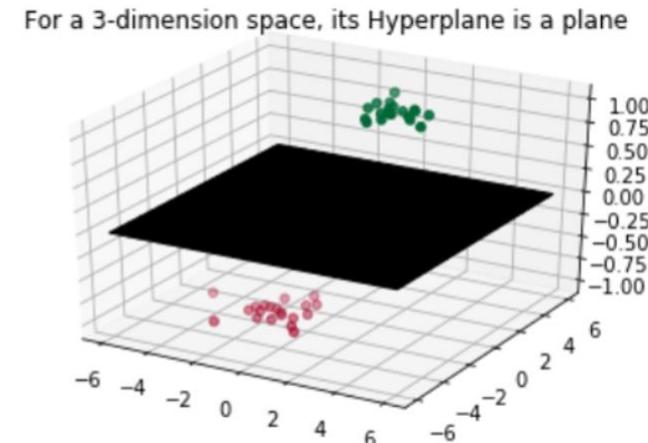
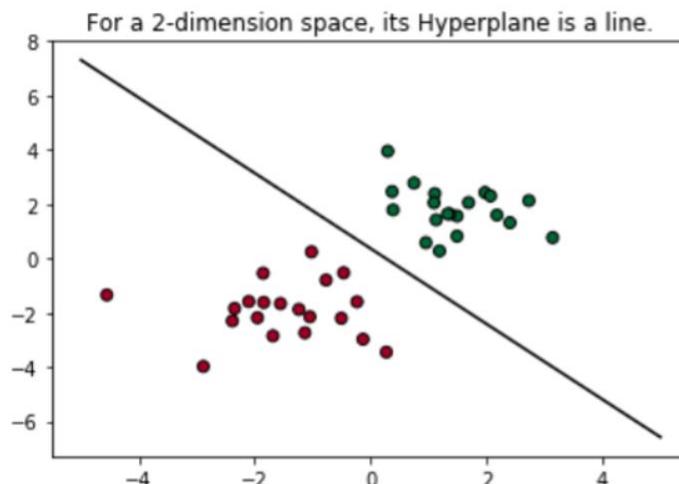
```
In [1]: from keras.models import load_model
In [2]: model.save('model_file.h5')
In [3]: my_model = load_model('my_model.h5')
In [4]: predictions = my_model.predict(data_to_predict_with)
```

Support Vector Machine - SVM

É um algoritmo supervisionado de aprendizagem de máquina para problemas de classificação e regressão.

A ideia de SVM é simples: O algoritmo cria uma linha ou um hiperplano que separa os dados em classes.

A operação do algoritmo SVM é baseada em encontrar o hiperplano que dá a maior distância mínima para os exemplos de treinamento.



Support Vector Machine – SVM - Parâmetros

Kernel: A aprendizagem do hiperplano em SVM é feita através da transformação do problema usando alguma álgebra linear. É aqui que o kernel desempenha o seu papel. Existem várias opções disponíveis com kernels como "linear", "rbf", "poly" e outros.

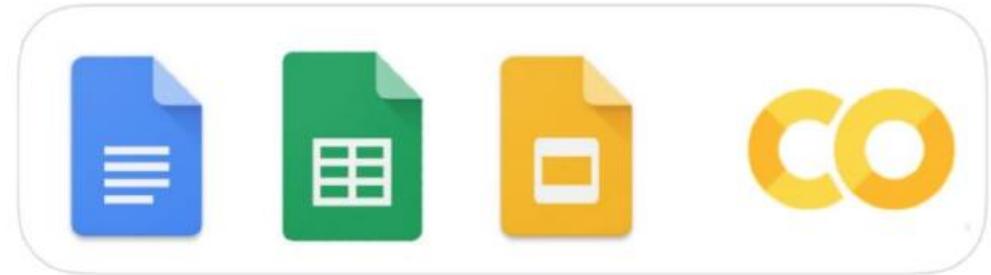
C: Parâmetro de penalidade. Ele também controla o trade off entre o limite de decisão e a classificação correta dos pontos de treinamento. É um valor numérico.

Google Colaboratory



Google Drive

<https://colab.research.google.com/>



Colaboratory é um projeto de pesquisa do Google criado para ajudar a divulgar educação e pesquisa em aprendizagem de máquina. É um ambiente de notebook Jupyter que não requer configuração para ser executado e é executado inteiramente na nuvem.

Os *notebooks* colaborativos são armazenados no Google Drive e podem ser compartilhados da mesma forma que você faria com o Google Docs ou Sheets. O Google Colaboratory é gratuito para uso