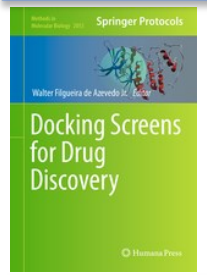
# Classificação

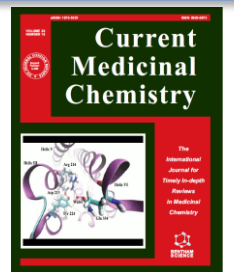
**Prof. Dr. Walter F. de Azevedo, Jr.**

walter@azevedolab.net

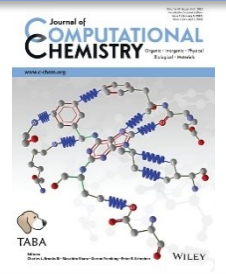
Biography 01♥
Biography 02♥
Biography 03♥
Biography 04♥

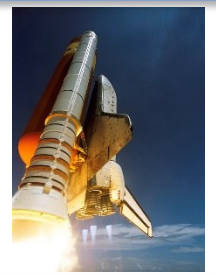
Frontiers Section Editor (Bioinformatics and Biophysics) for the Current Drug Targets ISSN: 1873-5592
Section Editor (Bioinformatics in Drug Design and Discovery) for the Current Medicinal Chemistry ISSN: 1875-533X



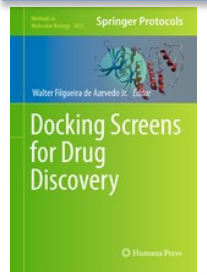


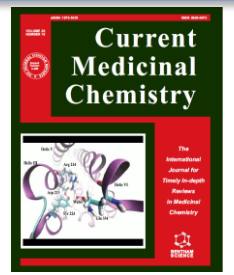


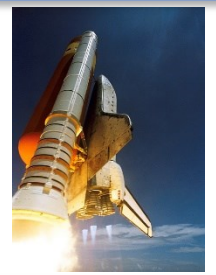




# Conteúdo










## Resumo

Hoje nosso foco está nos métodos de classificação. Veremos a ideia geral desses métodos e discutiremos dois códigos em Python que implementam classificadores (gradiente descendente estocástico e floresta aleatória). Apresentaremos o Jupyter que facilitará a execução dos códigos em Python. Discutiremos em detalhes as principais métricas para a avaliação do poder de previsão de classificadores. Usaremos como caso de estudo um conjunto de dados com números escritos à mão. Os nossos classificadores funcionam como detectores do número 5 entre os dígitos disponíveis no conjunto de dados MNIST (*Modified National Institute of Standards and Technology*). Discutiremos o poder de previsão dos métodos ensemble.

Palavras-chave: aprendizado de máquina, *machine learning*, modelo de aprendizado de máquina, classificação, classificadores, Python, Scikit-Learn, NumPy, Matplotlib, Pandas, Jupyter, MNIST, matriz de confusão, precisão, revocação, curva ROC, gradiente descendente estocástico, floresta aleatória, *random forest,* validação cruzada, *k-fold*, *cross-validation*, métodos ensemble.

Fonte: https://pixabay.com/photos/forest-night-landscape-fantasy-5167332/

## Dados Reais para Estudos de Aprendizado de Máquina

Boa parte dos conceitos discutidos aqui estão descritos em no capítulo 3 do livro Géron, Aurélien. Mãos A Obra: Aprendizado De Máquina Com Scikit-Learn, Keras & TensorFlow: Conceitos, Ferramentas e Técnicas Para a Construção de Sistemas Inteligentes (Portuguese Edition). Alta Books. Edição do Kindle.


GÉRON, Aurélien. Mãos A Obra: **Aprendizado De Máquina Com Scikit-Learn, Keras & TensorFlow: Conceitos, Ferramentas e Técnicas Para a Construção de Sistemas Inteligentes** (Portuguese Edition). Alta Books. Edição do Kindle.

## Dados Reais para Estudos de Aprendizado de Máquina

Uma forma de se familiarizar com todas etapas de um projeto de aprendizado de máquina é a partir da elaboração de modelos focados em dados reais. Cientes dessa necessidade, desenvolvedores de ferramentas de aprendizado de máquina criaram bases de dados de acesso aberto onde conjuntos de dados estão disponíveis para a criação e teste de modelos. Especificamente para a criação de modelos para o estudo de sistemas proteicos temos o protein data bank (PDB) e BindingDB. O PDB traz informação sobre a estrutura tridimensional de moléculas biológicas, incluindo as proteínas. O BindingDB complementa o PDB com dados sobre a afinidade de moléculas com potencial farmacológico devido sua interação com proteínas.

## Dados Reais para Estudos de Aprendizado de Máquina

Além das duas bases de dados citadas, há diversas outras de interesse geral, como o UC Irvine Machine Learning Repository e Kaggle. Há a possibilidade de acessar dados reais a partir da biblioteca Scikit-Learn.


**Repositórios populares de open data**

UC Irvine Machine Learning Repository (http://archive.ics.uci.edu/ml/)

Conjunto de dados no Kaggle (https://www.kaggle.com/datasets)

Conjunto de Dados no AWS da Amazon (https://registry.opendata.aws/)

**Metaportais de dados (listam os repositórios open data)**

Data Portals (http://dataportals.org/)

OpenDataMonitor (http://opendatamonitor.eu/)

Quandl (http://quandl.com/)

**Outras páginas que listam muitos repositórios populares de open data**

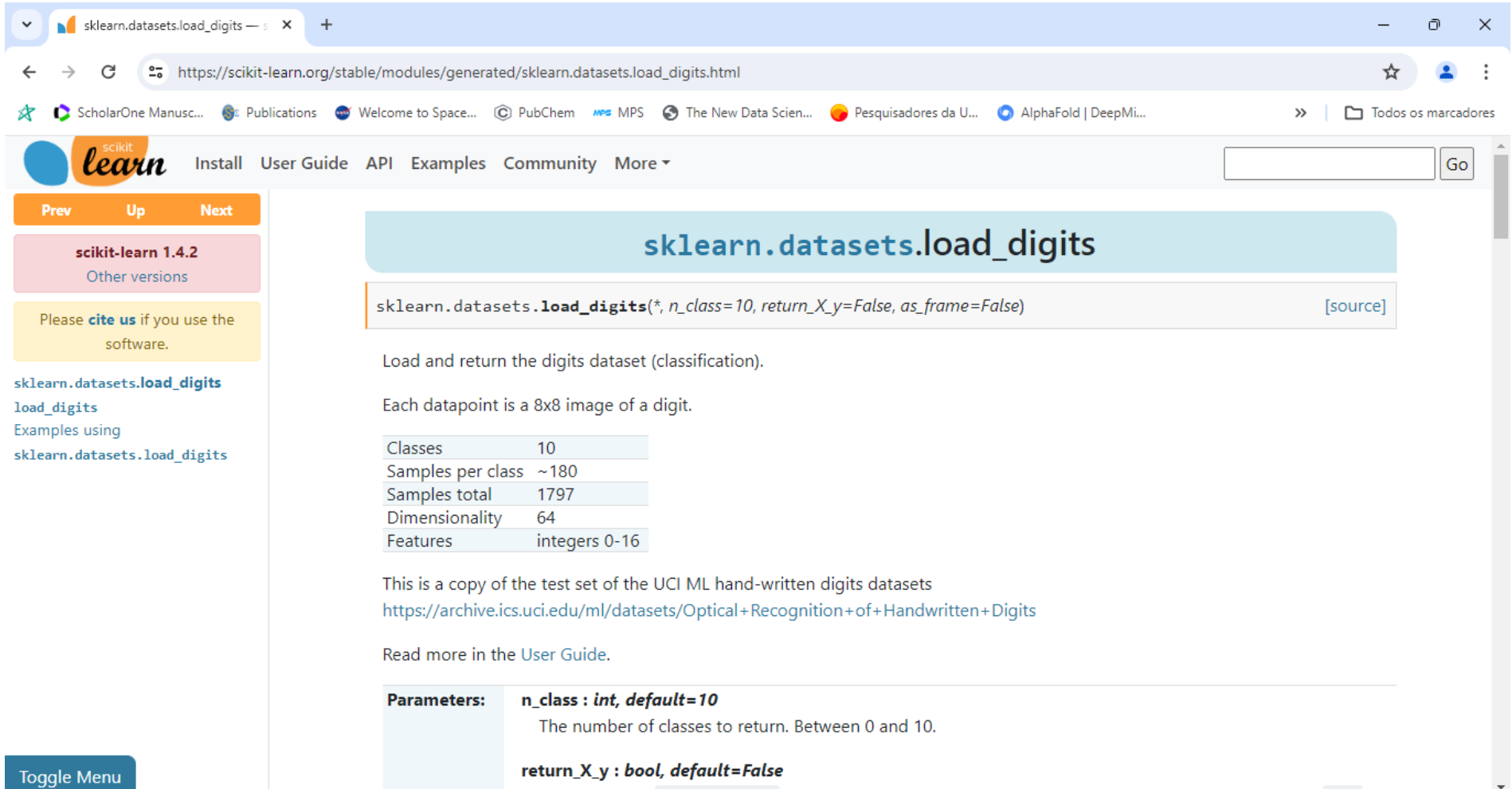
Lista de conjuntos de dados de aprendizado de máquina do Wikipedia (https://homl.info/9)

Quora.com (https://homl.info/10)

Conjuntos de dados em subseção do Reddit (https://www.reddit.com/r/datasets)

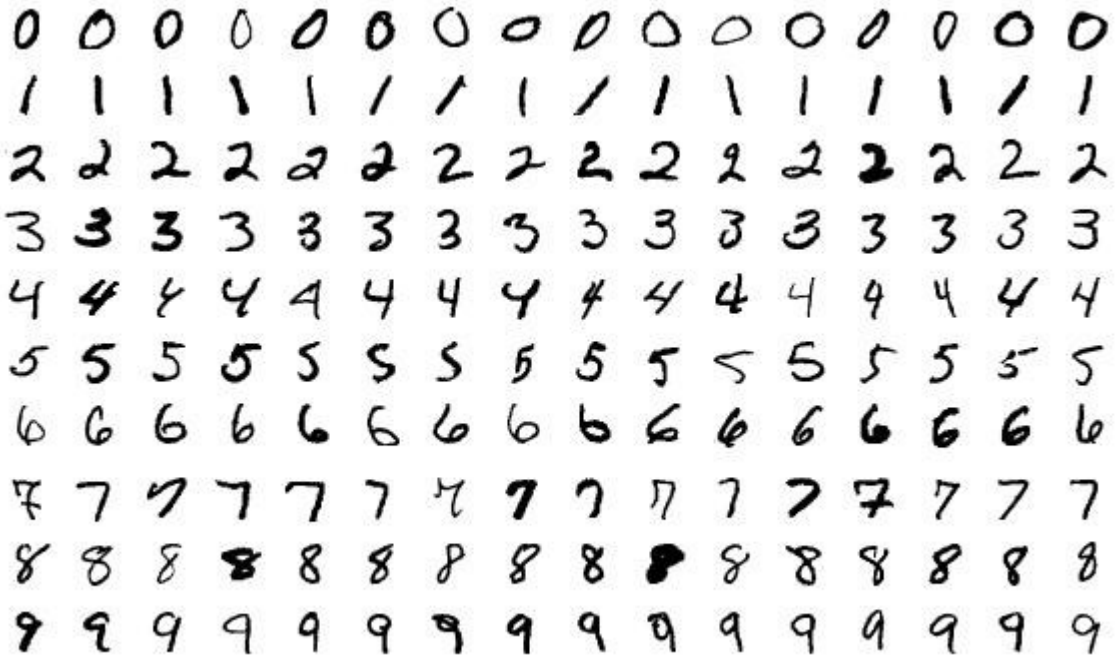
## Dados Reais para Estudos de Aprendizado de Máquina

O Sckit-Learn tem alguns conjuntos de dados definidos para uso no desenvolvimento de modelos de aprendizado de máquina. Na aula de hoje, usaremos um conjunto de dados com a escrita manual de números. O objetivo é treinar um modelo de aprendizado de máquina supervisionado para classificação das imagens dos números escritos à mão.



## Dados Reais para Estudos de Aprendizado de Máquina

O conjunto de imagens é chamado de conjunto de dados MNIST (*Modified National Institute of Standards and Technology*). Podemos usar o conjunto de dados MNIST disponível no Scikit-Learn. Essa abordagem permite a integração de todas as etapas da elaboração do modelo a partir do uso da biblioteca Sckit-Learn. O conjunto de dados MNIST tem 70 mil imagens com os números escritos à mão e os respectivos rótulos identificando cada número de 0 a 9.

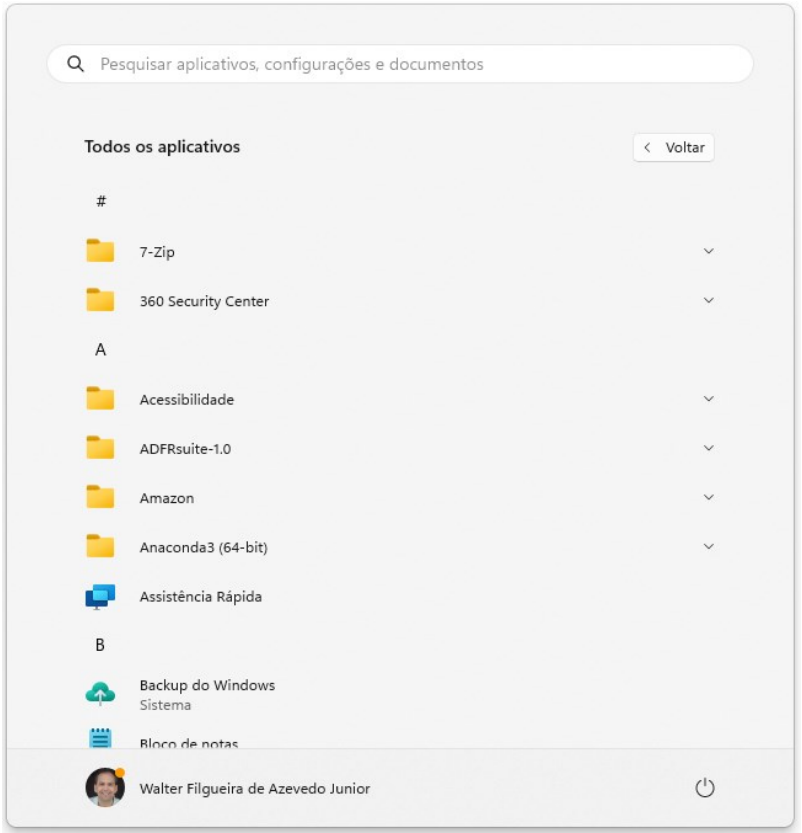


## Instalação do Python

O objetivo da presente disciplina não é ensinar a linguagem de programação Python, mas iremos usar códigos em Python para criar os modelos de aprendizado de máquina. É preciso instalar o Python 3 no seu computador para podermos explorar as bibliotecas que trazem os recursos do aprendizado de máquina. A forma mais direta de instalar o Python e por meio do Anaconda. Por meio do Anaconda, temos a instalação das bibliotecas necessárias para os códigos discutidos aqui. Para facilitar o uso dos códigos em Python, usaremos o Jupyter. O Jupyter é um ambiente de desenvolvimento de códigos interativo e que permite a execução do seu código diretamente no navegador. Depois de instalado o Anaconda, instale o Jupyter.

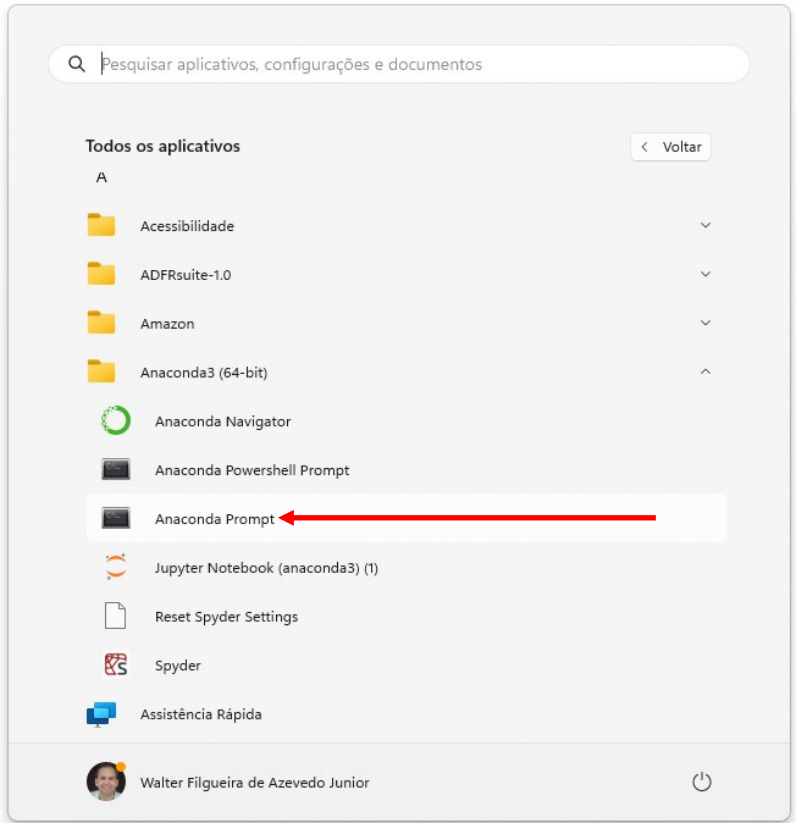

## Ambiente Jupyter

Depois de instalado o Anaconda, temos as principais bibliotecas usadas em aprendizado de máquina. Para instalar a última versão do Jupyter é necessário termos o pip. As instruções a seguir para o sistema operacional Windows. Informações sobre a instalação do Jupyter em Linux podem ser encontradas no livro Géron 2023. No Windows, abra um prompt de comandos (Anaconda Prompt). Abra o prompt a partir do Menu Iniciar. Depois clique na opção Todos os aplicativos. Você terá a tela de opções abaixo.



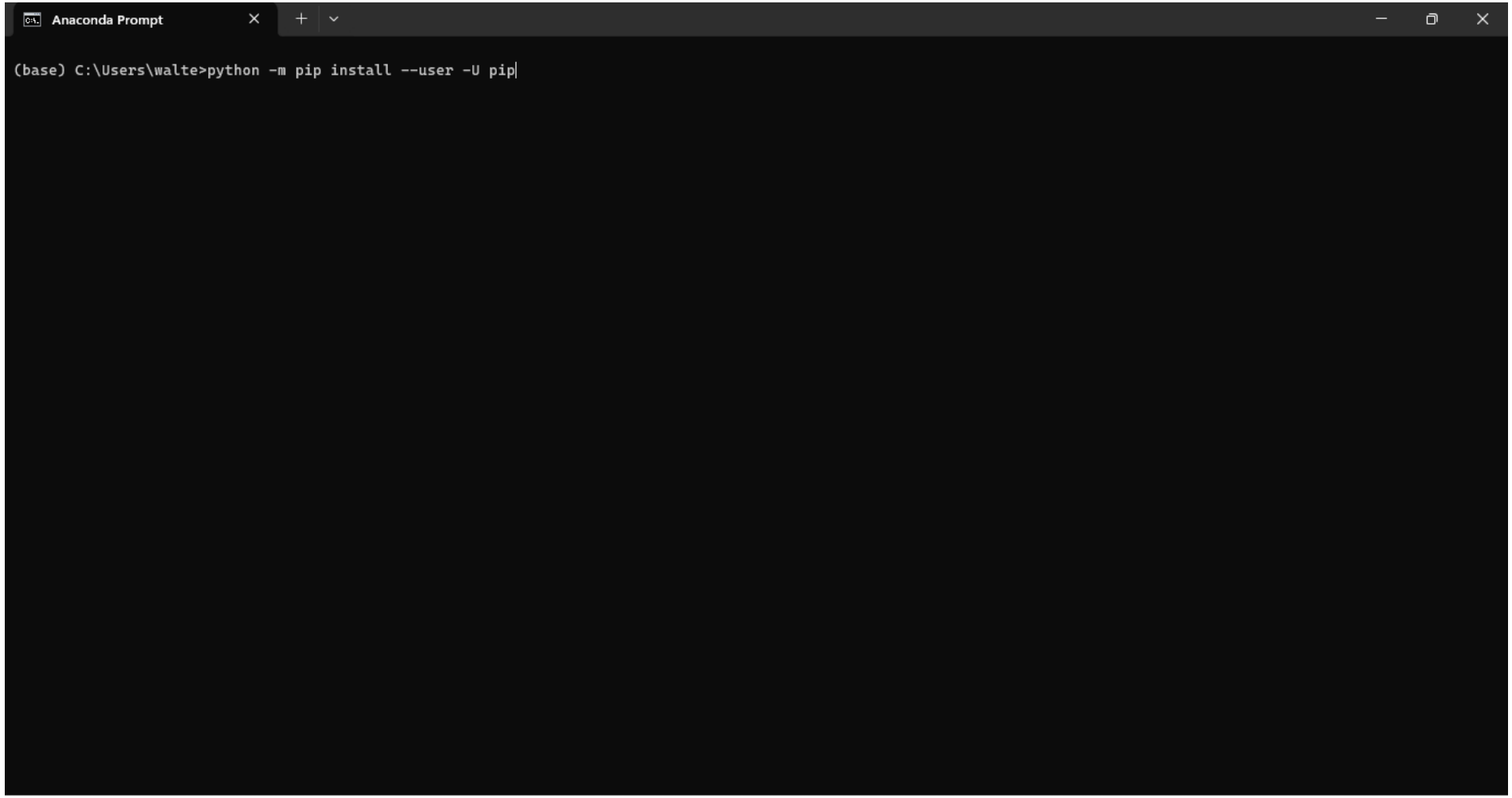
## Ambiente Jupyter

Expanda a pasta do Anaconda e escolha a opção Anaconda Prompt.



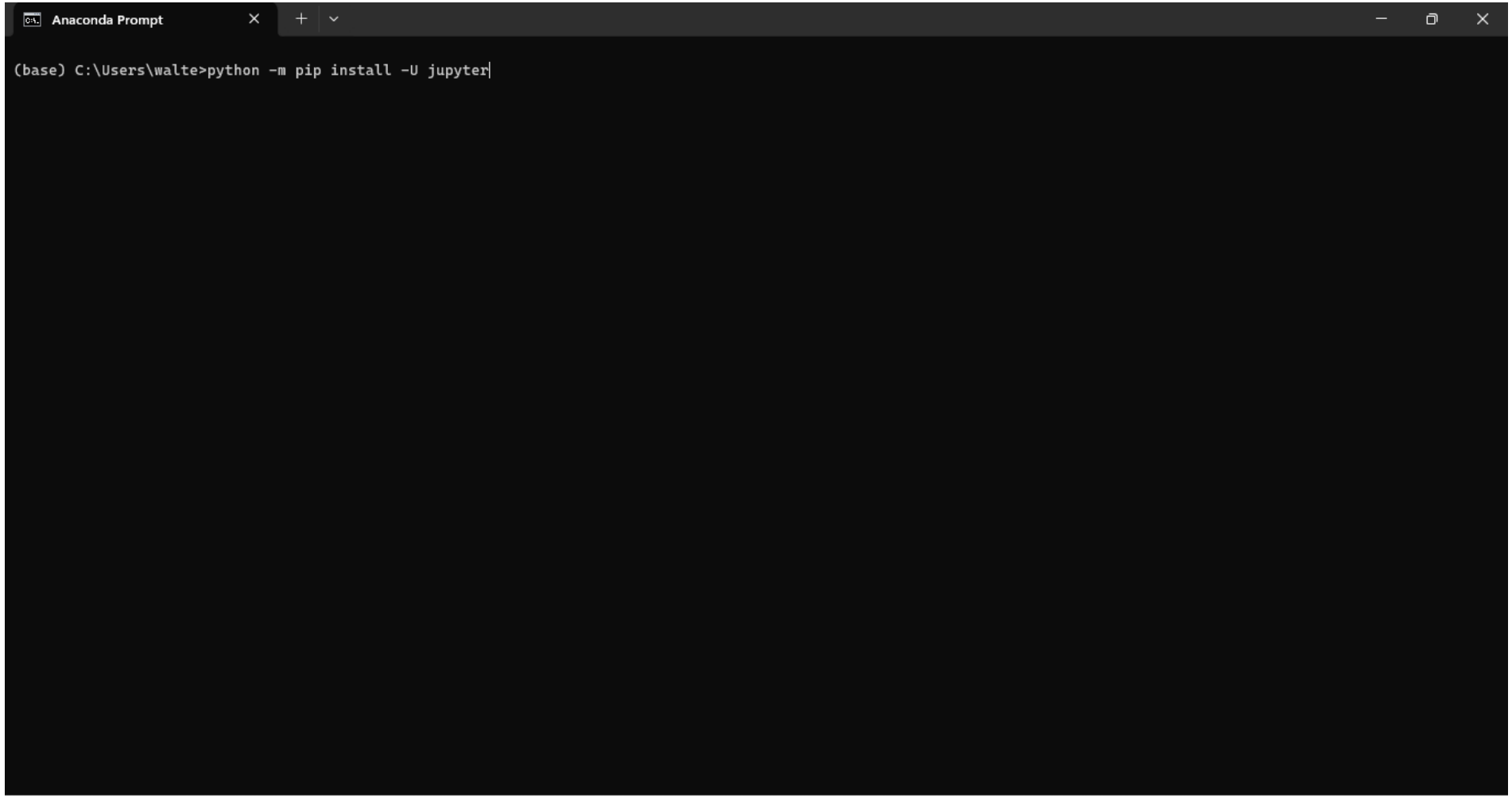
## Ambiente Jupyter

No Anaconda Prompt digite o seguinte comando: *python -m pip install --user -U pip*



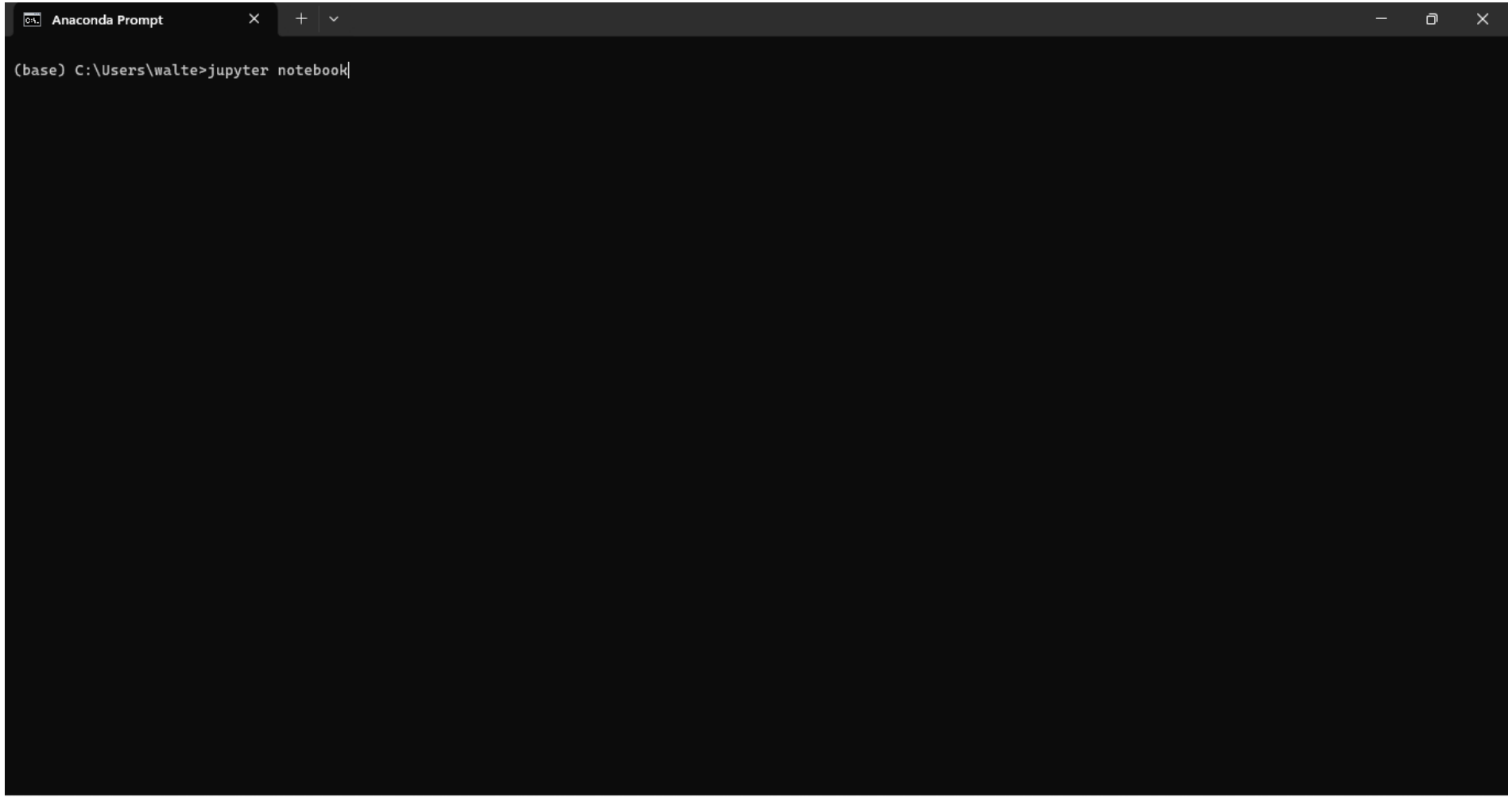
## Ambiente Jupyter

Depois de finalizar a instalação do pip, digite: *python -m pip install -U jupyter*



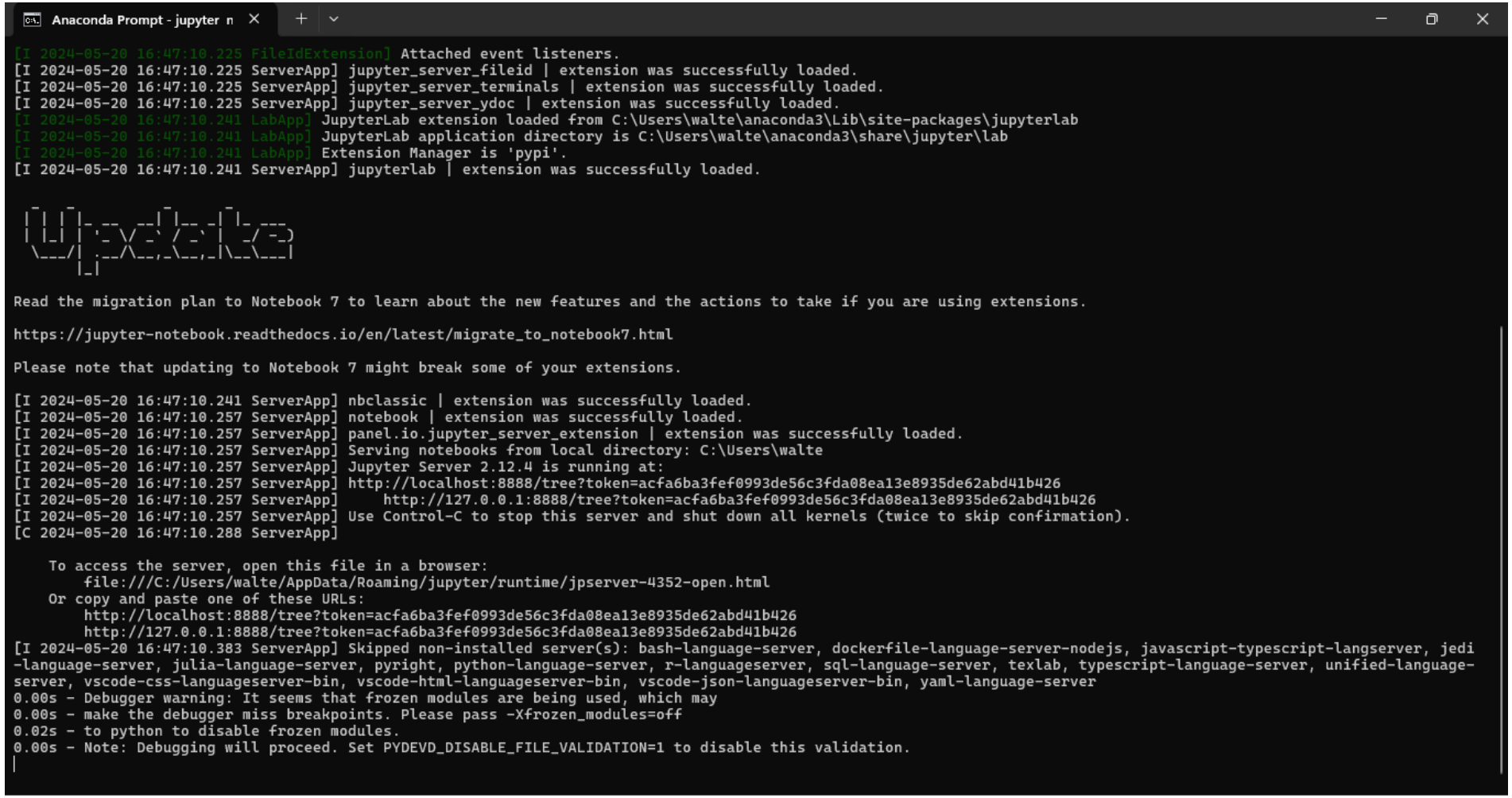
## Ambiente Jupyter

Finalizada a instalação do Jupyter, digite: *jupyter notebook*



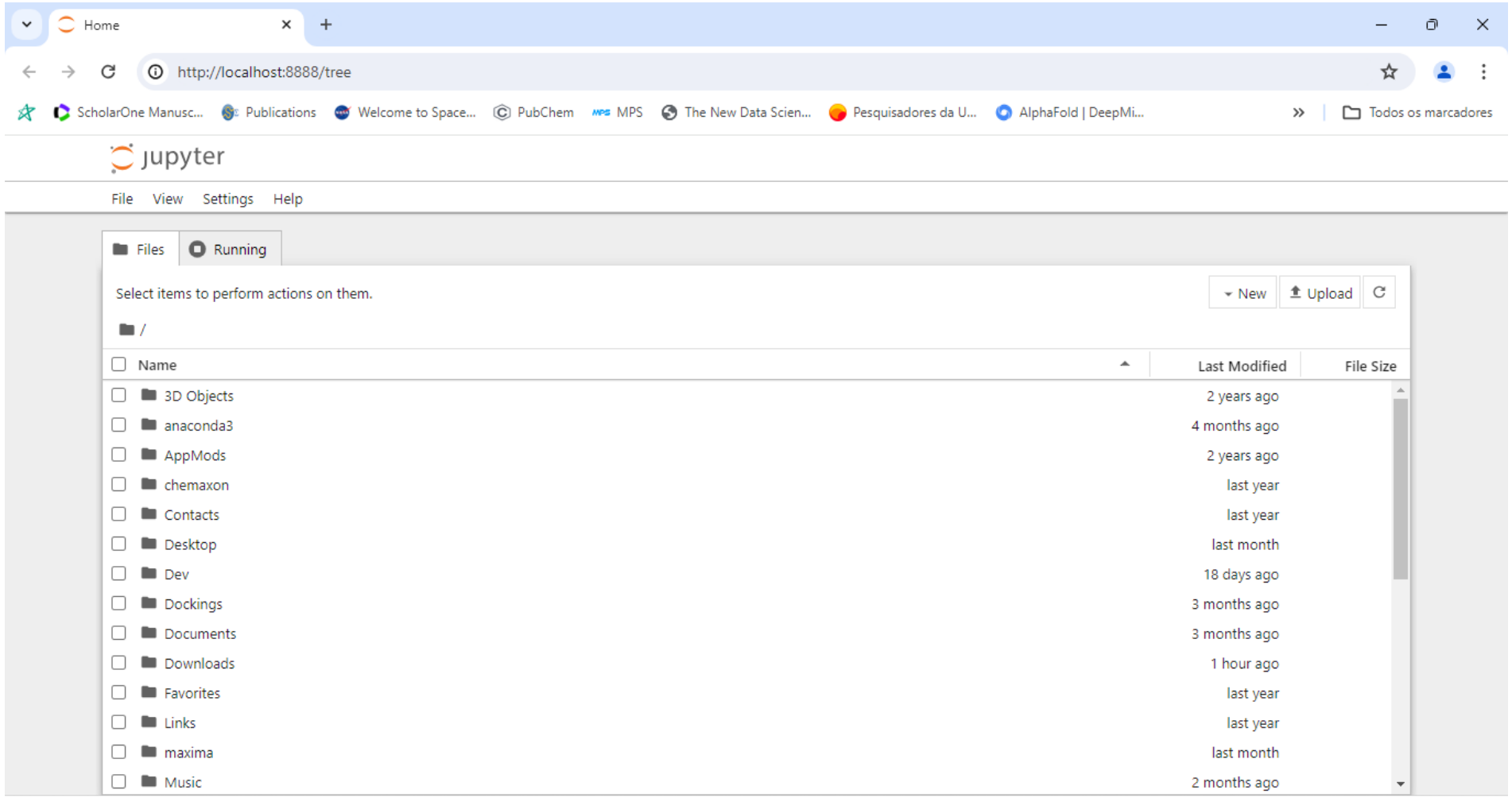
## Ambiente Jupyter

No Anaconda Prompt, você terá as informações sobre a execução do Jupyter e na sequência a abertura do seu navegador.



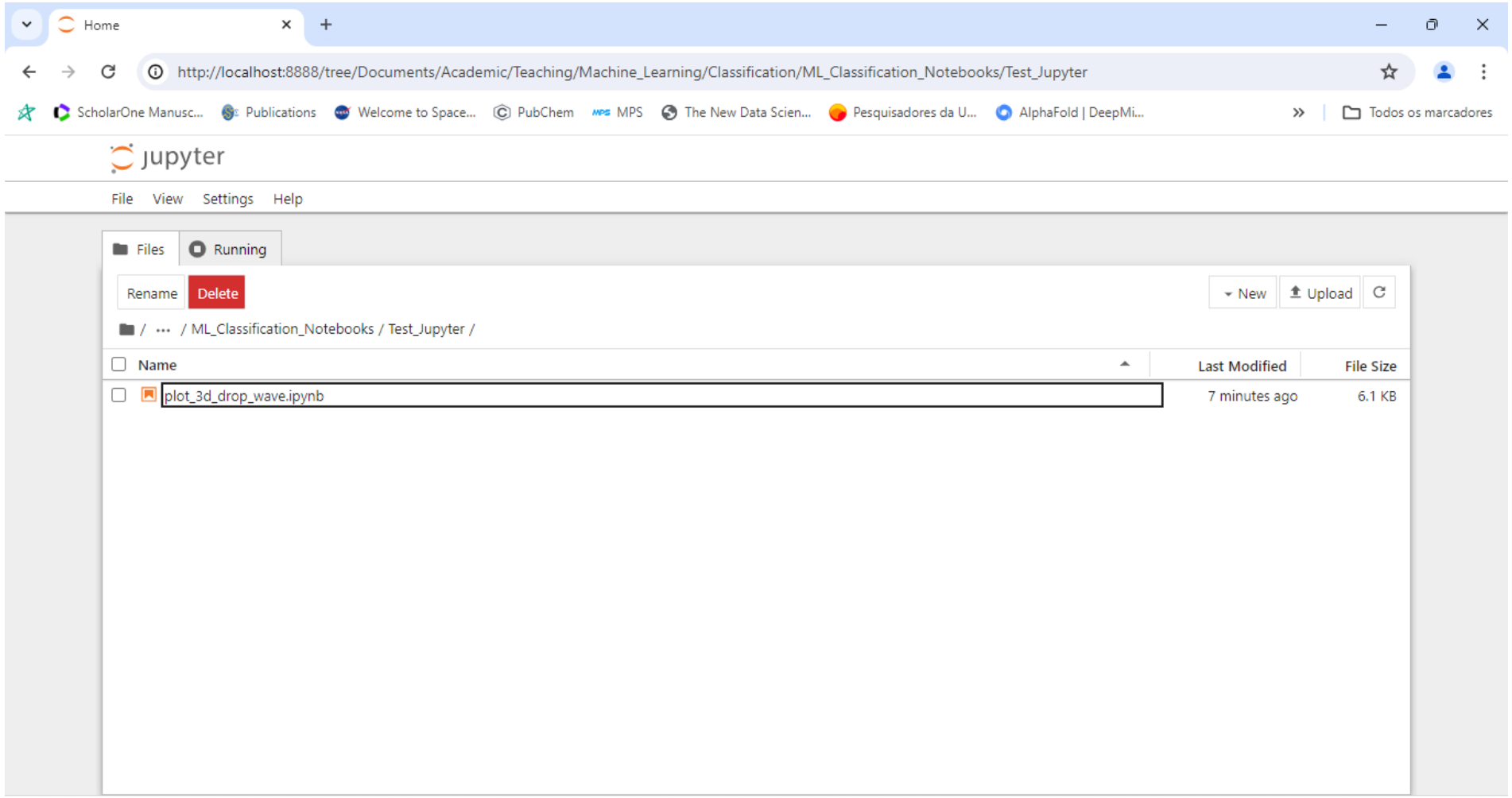
## Ambiente Jupyter

Abaixo temos a tela do Jupyter. Navegaremos para a pasta dos notebooks (formato Jupyter) da aula de hoje (*ML_Classification_Notebooks*). Para o teste usaremos a pasta *ML_Classification_Notebooks/Test_Jupyter.*



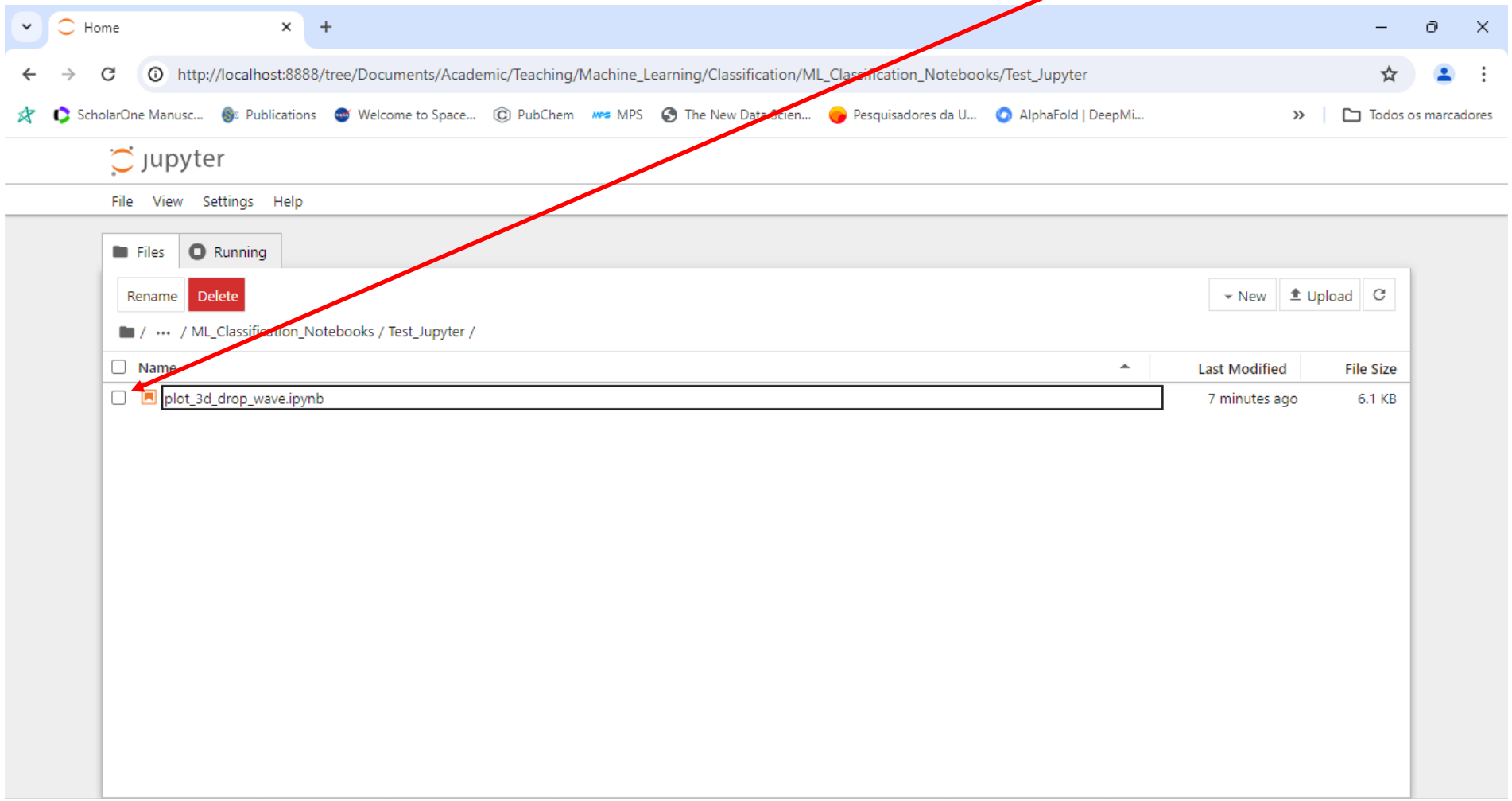
## Ambiente Jupyter

Na pasta *ML_Classification_Notebooks/Test_Jupyter* temos o notebook *plot_3d_drop_wave.ipynb*. Os notebooks do Jupyter tem extensão *.ipynb*. Os códigos em Python tem extensão *.py*.



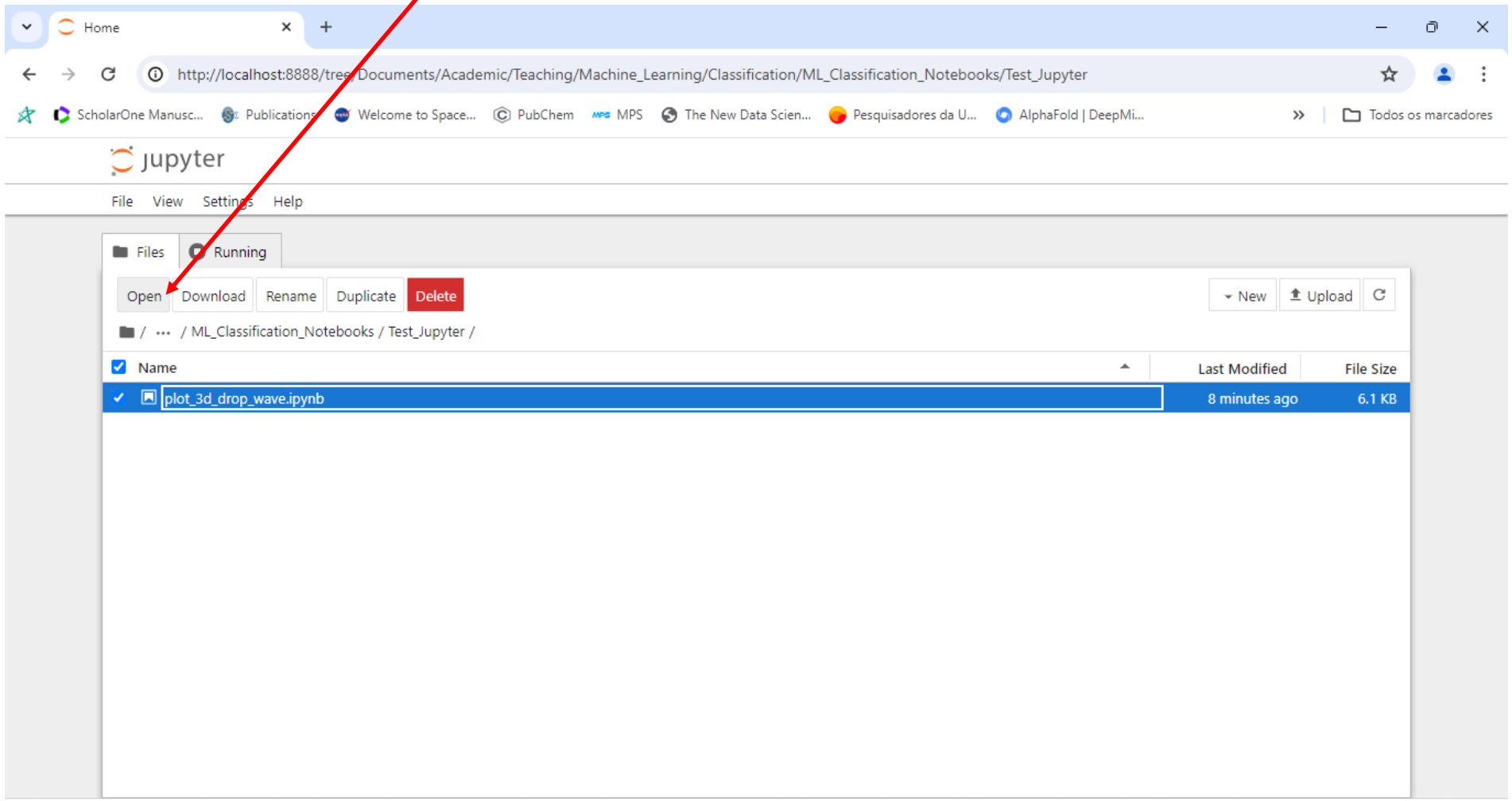
## Ambiente Jupyter

Para carregar o notebook *plot_3d_drop_wave.ipynb*. Clique no campo à esquerda do nome.



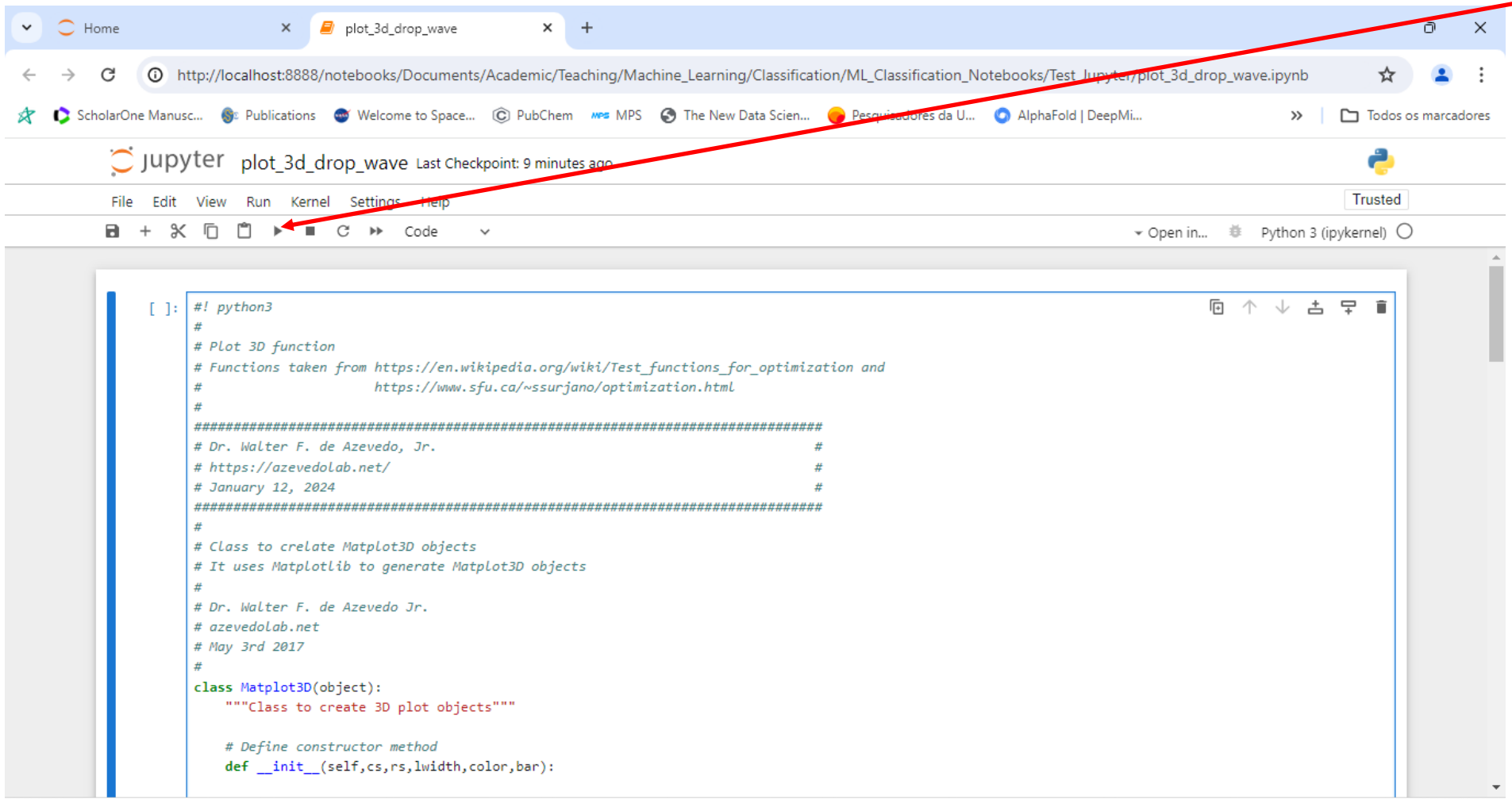
## Ambiente Jupyter

Em seguida, clique em Open.



## Ambiente Jupyter

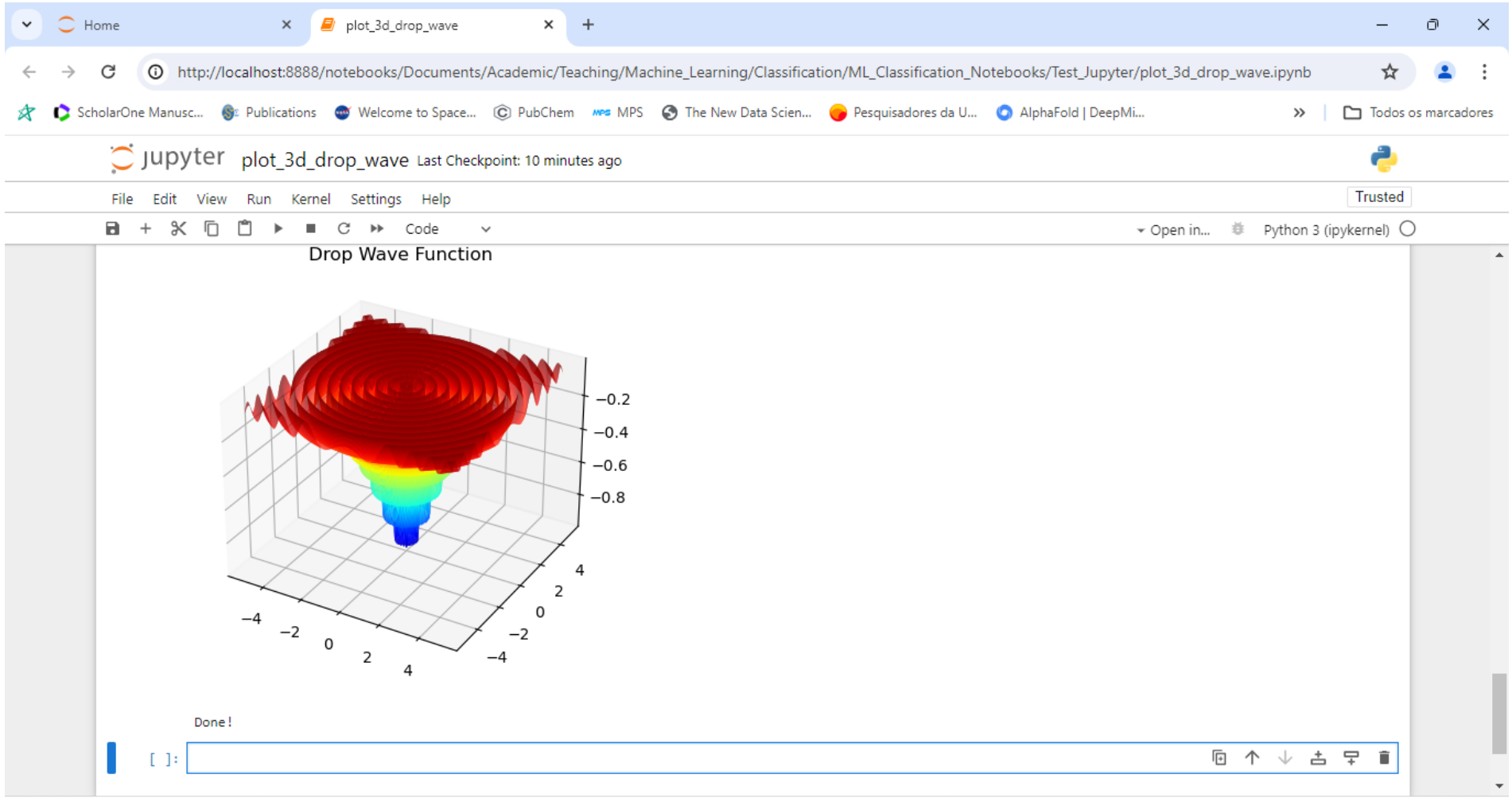
O Jupyter abriu o notebook. Como destacado, nosso objetivo não é ensinar Python nesta disciplina. Iremos somente mostrar como rodar o código. Clique em qualquer parte do código e depois no botão indicado.




Alternativamente, você pode pressionar as teclas *Shift+Enter* para rodar seu código.

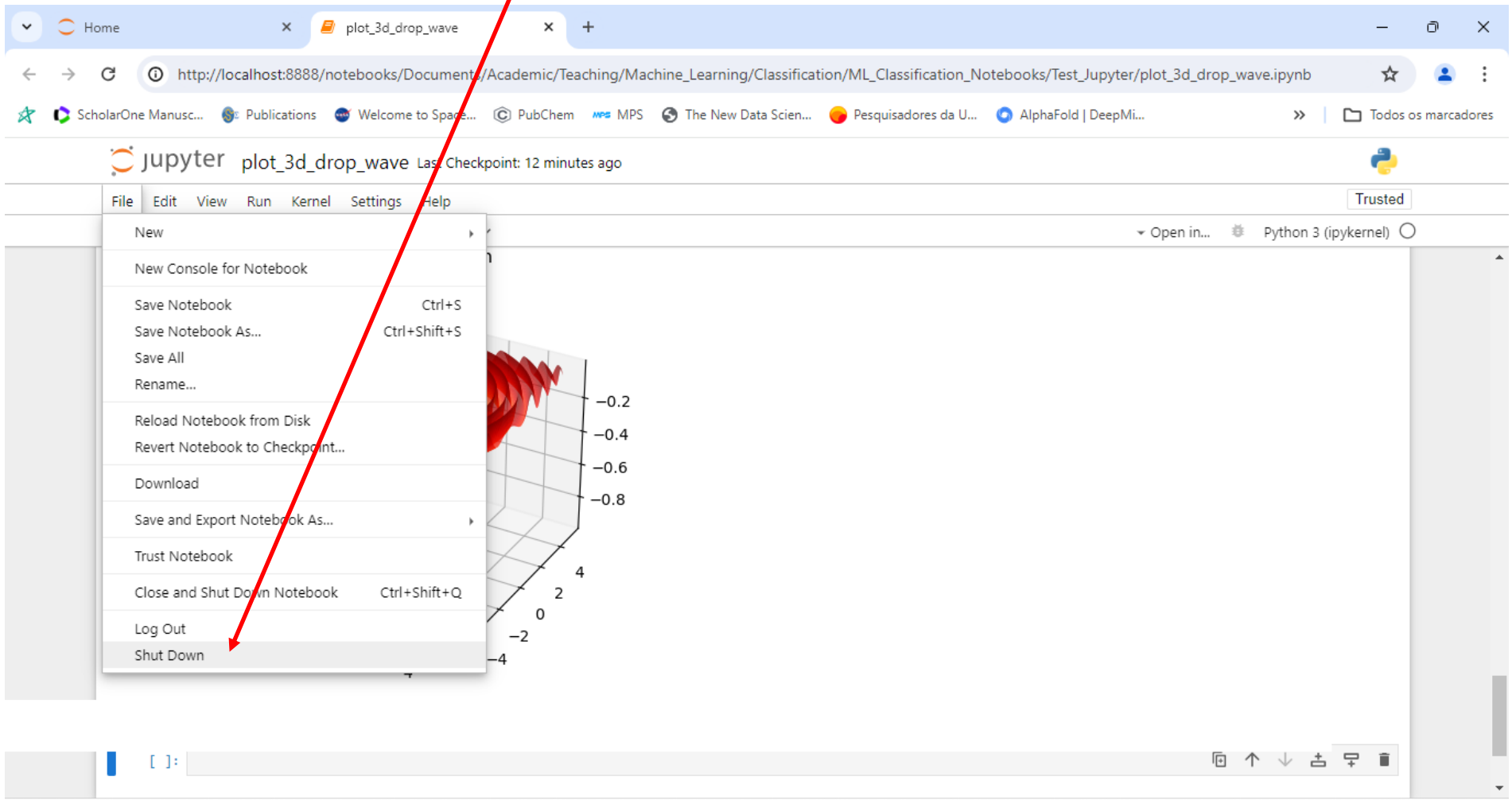
## Ambiente Jupyter

Depois de alguns segundos, temos o gráfico 3D gerado, como mostrado abaixo. Mais adiante retornaremos ao Jupyter para rodar os códigos da aula. Essa amostra foi só para mostrar como rodar os códigos no Jupyter.



## Ambiente Jupyter

Para encerrar, clique *File->Shut Down*. Confirme clicando no Botão *Shut Down*.




**Dica:** Repita os passos de abertura e execução do notebook *plot_3d_drop_wave.ipynb* para se familiarizar com o processo.

## Bibliotecas do Python

Como previamente destacado, com a instalação do Anaconda teremos diversas bibliotecas úteis para o desenvolvimento de modelos de aprendizado de máquina. Abaixo destacamos as principais usadas nesta aula com uma breve descrição de cada uma. Maiores informações sobre o funcionamento de cada biblioteca estão disponíveis clicando no logo de cada uma.

NumPy é uma biblioteca de recursos para computação numérica. Originalmente desenvolvida usando outras linguagens de programação diferentes do Python, a biblioteca NumPy possibilita a rápida integração de um amplo leque de rotinas numéricas aos programas escritos em Python.


Scikit-learn (Pedregosa et al., 2011) é uma das mais citadas bibliotecas dedicadas ao desenvolvimento de ferramentas de aprendizado de máquina para uso na linguagem de programação Python. Temos na biblioteca Scikit-learn métodos para regressão, classificação e *clustering*. Além de vasta gama de algoritmos para geração de modelos, a Scikit-learn tem disponíveis ferramentas adicionais que disponibilizam conjuntos de dados, algoritmos de escalonamento e uma coleção de métricas para avaliar o poder de previsão de modelos.


Matplotlib é uma biblioteca gráfica que permite a criação de gráficos para a análise de dados. A riqueza de recursos permite a geração de gráficos bidimensionais e tridimensionais normalmente usados na análise estatística como gráficos de dispersão e histograma.


Pandas é uma biblioteca da linguagem de programação Python para a análise e manipulação de dados. Há recursos para leitura e escrita de arquivos, o que facilita a entrada e saída de dados dos programas desenvolvidos para implementação de métodos de aprendizado de máquina.

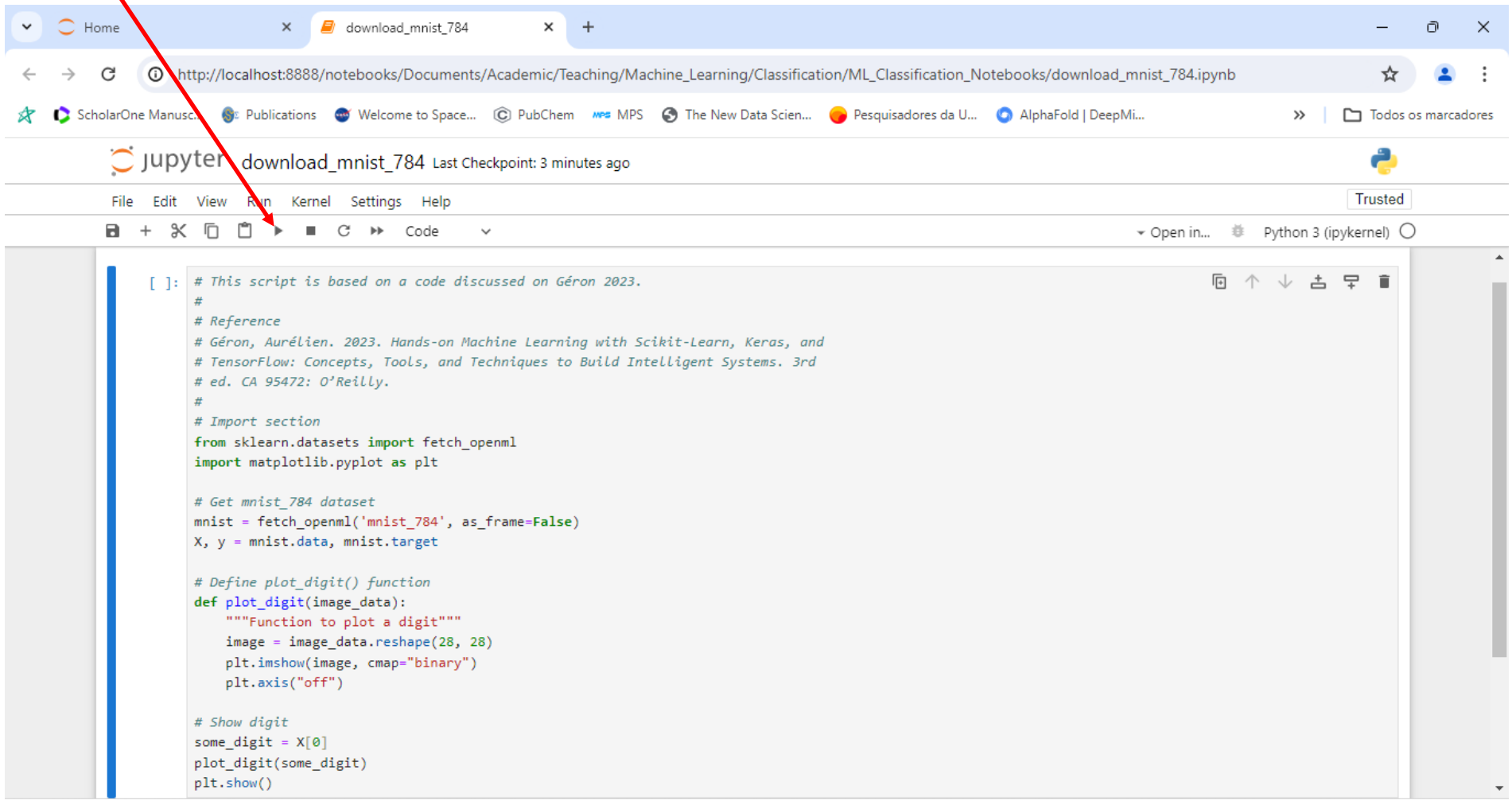
## Download da Base da Dados MNIST

A biblioteca Scikit-Learn tem diversos recursos para fazer o download de conjuntos de dados para treinamento e teste de modelos de aprendizado de máquina. O código *download_mnist_784.ipynb* faz o download do conjunto de dados MNIST e mostra um dos dígitos na tela. Abra o Jupyter e vá para pasta de código e abra o código *download_mnist_784.ipynb*.

Fonte: https://pixabay.com/illustrations/machine-learning-intelligence-8530772/

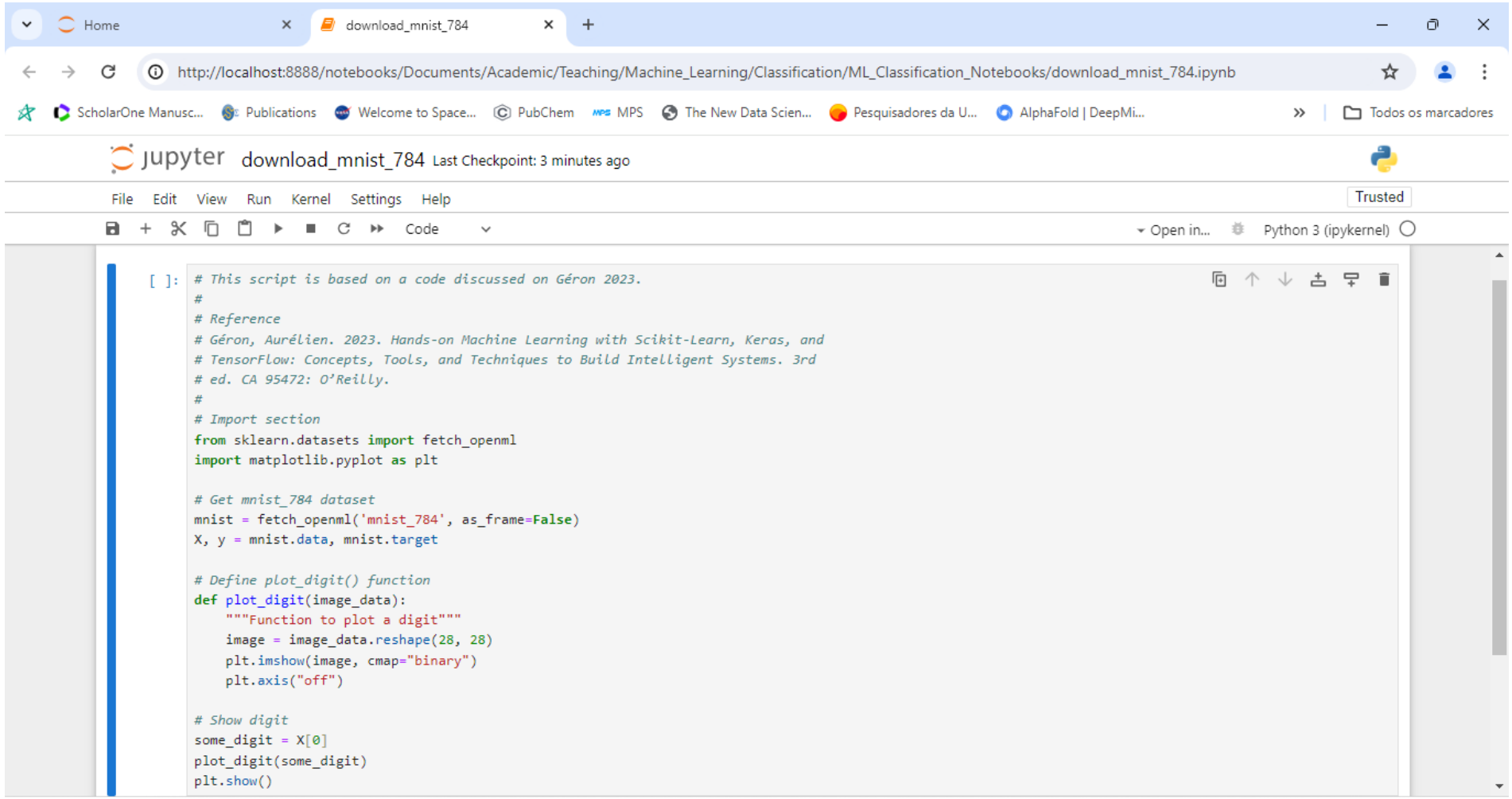
## Download da Base da Dados MNIST

Abaixo temos a janela do Jupyter com o código aberto. Clique em qualquer parte do código. Em seguida clique no botão para executar o código.



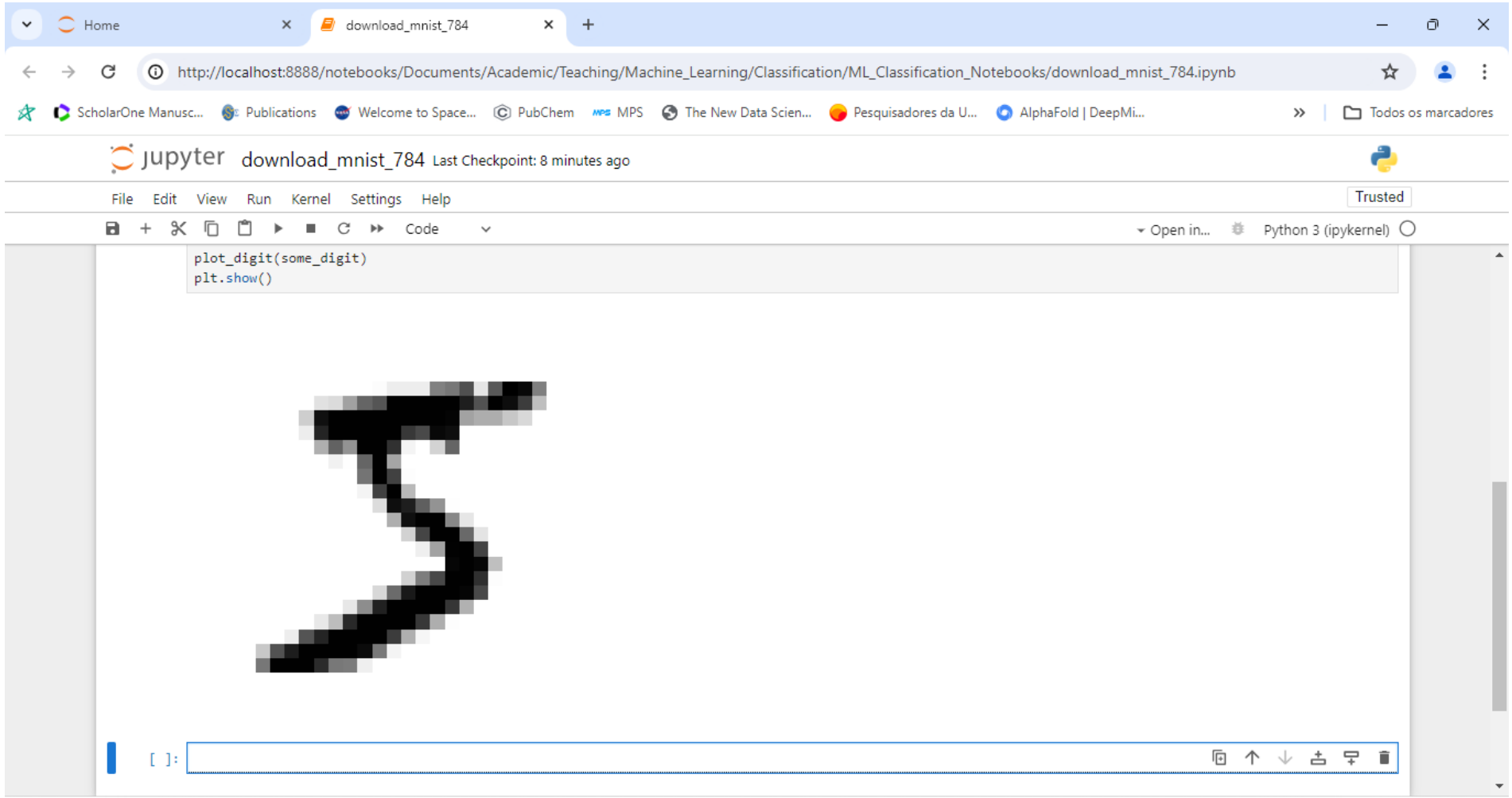
## Download da Base da Dados MNIST

A MNIST dispõe 70 mil imagens onde cada uma tem 784 características (*features*). O resultado de 784 *features* é devido ao número de pixels. Temos 28×28 pixels, e cada um é binário indicando 0 (branco) ou 255 (preto).



## Download da Base da Dados MNIST

Depois de alguns segundos, a figura do dígito é mostrada no Jupyter.

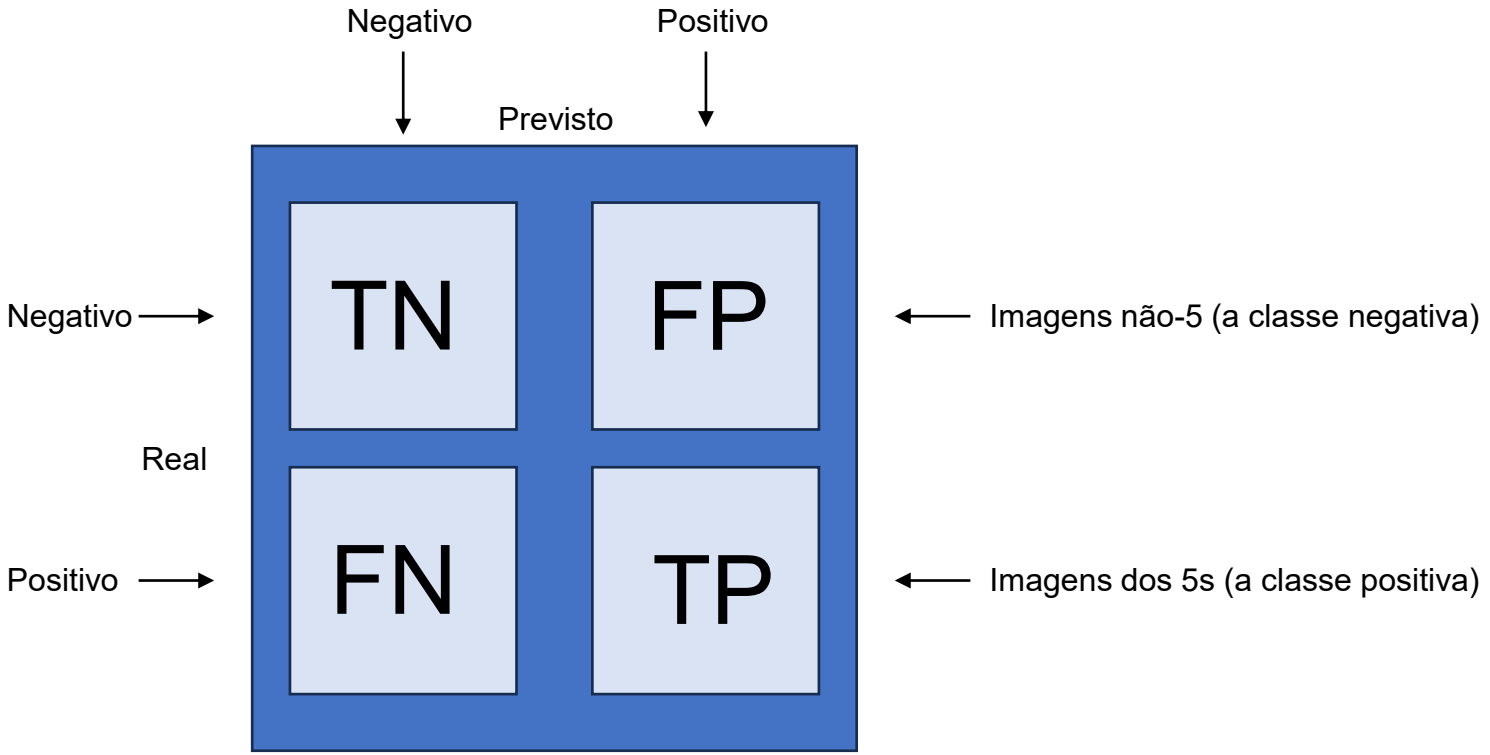


## Download da Base da Dados MNIST

Como destacado, temos 70 mil imagens e os dados já estão divididos em conjunto de treinamento (as primeiras 60 mil imagens) e conjunto de teste (as últimas 10 mil imagens). Os dados dos conjuntos de treinamento já estão embaralhados de forma a que podemos dividi-los em subconjuntos para a validação cruzada (*cross-validation*).

## Matriz de Confusão

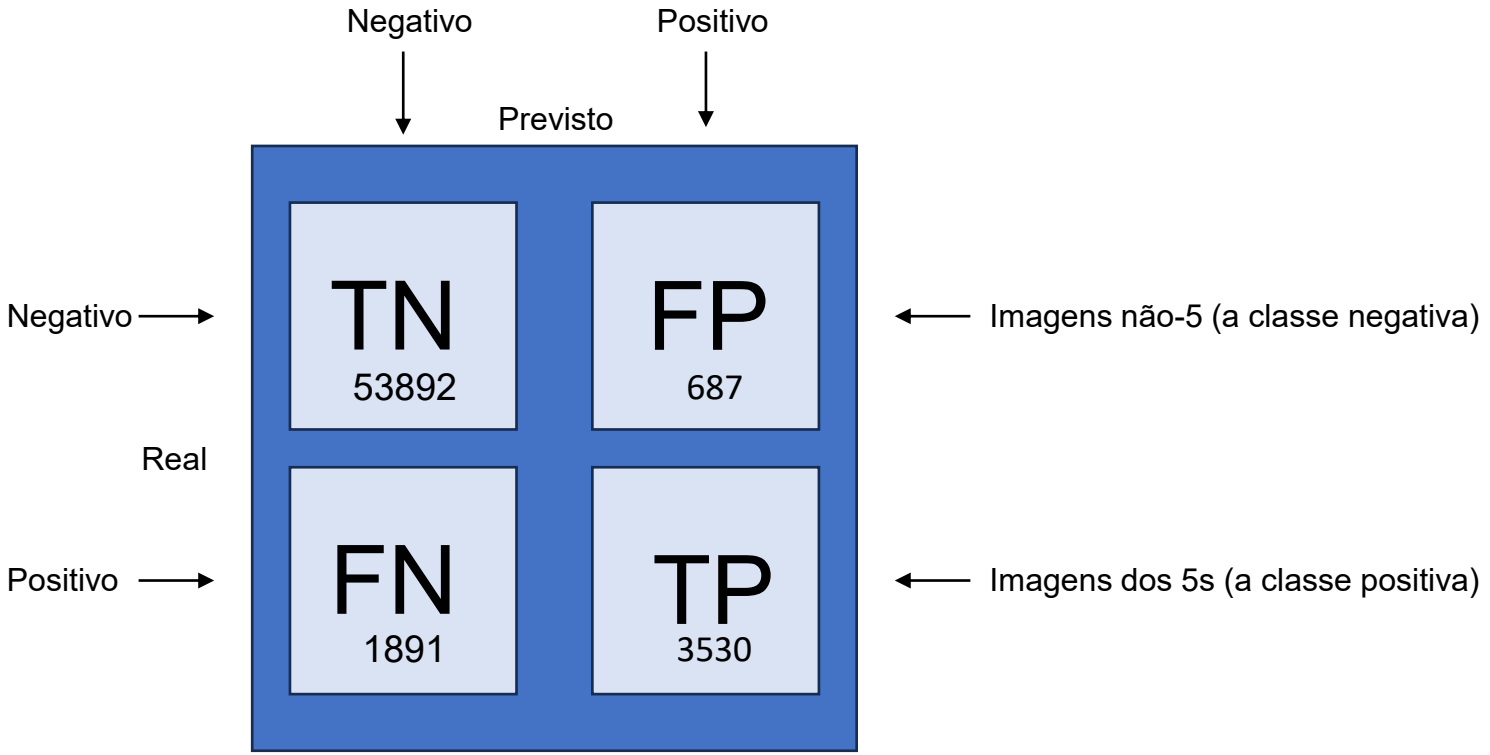
A matriz de confusão é usada para avaliar um algoritmo classificador. Abaixo temos um diagrama esquemático da matriz de confusão com a explicação sumária de cada campo. No diagrama temos quatro campos que indicam o número de previsões certas e erradas. Mais adiante veremos o código em Python de um programa que detecta 5 no conjunto de dados MNIST. Para a montagem da matriz de confusão, usamos nas colunas os valores previstos e nas linhas os reais. Tanto nas colunas quanto nas linhas usamos a sequência negativo e positivo. O verdadeiro indica previsão certa e falso previsão errada, no total temos as quatro situações indicadas.




TN: *True negative* (verdadeiro negativo)
FP: *False positive* (falso positivo)
FN: *False negative* (falso negativo)
TP: *True positive* (verdadeiro positivo)

## Matriz de Confusão

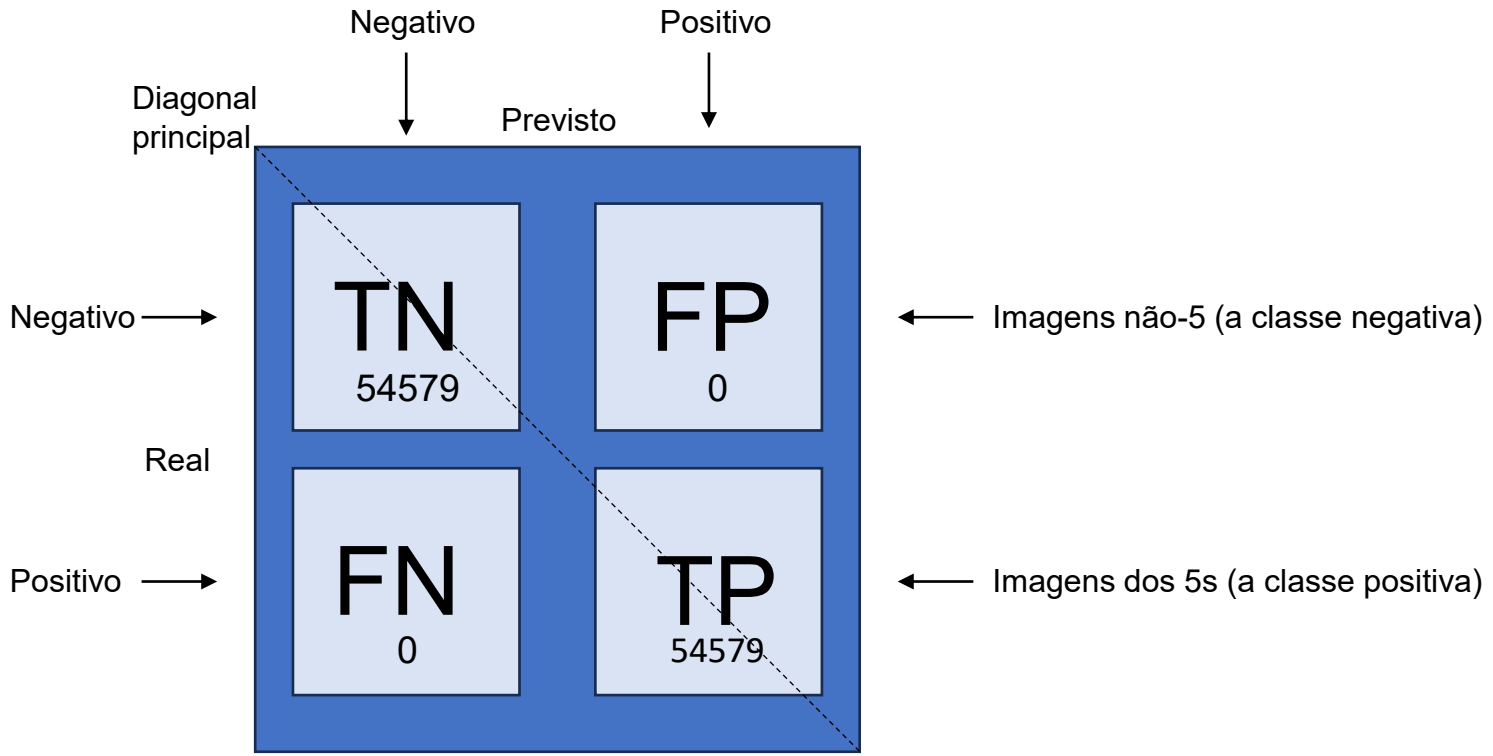
Para fixar o conceito, vamos destacar um exemplo numérico. A primeira linha da matriz de confusão avalia imagens não-5 (a classe negativa): 53892 delas foram classificadas corretamente como não-5 (**verdadeiros negativos**) (TN), enquanto as 687 restantes foram erroneamente classificadas como 5s (**falsos positivos**) (FP). A segunda linha considera as imagens dos 5s (a classe positiva): 1891 foram classificadas erroneamente como não-5s (**falsos negativos**) (FN), ao passo que as 3530 restantes foram classificadas perfeitamente como 5s (**verdadeiros positivos**) (TP).




TN: *True negative* (verdadeiro negativo)
FP: *False positive* (falso positivo)
FN: *False negative* (falso negativo)
TP: *True positive* (verdadeiro positivo)

## Matriz de Confusão

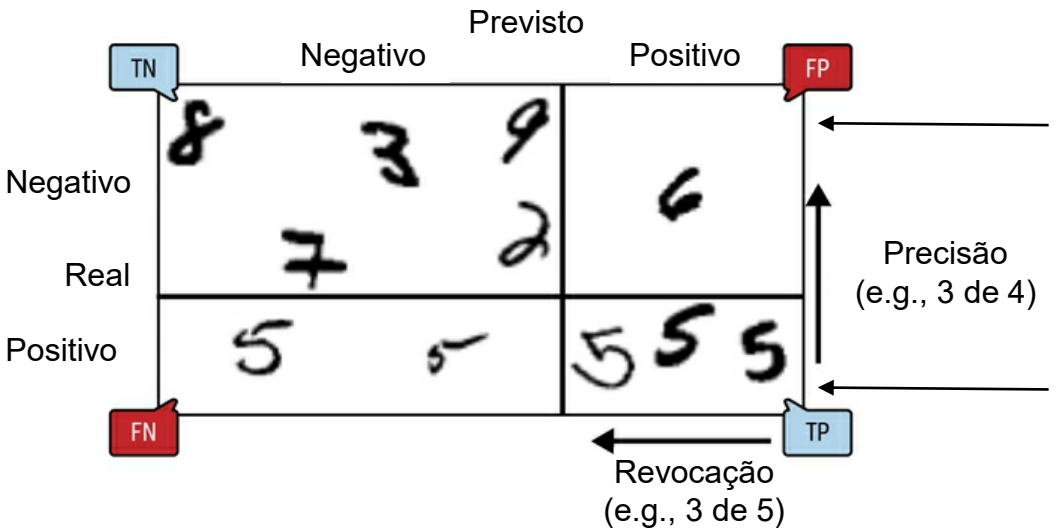
Um algoritmo classificador perfeito teria somente **verdadeiros positivos** (TP) e **verdadeiros negativos** (TN), ou seja, a matriz de confusão traria na diagonal principal (da esquerda superior para a direita inferior) números diferentes de zero e o restante seria zero. Os números indicados abaixo mostram os resultados esperados para um classificador perfeito.




TN: *True negative* (verdadeiro negativo)
FP: *False positive* (falso positivo)
FN: *False negative* (falso negativo)
TP: *True positive* (verdadeiro positivo)

## Matriz de Confusão

A figura abaixo ilustra instâncias classificadas pelo modelo de aprendizado de máquina. A matriz de confusão ilustrada mostra exemplos de verdadeiros negativos (canto superior esquerdo), falsos positivos (canto superior direito), falsos negativos (canto inferior esquerdo) e verdadeiros positivos (canto inferior direito)

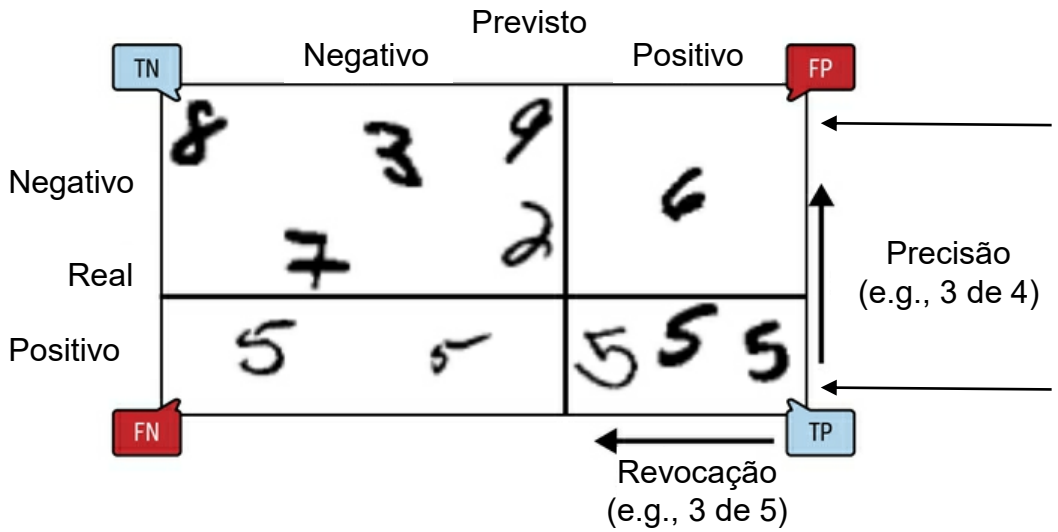



TN: *True negative* (verdadeiro negativo)
FP: *False positive* (falso positivo)
FN: *False negative* (falso negativo)
TP: *True positive* (verdadeiro positivo)

## Precisão, Revocação e Score $F_1$

Uma métrica interessante é a acurácia das predições positivas, que se chama precisão (*precision*) do classificador indicada pela equação abaixo. Como indicado na legenda à esquerda abaixo, TP representa o número de verdadeiros positivos, e FP designa o número de falsos positivos. A precisão normalmente é empregada com outra métrica denominada em português de revocação (*recall*). A **revocação** é também chamada de **sensibilidade** ou **taxa de verdadeiros positivos** (*true positive rate*) (TPR): essa é a proporção de instâncias positivas que são detectadas corretamente pelo classificador.

$$Precisão = \frac{TP}{TP + FP} \qquad \text{Revocação} = \frac{TP}{TP + FN}$$




A primeira linha desta matriz considera imagens não-5 (a classe negativa).

A segunda linha considera as imagens dos 5s (a classe positiva)

TN: *True negative* (verdadeiro negativo)
FP: *False positive* (falso positivo)
FN: *False negative* (falso negativo)
TP: *True positive* (verdadeiro positivo)

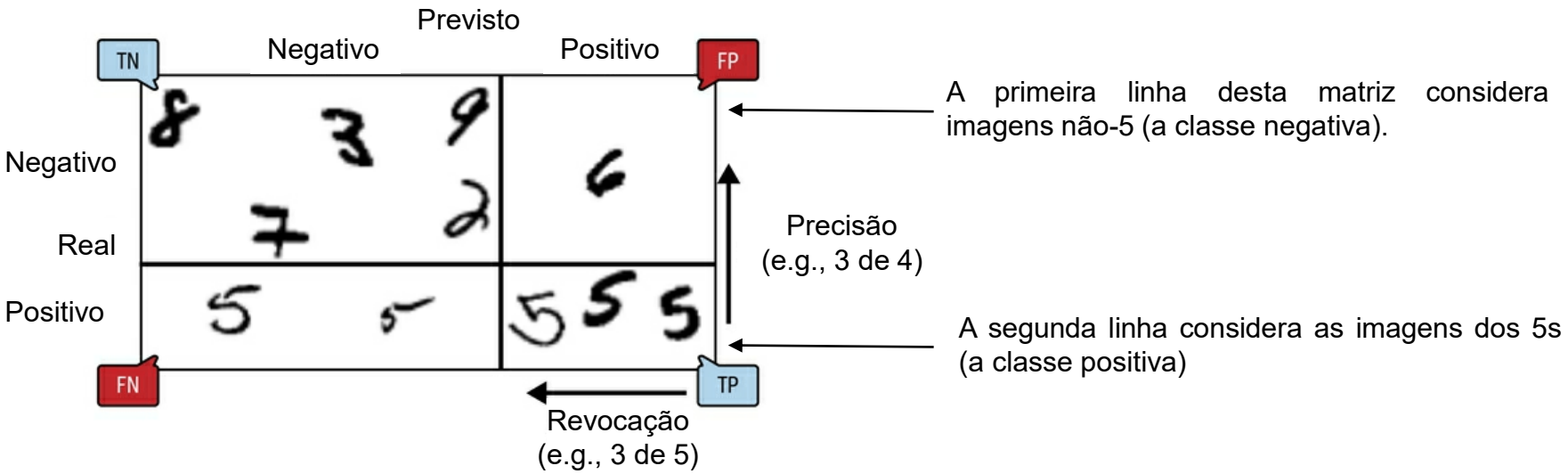
## Precisão, Revocação e Score $F_1$

Geralmente, é pertinente combinar a precisão e a revocação em uma única métrica que se chama $F_1$ score [pontuação $F_1$], ainda mais se você precisar de uma maneira simples de comparar dois classificadores. A $F_1$ score é a média harmônica da precisão e revocação (equação abaixo). Ao passo que a média regular trata igualmente todos os valores, a média harmônica dá mais importância aos valores mais baixos. Como resultado, o classificador só obterá um score $F_1$ alto se a revocação e a precisão forem altas.

$$F_1 = \frac{2}{\frac{1}{precisão} + \frac{1}{revocação}} = 2 \times \frac{precisão \times revocação}{precisão + revocação} = \frac{TP}{TP + \frac{FN + FP}{2}}$$

$$Precisão = \frac{TP}{TP + FP}$$

$$\text{Revocação} = \frac{TP}{TP + FN}$$



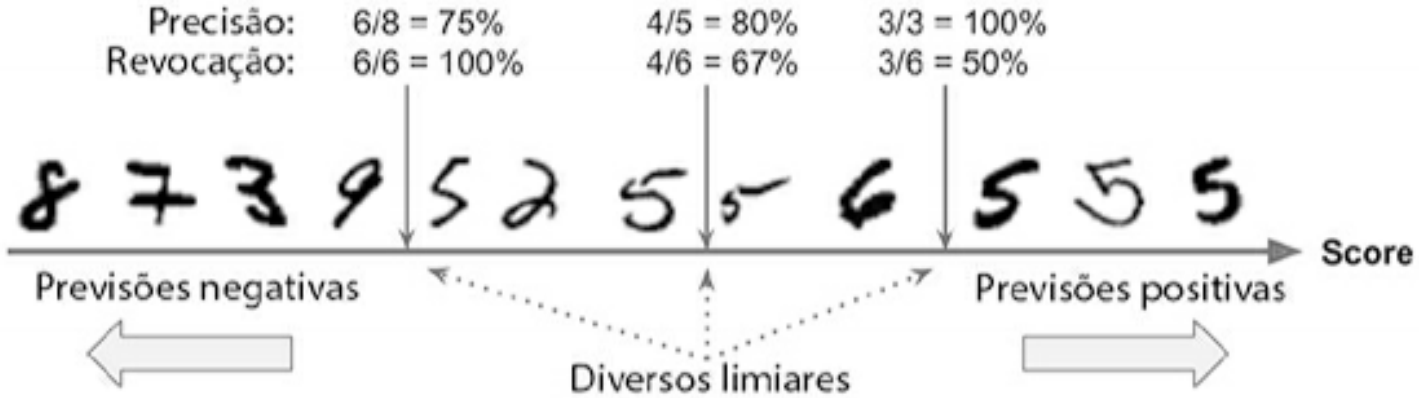

TN: *True negative* (verdadeiro negativo)
FP: *False positive* (falso positivo)
FN: *False negative* (falso negativo)
TP: *True positive* (verdadeiro positivo)

## Precisão, Revocação e Score $F_1$

Com o intuito de entender o trade-off, vejamos como o *SGDClassifier* toma suas decisões de classificação. Para cada instância, ele calcula um score baseado em uma função de decisão e, se esse score for maior que um limiar, ele atribui a instância à classe positiva, ou então a atribui à classe negativa. A figura abaixo ilustra alguns algarismos, desde o score mais baixo à esquerda até o mais alto à direita. Suponha que o limiar de decisão esteja posicionado na seta central (entre os dois 5s): você encontrará 4 positivos verdadeiros (5s reais) à direita desse limiar e 1 falso positivo (na verdade um 6). Portanto, com esse limiar, a precisão é de 80% (4 de 5). Mas, quando se trata dos 6 dos 5s reais, o classificador detecta apenas 4, logo, a revocação é de 67% (4 de 6). Se você aumentar o limiar (movendo a seta à direita), o falso positivo (o 6) se tornará um verdadeiro negativo, aumentando assim a precisão (até 100%, nesse caso), mas um positivo verdadeiro se tornará um falso negativo, diminuindo a revocação para 50%. Por outro lado, diminuir o limiar aumenta a revocação e reduz a precisão. Nos próximos slides mostramos os três casos em detalhes.

$$Precisão = \frac{TP}{TP + FP}$$

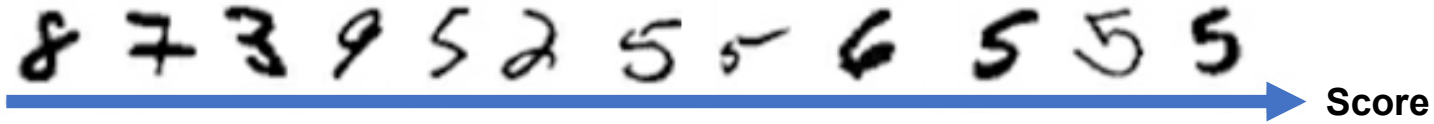
$$\text{Revocação} = \frac{TP}{TP + FN}$$




Neste trade-off de precisão/revocação, as imagens são classificadas por meio do score do classificador e aquelas acima do limiar de decisão escolhido são consideradas positivas; quanto maior o limiar, menor a revocação e maior (em geral) a precisão.

## Precisão, Revocação e Score $F_1$

Vamos calcular os valores da precisão e revocação para três casos hipotéticos com limiares (*thresholds*) distintos. O conjunto de dados tem 12 algarismos mostrados abaixo. Uma análise preliminar geral indica que temos 6 números 5 no nosso conjunto de dados. Não se esqueça, o nosso classificador é um detector de números 5.




$$Precisão = \frac{TP}{TP + FP}$$

$$\text{Revocação} = \frac{TP}{TP + FN}$$

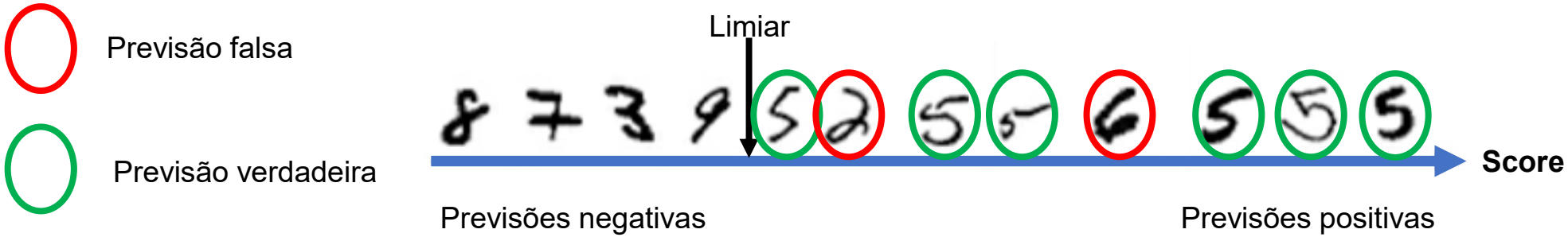
$$F_1 = \frac{2}{\frac{1}{precisão} + \frac{1}{revocação}}$$

TN: *True negative* (verdadeiro negativo)
FP: *False positive* (falso positivo)
FN: *False negative* (falso negativo)
TP: *True positive* (verdadeiro positivo)

## Precisão, Revocação e Score $F_1$

### Caso 1

No caso 1 temos o limiar indicado abaixo pela seta entre os números 9 e 5. À direita da seta, vemos 6 números 5, ou seja, temos um total de positivos verdadeiros de 6 (TP = 6). Com esse limiar todos os números 5 estão do lado direito. A análise dos dados do lado direito indica dois falsos positivos (FP = 2). Aqui temos uma precisão de 75 %.




$$Precisão = \frac{TP}{TP + FP} = \frac{6}{6 + 2} = 0{,}75$$

$$F_1 = \frac{2}{\frac{1}{precisão} + \frac{1}{revocação}}$$

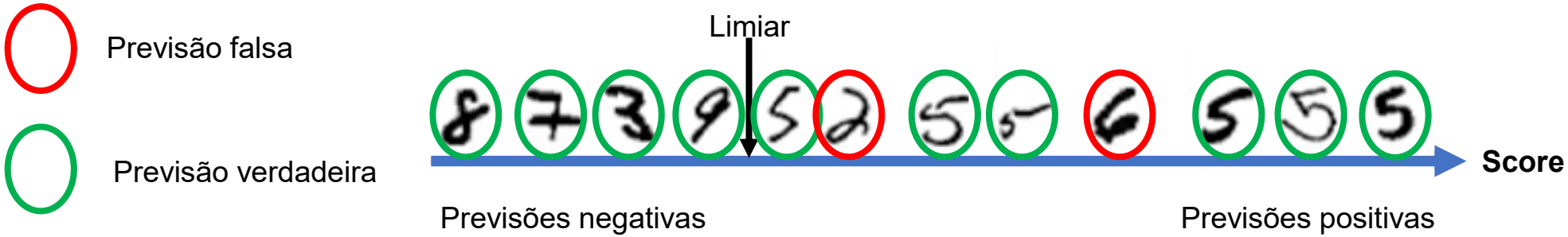
$$\text{Revocação} = \frac{TP}{TP + FN}$$

TN: *True negative* (verdadeiro negativo)
FP: *False positive* (falso positivo)
FN: *False negative* (falso negativo)
TP: *True positive* (verdadeiro positivo)

## Precisão, Revocação e Score $F_1$

### Caso 1

A análise dos dados do lado esquerdo indica que temos zero falsos negativos (FN = 0). A revocação é 100 %. Usando as informações da precisão e revocação chegamos a um score $F_1$ de 85,71%.




$$Precisão = \frac{TP}{TP + FP} = \frac{6}{6 + 2} = 0{,}75$$

$$\text{Revocação} = \frac{TP}{TP + FN} = \frac{6}{6 + 0} = 1{,}0$$

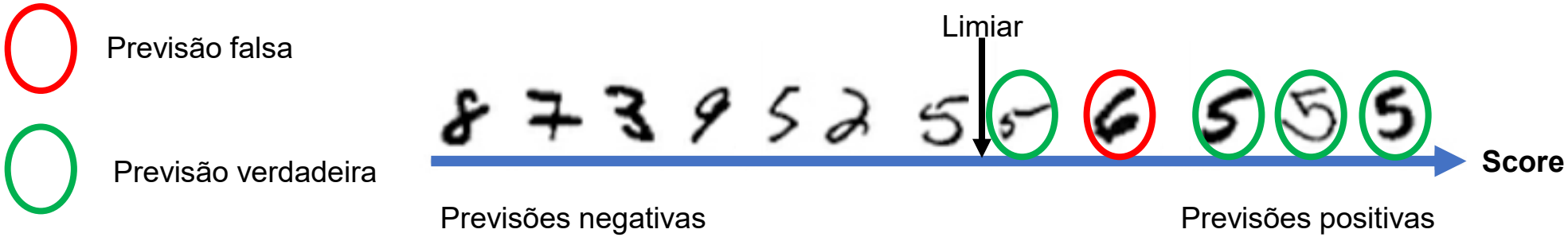
$$F_1 = \frac{2}{\frac{1}{precisão} + \frac{1}{revocação}} = \frac{2}{\frac{1}{0{,}75} + \frac{1}{1{,}0}} = 0{,}8571$$

TN: *True negative* (verdadeiro negativo)
FP: *False positive* (falso positivo)
FN: *False negative* (falso negativo)
TP: *True positive* (verdadeiro positivo)

## Precisão, Revocação e Score $F_1$

### Caso 2

O limiar está colocado entre os 5s indicados abaixo. À direita temos as previsões positivas, com um total de 4 acertos, ou seja, o número de verdadeiros positivos é 4 (TP = 4). Ainda olhando os dados do lado direito do limiar, vemos um número 6 identificado como 5, ou seja, um falso positivo. Assim, o número de falsos positivos é 1 (FP = 1), podemos calcular a precisão, como indicado abaixo. Temos uma precisão de 80 % para este caso.




$$Precisão = \frac{TP}{TP + FP} = \frac{4}{4 + 1} = 0{,}8$$

$$\text{Revocação} = \frac{TP}{TP + FN}$$

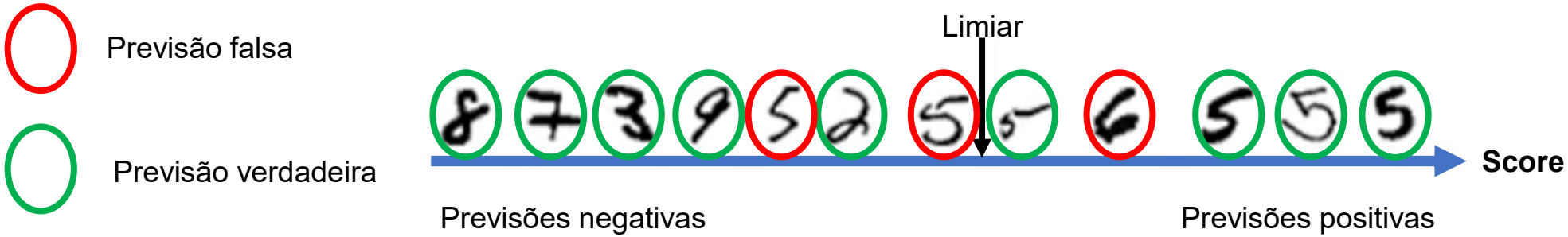
$$F_1 = \frac{2}{\frac{1}{precisão} + \frac{1}{revocação}}$$

TN: *True negative* (verdadeiro negativo)
FP: *False positive* (falso positivo)
FN: *False negative* (falso negativo)
TP: *True positive* (verdadeiro positivo)

## Precisão, Revocação e Score $F_1$

### Caso 2

Como já destacado, temos um total 6 números 5. Neste caso vemos que 4 foram previstos corretamente e 2 erroneamente, ou seja, são falsos negativos (FN = 2). Agora podemos calcular a revocação, como indicado no cálculo abaixo. Temos uma revocação de 66,7 %. A partir da revocação e precisão temos um score $F_1$ de 72,7 %.




$$Precisão = \frac{TP}{TP + FP} = \frac{4}{4 + 1} = 0{,}8$$

$$\text{Revocação} = \frac{TP}{TP + FN} = \frac{4}{4 + 2} = 0{,}667$$

$$F_1 = \frac{2}{\frac{1}{precisão} + \frac{1}{revocação}} = \frac{2}{\frac{1}{0{,}8} + \frac{1}{0{,}667}} = 0{,}727$$

TN: *True negative* (verdadeiro negativo)
FP: *False positive* (falso positivo)
FN: *False negative* (falso negativo)
TP: *True positive* (verdadeiro positivo)

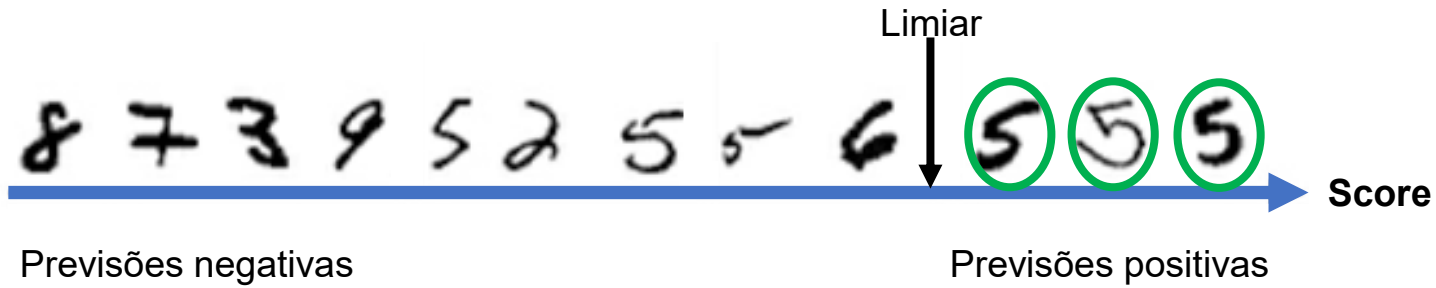
## Precisão, Revocação e Score $F_1$

### Caso 3

O limiar está entre os números 6 e 5, como indicado abaixo. À direita do limiar temos as previsões positivas, com um total de 3 acertos. Agora temos o número de verdadeiros positivos de 3 (TP = 3). Ainda focando nos dados do lado direito do limiar, vemos que todos foram identificados como 5. Assim, o número de falsos positivos é 0 (FP = 0). Temos uma precisão de 100 %.

Previsão falsa

Previsão verdadeira




$$Precisão = \frac{TP}{TP + FP} = \frac{3}{3 + 0} = 1{,}0$$

$$F_1 = \frac{2}{\frac{1}{precisão} + \frac{1}{revocação}}$$

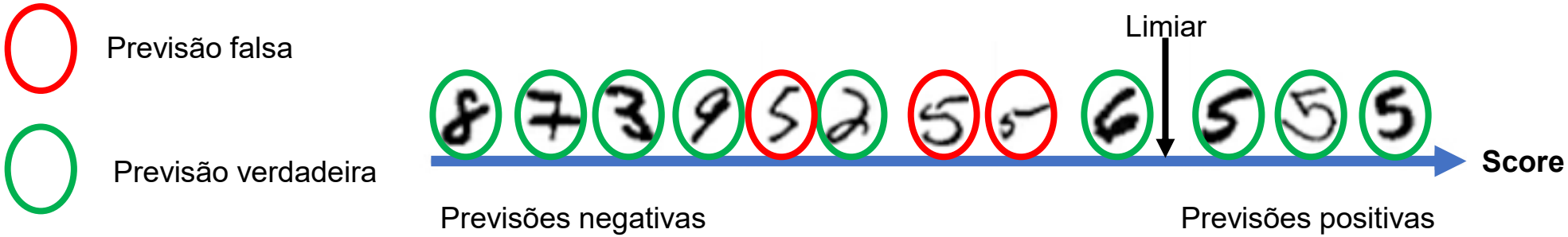
$$\text{Revocação} = \frac{TP}{TP + FN}$$

TN: *True negative* (verdadeiro negativo)
FP: *False positive* (falso positivo)
FN: *False negative* (falso negativo)
TP: *True positive* (verdadeiro positivo)

## Precisão, Revocação e Score $F_1$

### Caso 3

Neste limiar temos três 5 à esquerda, ou seja, temos três falsos negativos (FN = 3). Calculando-se a revocação chegamos a 50 %. A movimentação do limiar para a direita aumentou a precisão e diminuiu a revocação. Temos um score $F_1$ de 66,7 %.




$$Precisão = \frac{TP}{TP + FP} = \frac{3}{3 + 0} = 1{,}0$$

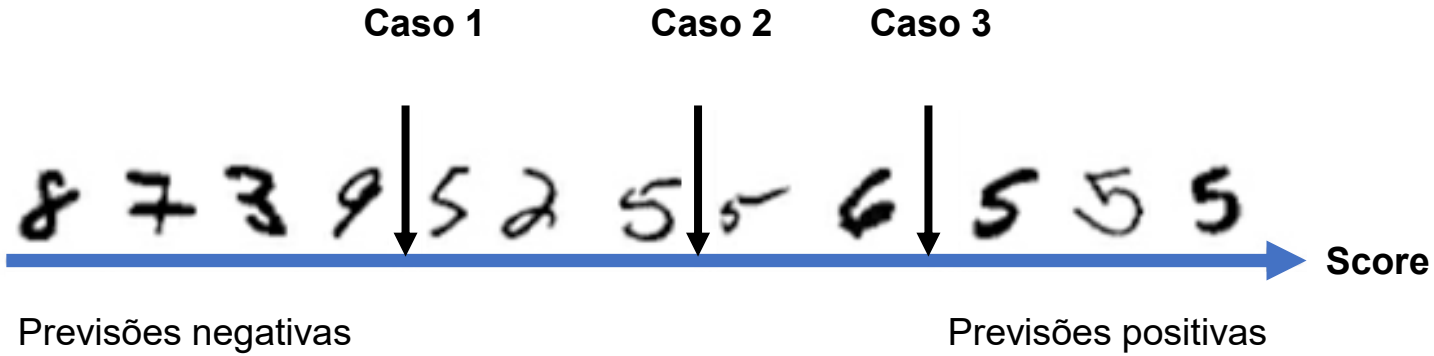
$$\text{Revocação} = \frac{TP}{TP + FN} = \frac{3}{3 + 3} = 0{,}5$$

$$F_1 = \frac{2}{\frac{1}{precisão} + \frac{1}{revocação}} = \frac{2}{\frac{1}{1{,}0} + \frac{1}{0{,}5}} = 0{,}667$$

TN: *True negative* (verdadeiro negativo)
FP: *False positive* (falso positivo)
FN: *False negative* (falso negativo)
TP: *True positive* (verdadeiro positivo)

## Precisão, Revocação e Score $F_1$

Abaixo temos o resumo dos três casos analisados. Se usarmos o score $F_1$ como critério de seleção, vemos que o modelo do caso 1 tem o melhor desempenho. Há outras métricas para a análise de modelos de classificação, uma das mais usadas faz uso do gráfico chamado de **curva de característica de operação**.




$$Precisão = \frac{TP}{TP + FP}$$

$$\text{Revocação} = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2}{\frac{1}{precisão} + \frac{1}{revocação}}$$

TN: *True negative* (verdadeiro negativo)
FP: *False positive* (falso positivo)
FN: *False negative* (falso negativo)
TP: *True positive* (verdadeiro positivo)

| Caso | TP | FP | FN | Precisão | Revocação | $F_1$ |
|---|---|---|---|---|---|---|
| 1 | 6 | 2 | 0 | 0,750 | 1,000 | 0,857 |
| 2 | 4 | 1 | 2 | 0,800 | 0,667 | 0,727 |
| 3 | 3 | 0 | 3 | 1,000 | 0,500 | 0,667 |

## Curva ROC

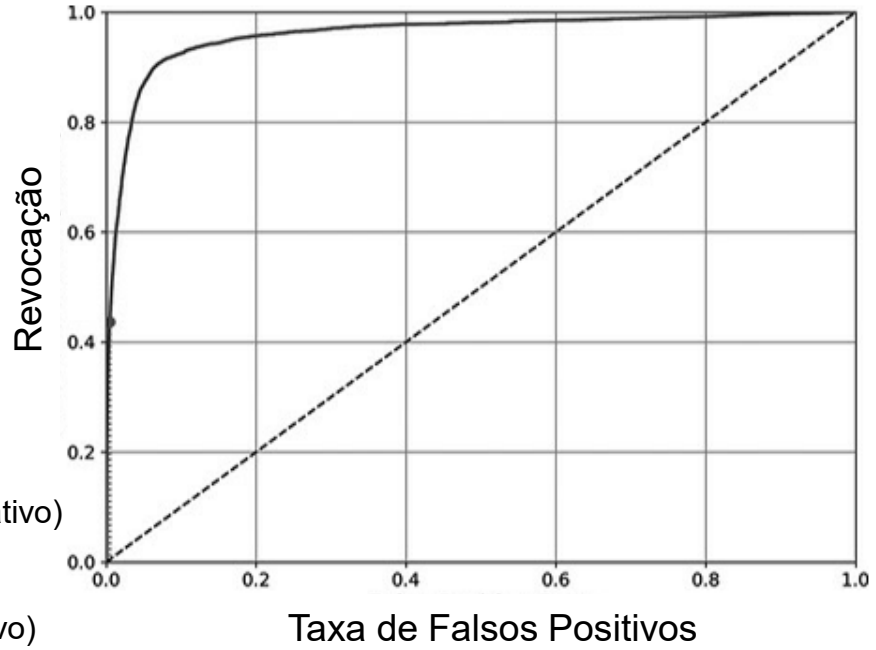
Na análise do poder de previsão de classificadores focados em sistemas complexos, é recomendado o uso da curva ROC (Walsh et 2021). Em português denominamos essa métrica de **curva de característica de operação**. O acrônimo em inglês ROC vem do termo *receiver operating characteristic* (ROC). A curva ROC é o gráfico da revocação (taxa de verdadeiros positivos) (TPR) com a taxa de positivos falsos (*false positive rate*) (FPR). O FPR indica a proporção de instâncias negativas classificadas incorretamente como positivas. É igual a 1 — a taxa de verdadeiros negativos (TNR), que é a proporção de instâncias negativas que são corretamente classificadas como negativas. O TNR também recebe a denominação de especificidade. Podemos dizer que a curva ROC traz o gráfico da sensibilidade (revocação) versus 1 — a especificidade.

$$FPR = \frac{FP}{FP + TN}$$

$$Precisão = \frac{TP}{TP + FP}$$

$$\text{Revocação} = \frac{TP}{TP + FN}$$

TN: *True negative* (verdadeiro negativo)
FP: *False positive* (falso positivo)
FN: *False negative* (falso negativo)
TP: *True positive* (verdadeiro positivo)

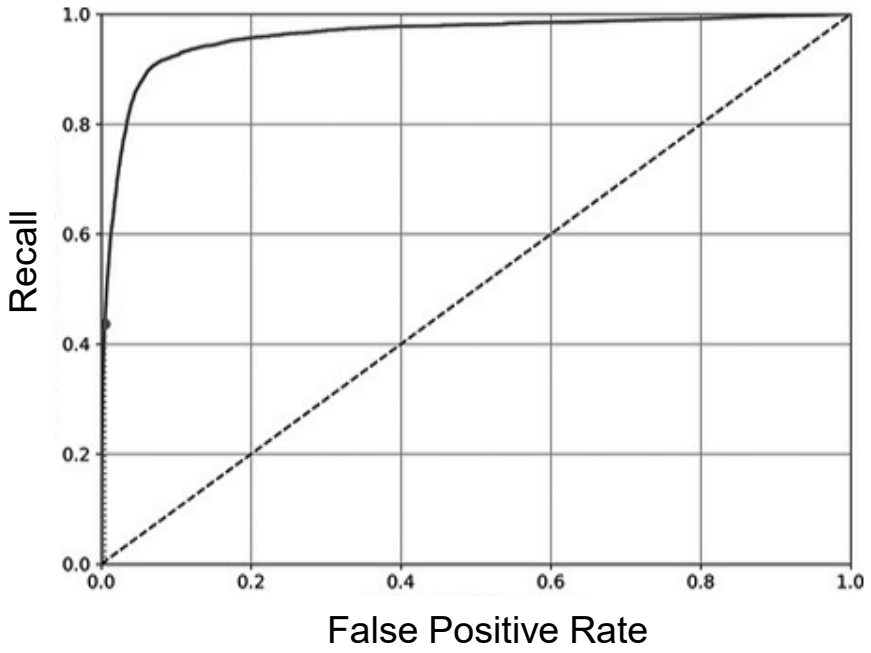



A linha pontilhada representa a curva ROC de um classificador exclusivamente aleatório; um bom classificador fica o mais distante possível dessa linha (em direção ao canto superior esquerdo).

Géron, Aurélien. Mãos A Obra: Aprendizado De Máquina Com Scikit-Learn, Keras & TensorFlow: Conceitos, Ferramentas e Técnicas Para a Construção de Sistemas Inteligentes (Portuguese Edition) (p. 177). Alta Books. Edição do Kindle.

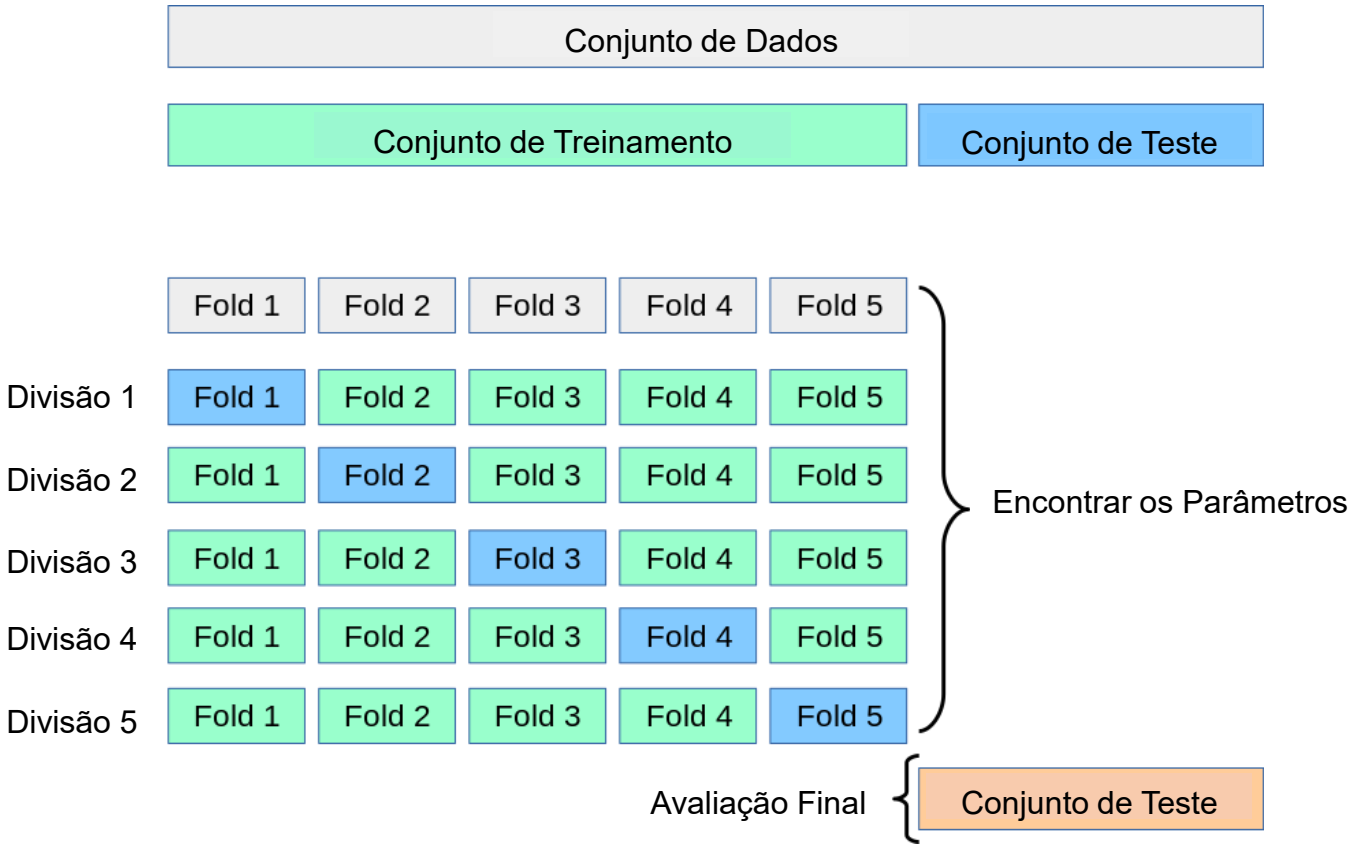
## Curva ROC

Em 2021 Walsh et al. propuseram um conjunto de métricas para avaliação de modelos de aprendizado de máquina construídos com foco em sistemas biológicos. Especificamente para problemas de classificação, os autores recomenda o uso da curva ROC e da área sob a curva (sigla em inglês AUC (*area under the curve*)). De uma forma geral, a análise da curva ROC é um dos métodos mais usados para avaliação de modelos de classificação. Uma forma de avaliar o poder de previsão dos classificadores é determinar a AUC. Um classificador perfeito tem AUC = 1. O classificador aleatório indicado pela linha tracejada tem uma AUC = 0,5.



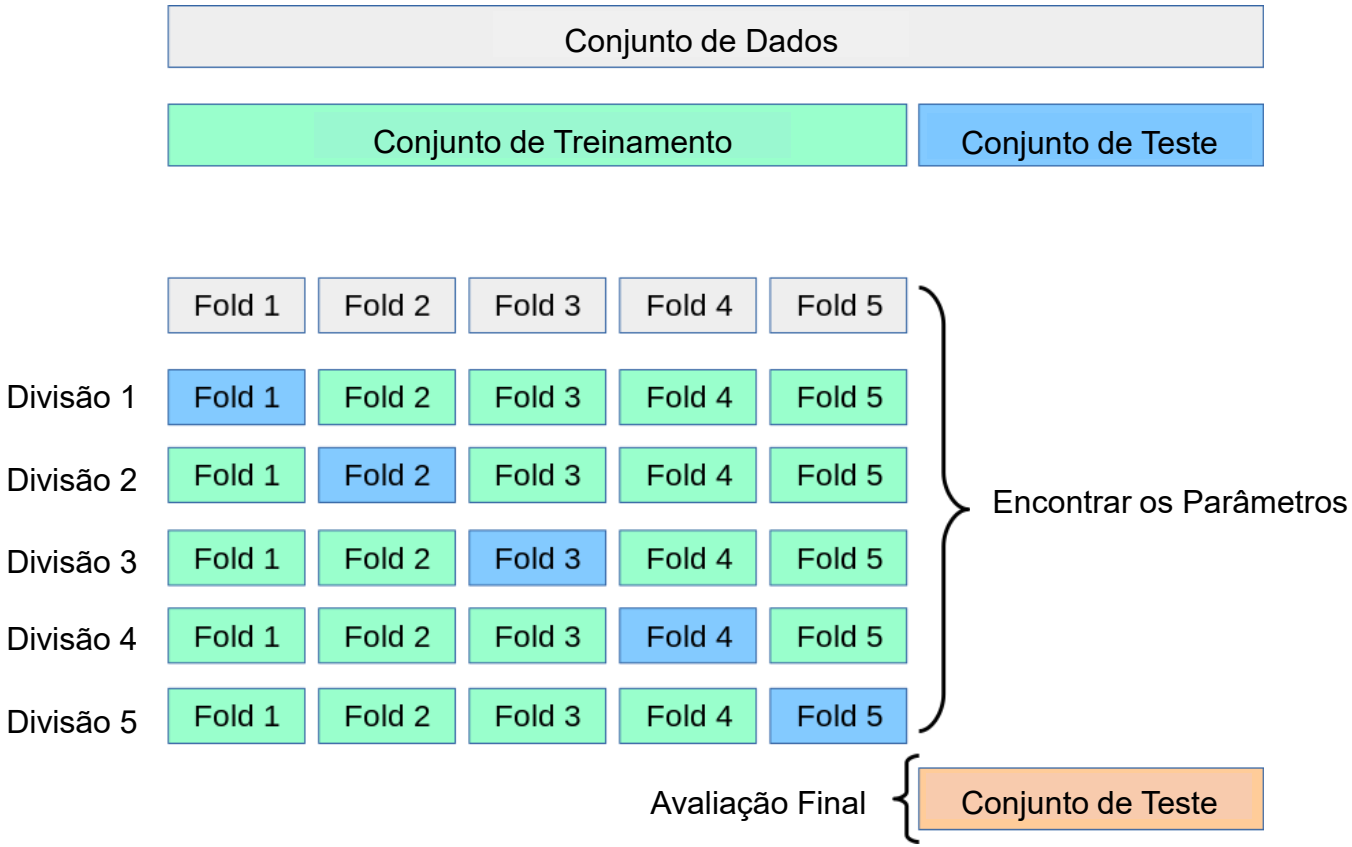
## Método *k-fold* de Validação Cruzada

Na elaboração de modelos de aprendizado de máquina, normalmente dividimos os conjunto de dados em dois grupos. Um chamado de conjunto de treinamento e outro chamado de conjunto de teste. O conjunto de treinamento do conjunto de dados analisado aqui é formado pelas 60 mil instâncias iniciais e o conjunto de teste das últimas 10 mil. O conjunto de treinamento pode ser dividido para aumentar o poder de previsão dos modelos gerado. Esse processo é chamado de validação cruzada (*cross validation*). O diagrama esquemático abaixo ilustra a ideia geral do método *k-fold* de validação cruzada.



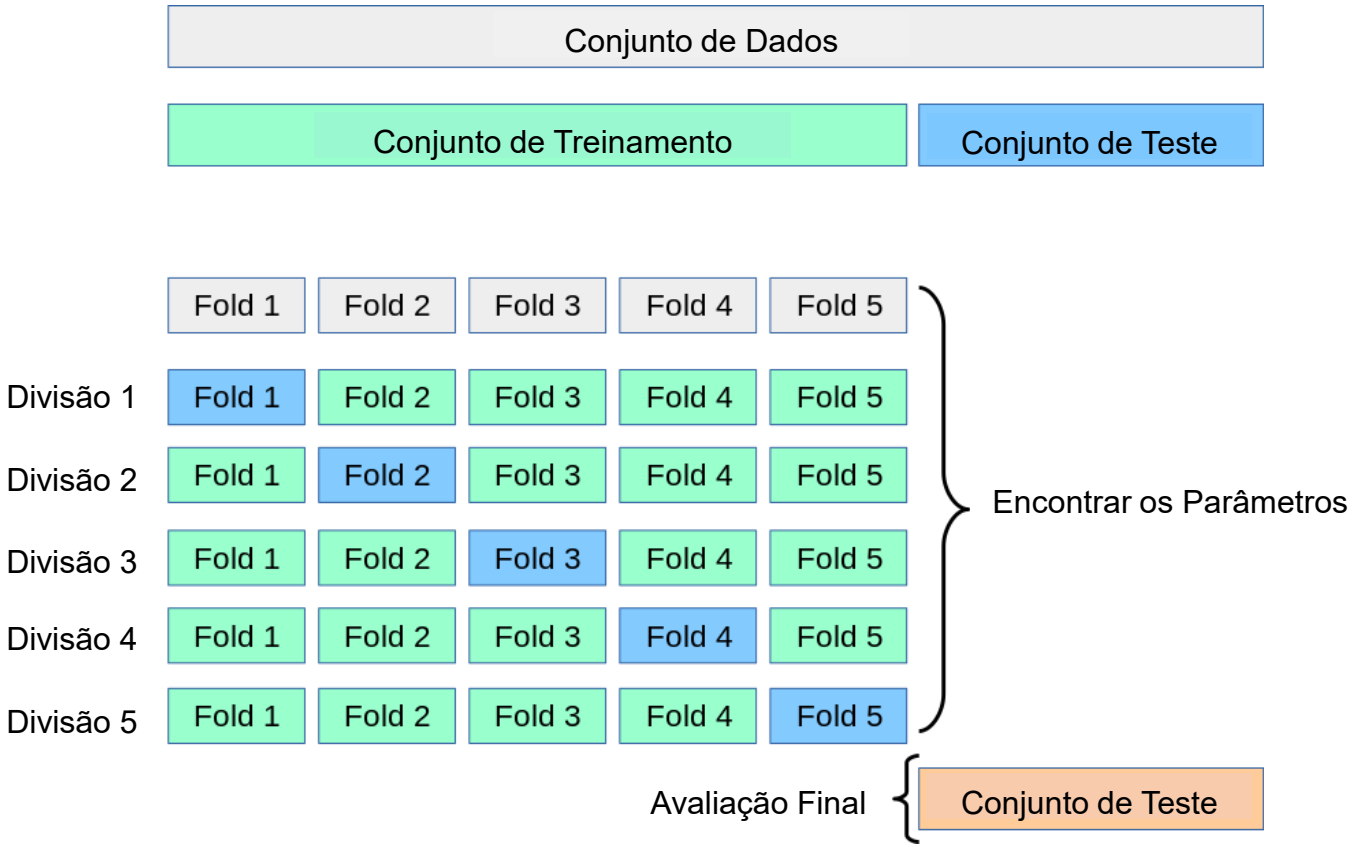
## Método *k-fold* de Validação Cruzada

O método *k-fold* consiste em dividir o conjunto de treinamento em k subconjuntos (chamados aqui de *folds*). O método é chamado em inglês de *k-fold cross validation*. O treinamento é feito no restante k-1 subconjuntos. A avaliação do poder de previsão é feita com os dados deixados de fora do treinamento. Por exemplo, com os dados da divisão 1, usamos os subconjuntos de 2 a 5 (*Fold 2, Fold 3, Fold 4* e *Fold 5*) para o treinamento (determinação dos parâmetros) e o subconjunto 1 (*Fold 1*) para avaliar a métricas. Podemos pensar que o conjunto *Fold 1* é um conjunto de teste local para a divisão 1. O processo é repetido para as outras divisões.



## Método *k-fold* de Validação Cruzada

A métrica final é a média obtida. O modelo gerado é o modelo médio das k divisões. No final, avaliamos o poder de previsão do modelo gerado com o conjunto de teste original.



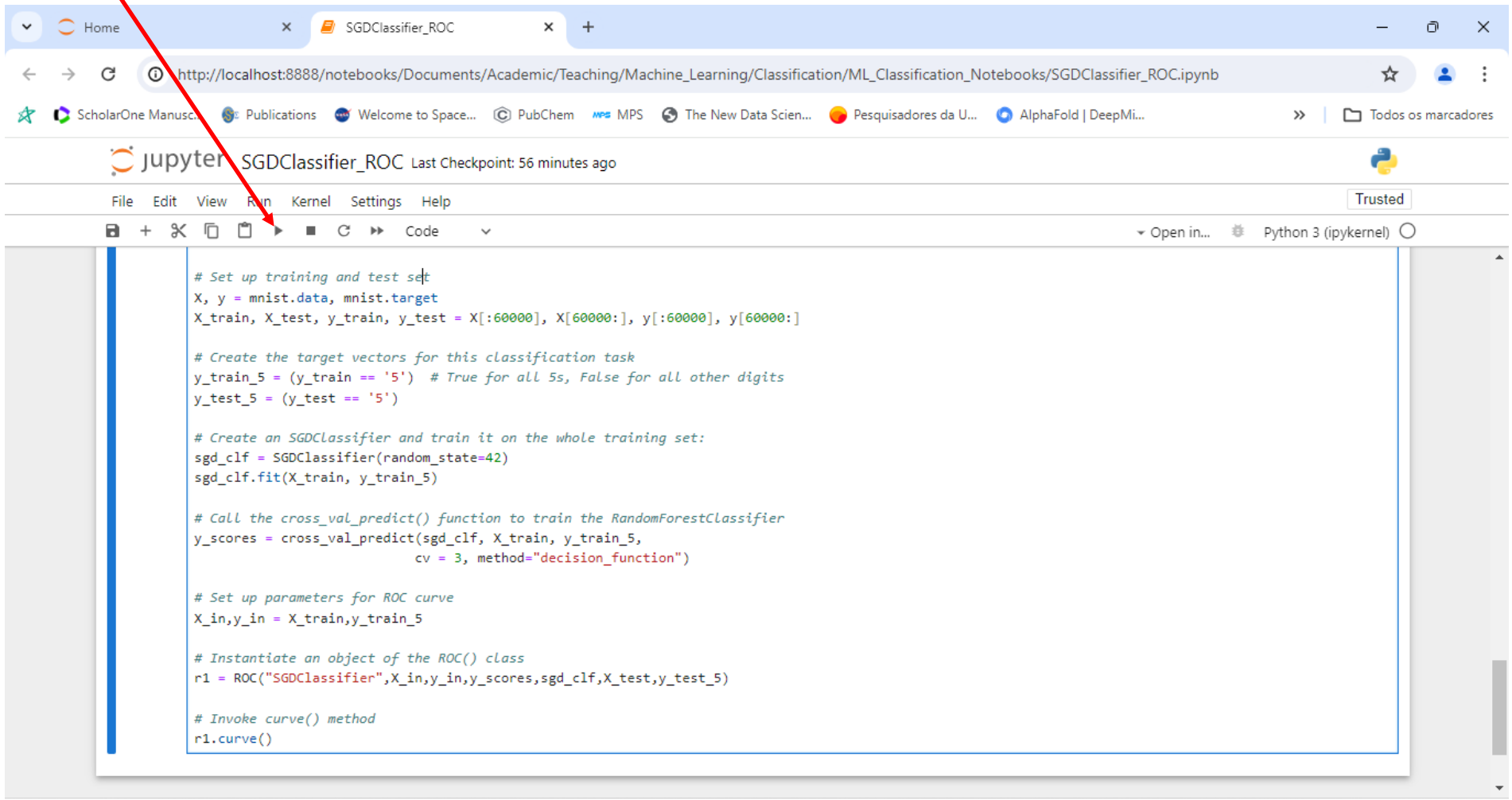
## Classificador Gradiente Descendente Estocástico

Agora veremos o código do classificador gradiente descendente estocástico. O código prevê se o dígito é 5 ou não. Como temos duas situações, denominamos esse classificador de binário. Abra o código *SGDClassifier_ROC.ipynb* com o Jupyter. O modelo gerado pelo código usa validação cruzada *k-fold* com 3 divisões.


Fonte: https://pixabay.com/illustrations/technology-robot-machine-5917370/

## Classificador Gradiente Descendente Estocástico

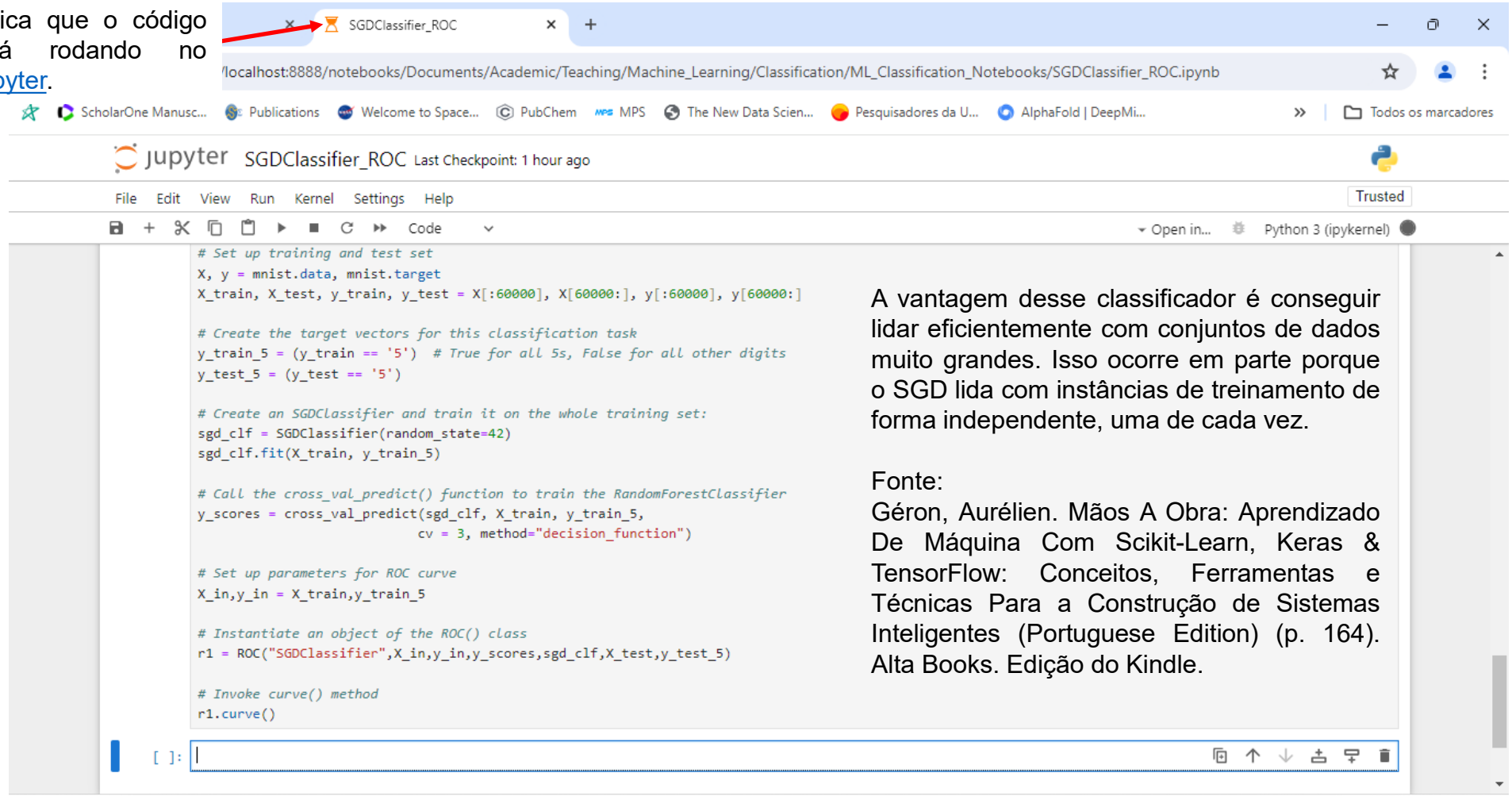
Abaixo temos a janela do Jupyter com o código aberto. Clique em qualquer parte do código. Em seguida clique no botão para executar o código. **A execução do código pode demorar alguns minutos.**



# Classificador Gradiente Descendente Estocástico

O código traz o método de classificação chamado **classificador gradiente descendente estocástico** (em inglês a *stochastic gradient descent* (SGD, ou *stochastic* GD)), a partir da classe *SGDClassifier* da Scikit-Learn.

Indica que o código está rodando no Jupyter.



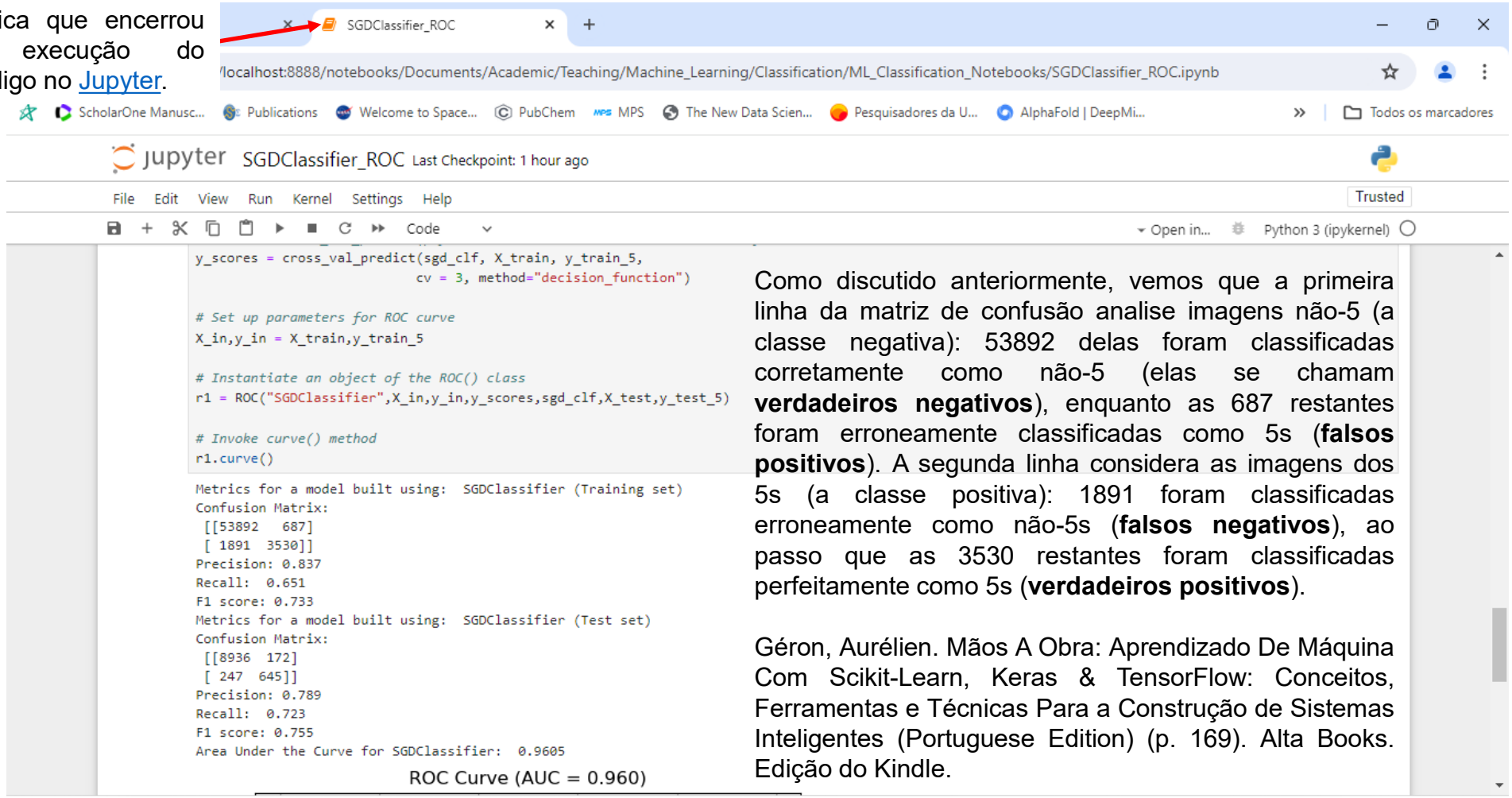

A vantagem desse classificador é conseguir lidar eficientemente com conjuntos de dados muito grandes. Isso ocorre em parte porque o SGD lida com instâncias de treinamento de forma independente, uma de cada vez.

Fonte:
Géron, Aurélien. Mãos A Obra: Aprendizado De Máquina Com Scikit-Learn, Keras & TensorFlow: Conceitos, Ferramentas e Técnicas Para a Construção de Sistemas Inteligentes (Portuguese Edition) (p. 164). Alta Books. Edição do Kindle.

# Classificador Gradiente Descendente Estocástico

Ao finalizar desaparece a ampulheta e as métricas e a curva ROC são mostradas. Determinamos a matriz de confusão, precisão (*precision*), revocação (*recall*) e $F_1$ score.

Indica que encerrou a execução do código no Jupyter.



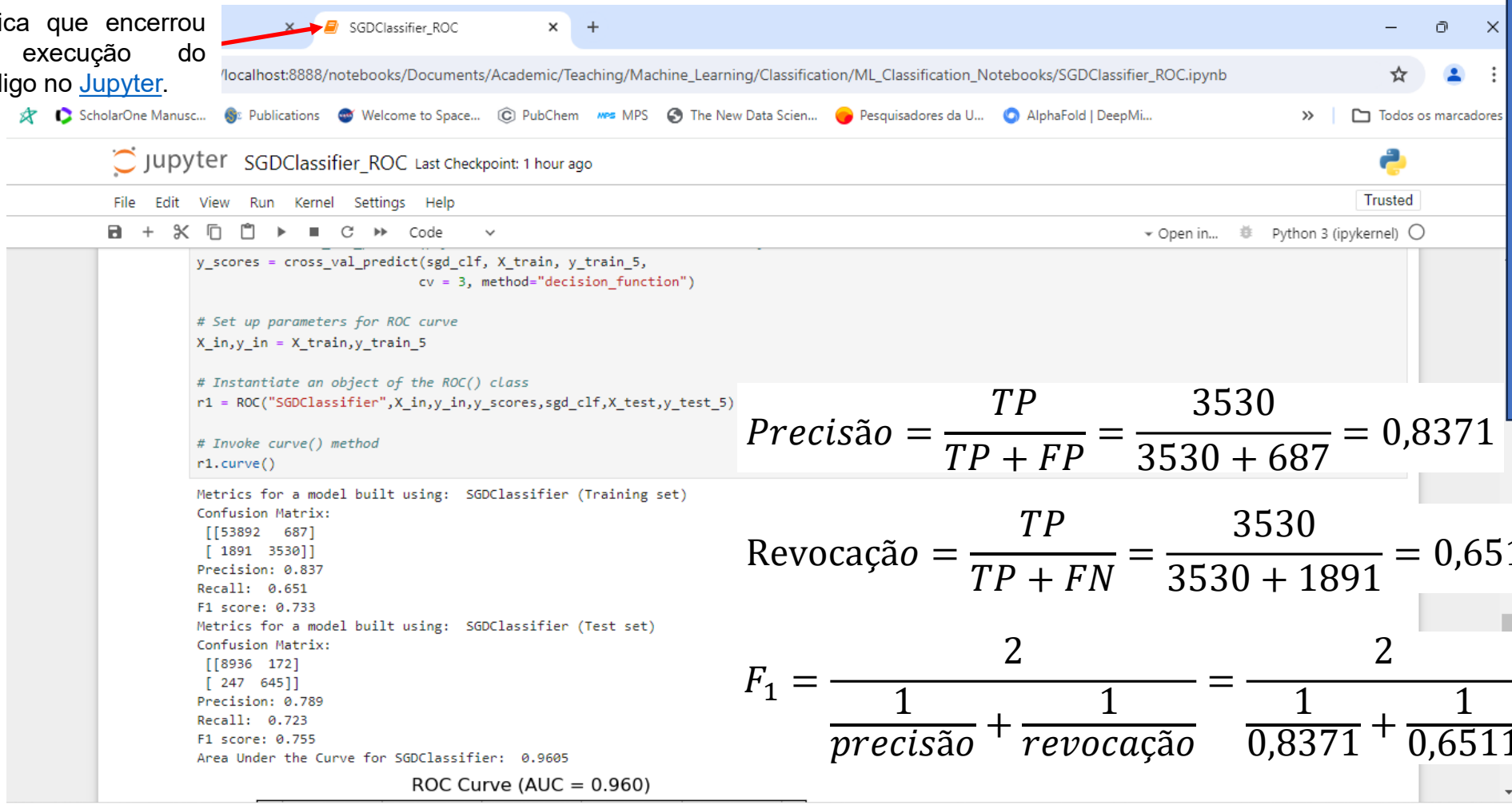

Como discutido anteriormente, vemos que a primeira linha da matriz de confusão analise imagens não-5 (a classe negativa): 53892 delas foram classificadas corretamente como não-5 (elas se chamam **verdadeiros negativos**), enquanto as 687 restantes foram erroneamente classificadas como 5s (**falsos positivos**). A segunda linha considera as imagens dos 5s (a classe positiva): 1891 foram classificadas erroneamente como não-5s (**falsos negativos**), ao passo que as 3530 restantes foram classificadas perfeitamente como 5s (**verdadeiros positivos**).

Géron, Aurélien. Mãos A Obra: Aprendizado De Máquina Com Scikit-Learn, Keras & TensorFlow: Conceitos, Ferramentas e Técnicas Para a Construção de Sistemas Inteligentes (Portuguese Edition) (p. 169). Alta Books. Edição do Kindle.

## Classificador Gradiente Descendente Estocástico

Nós podemos usar as equações vistas para confirmar se nosso código está gerando os resultados esperados a partir dos dados da matriz de confusão.

Indica que encerrou a execução do código no Jupyter.




| | |
|---|---|
| TN 53892 | FP 687 |
| FN 1891 | TP 3530 |

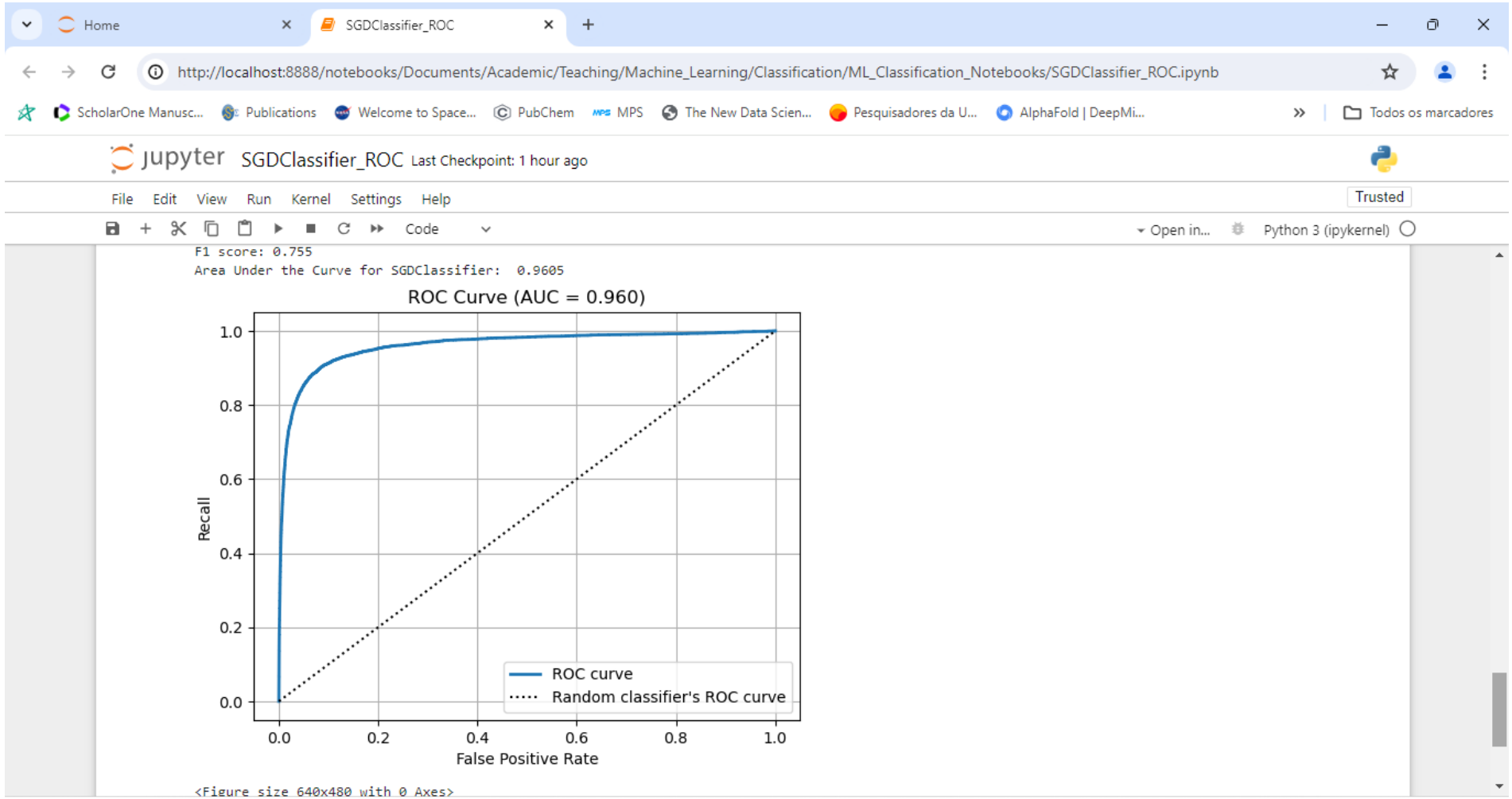
$$Precisão = \frac{TP}{TP + FP} = \frac{3530}{3530 + 687} = 0{,}8371$$

$$\text{Revocação} = \frac{TP}{TP + FN} = \frac{3530}{3530 + 1891} = 0{,}6511$$

$$F_1 = \frac{2}{\frac{1}{precisão} + \frac{1}{revocação}} = \frac{2}{\frac{1}{0{,}8371} + \frac{1}{0{,}6511}} = 0{,}73247$$

## Classificador Gradiente Descendente Estocástico

Rolando a barra vertical do navegador temos acesso à curva ROC. Temos um valor de AUC = 0,960.



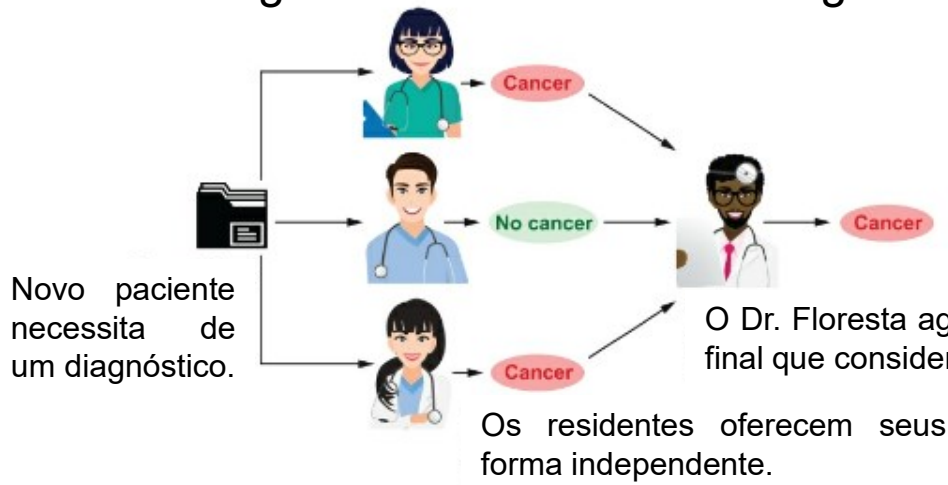
## Classificador Floresta Aleatória

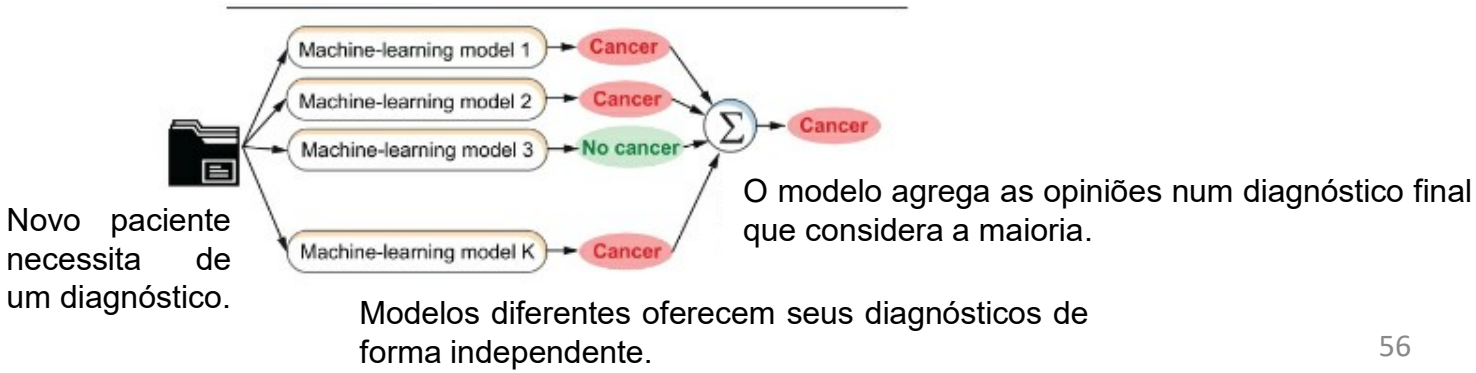
Agora veremos o código do classificador floresta aleatória. Como o código anterior, este prevê se o dígito é 5 ou não. Só que agora usamos como algoritmo classificador o *random forest* (floresta aleatória). A floresta aleatória faz parte de uma família de classificadores chamados de métodos ensemble. De uma maneira geral esses métodos consideram a ação de vários métodos e fazem a média dos seus resultados.

Fonte: https://pixabay.com/photos/forest-fog-woods-trees-mystical-3394066/

## Classificador Floresta Aleatória

De uma forma intuitiva podemos pensar que os métodos ensemble funcionam como um médico que orienta diversos especialistas, chamaremos aqui de médico orientador (ou Dr. Floresta). Esses especialistas são residentes num hospital escola e têm diversas especialidades distintas. Quando chegam os resultados dos exames de um dado paciente sem diagnóstico, o médico orientador (Dr. Floresta) passa os exames para todos os residentes. Depois considera as respostas de cada um. Na analogia com um classificador binário, o resultado mais comum é o considerado pelo médico orientador. A figura abaixo ilustra a analogia.

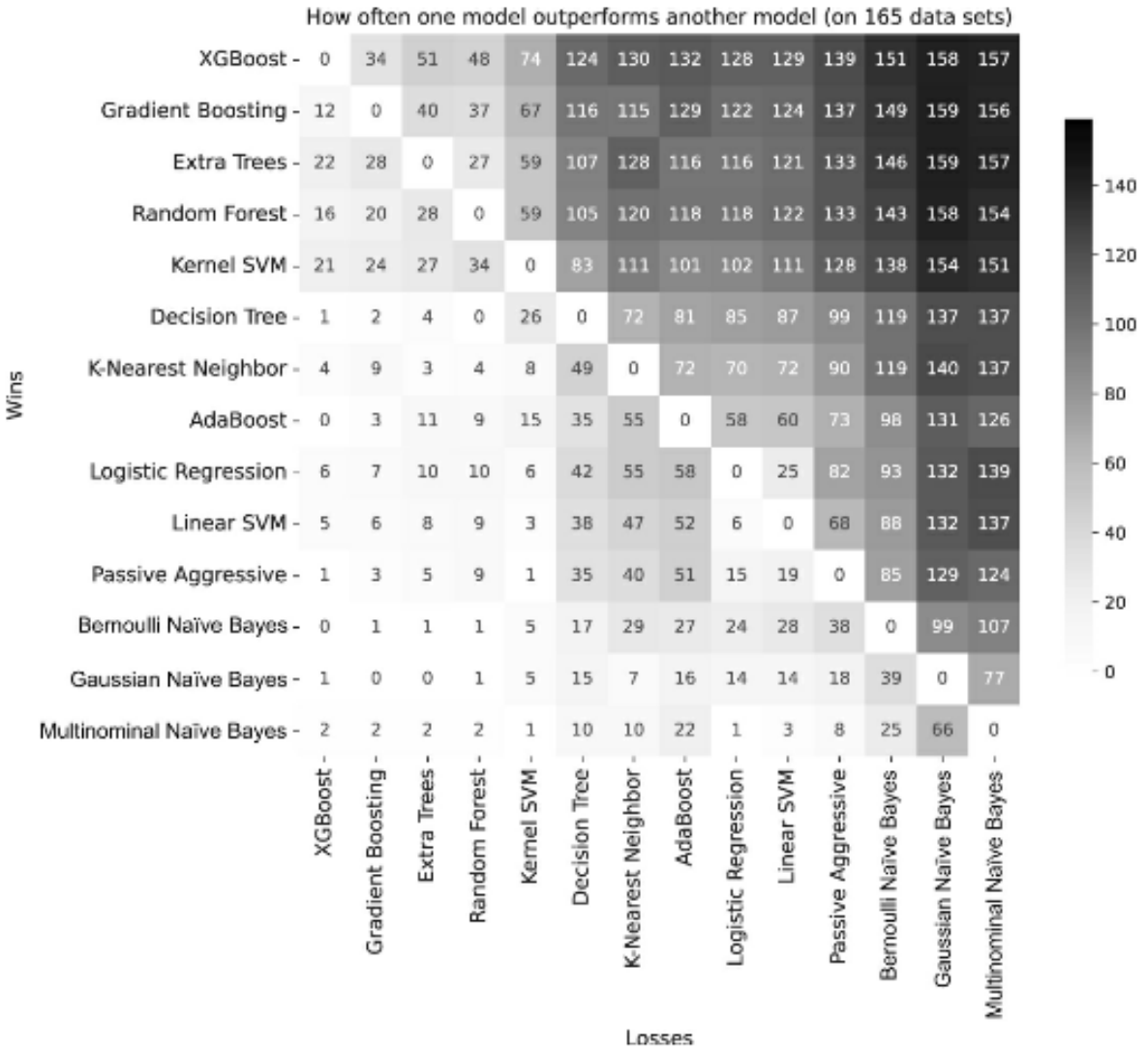



Fonte: Kunapuli, Gautam. Ensemble Methods for Machine Learning (p. 73). Manning. Edição do Kindle.

## Classificador Floresta Aleatória

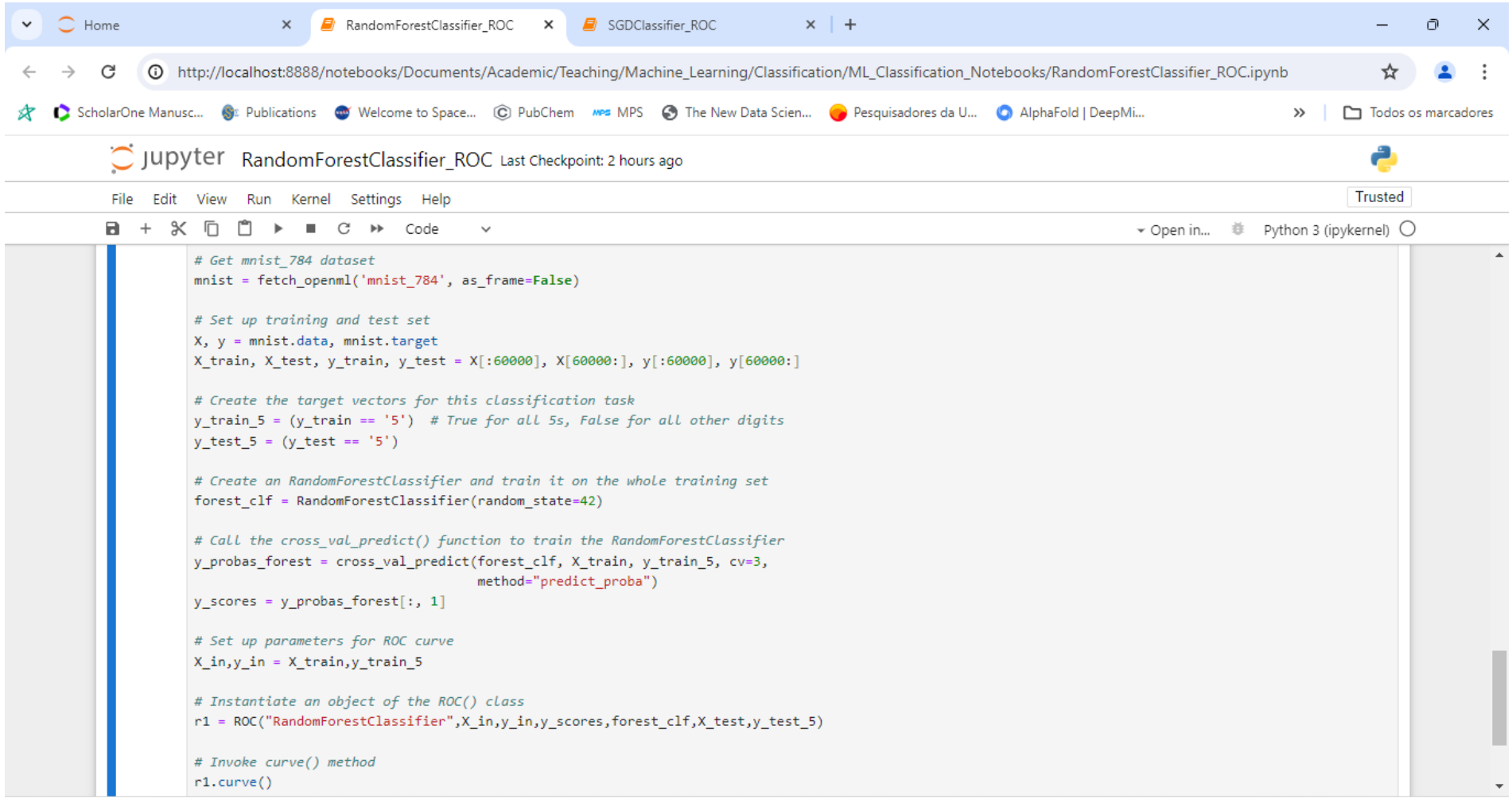
Um estudo comparativo do poder de previsão de classificadores distintos com 165 conjuntos de dados indicou que métodos ensemble apresentam resultados superiores (Kunapuli, 2023). No gráfico ao lado considere a comparação dos métodos XGBoost e *Gradient Boosting*. Na primeira linha segunda coluna vemos o número 34, que indica que dos 165 conjuntos de dados, para 34 conjuntos o método XGBoost foi melhor que o *Gradient Boosting*. Se olharmos a segunda linha e primeira coluna temos o número 12, que indica que o método *Gradient Boosting* funciona melhor em 12 conjuntos de dados. Para 119 conjuntos de dados os métodos têm desempenho similares. Assim, podemos dizer que o XGBoost é superior ao *Gradient Boosting*. Vejam que o método de floresta aleatória está num honroso quarto lugar.




Fonte: Kunapuli, Gautam. Ensemble Methods for Machine Learning (p. 36). Manning. Edição do Kindle.

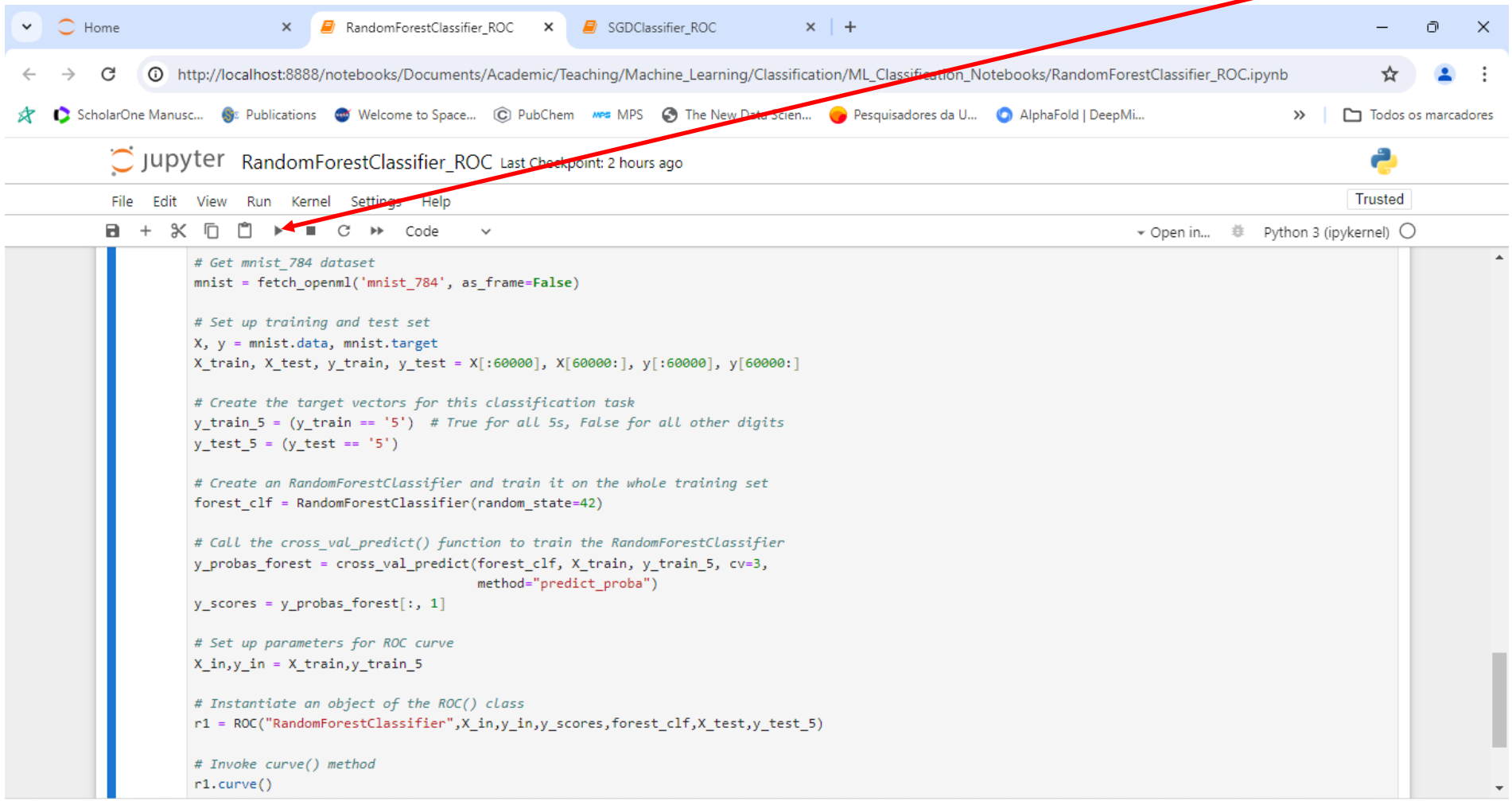
## Classificador Floresta Aleatória

Abra o código *RandomForestClassifier_ROC.ipynb* com o Jupyter. O modelo gerado pelo código usa validação cruzada *k-fold* com 3 divisões, como no código anterior.



## Classificador Floresta Aleatória

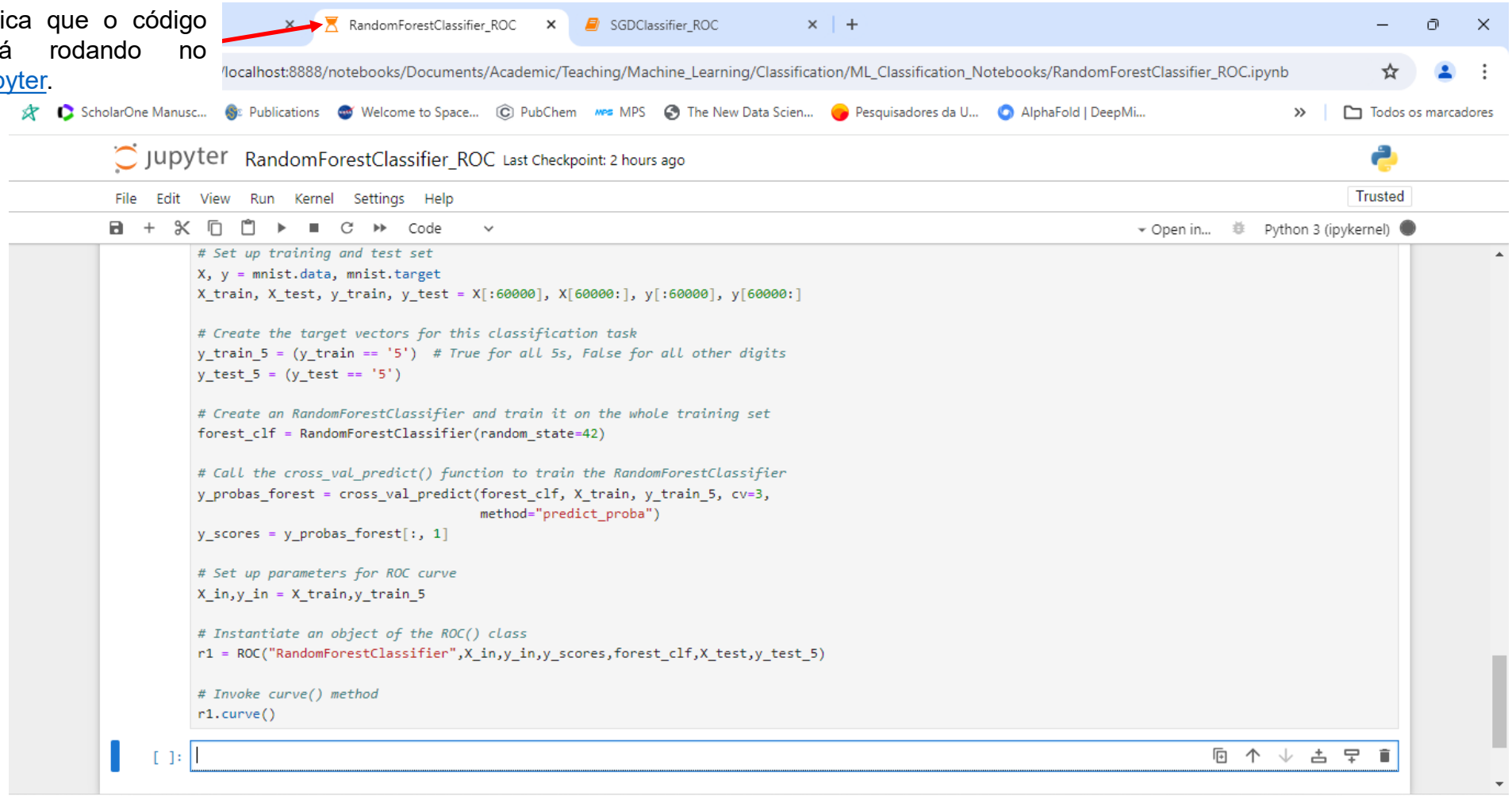
Abaixo temos a janela do Jupyter com o código do programa *RandomForestClassifier_ROC.ipynb*. Seguindo o mesmo procedimento, clique em qualquer parte do código. Depois clique no botão para executar o código.



# Classificador Floresta Aleatória

Como previamente destacado, a execução do código pode demorar alguns minutos.

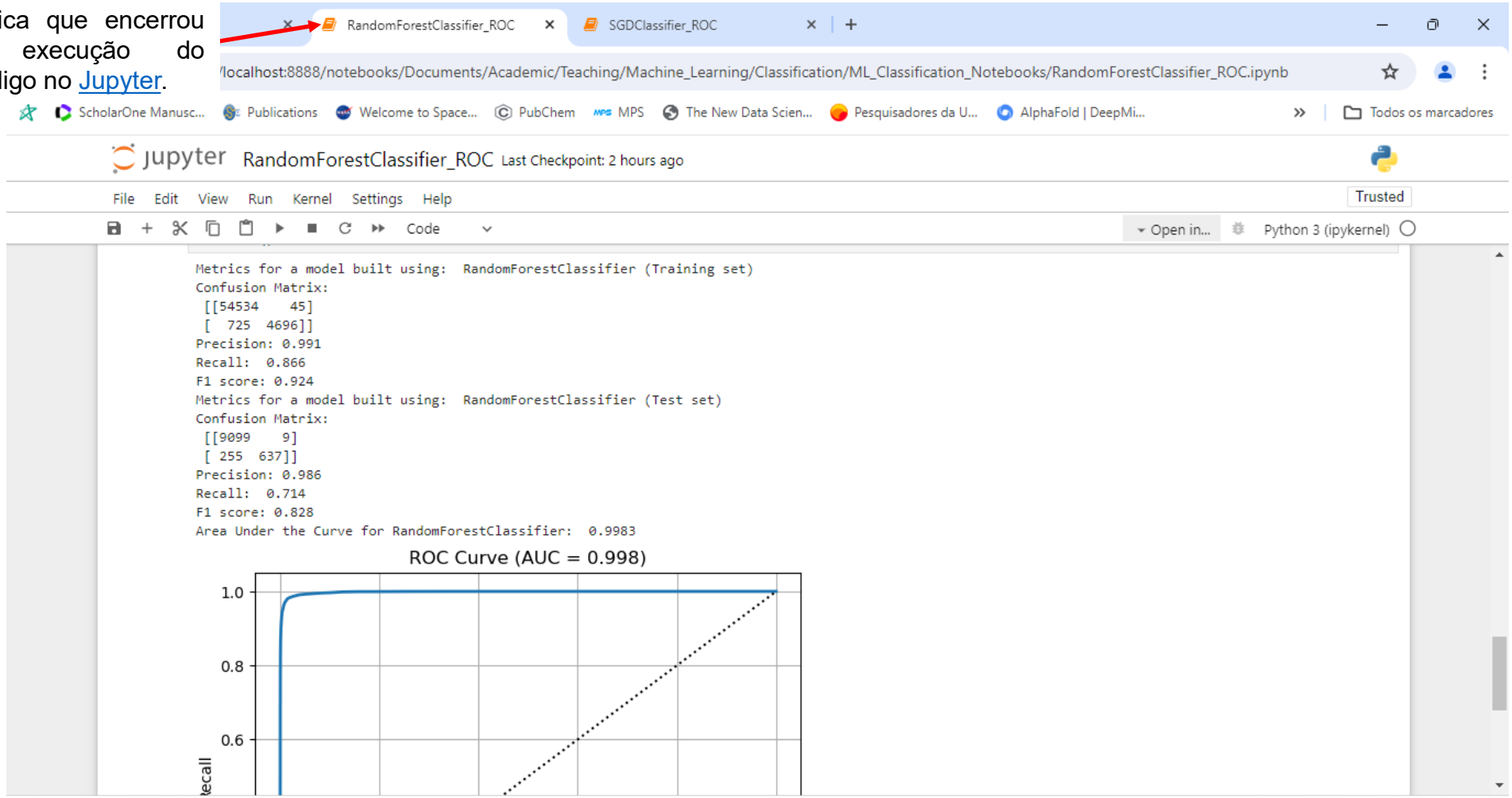
Indica que o código está rodando no Jupyter.



# Classificador Floresta Aleatória

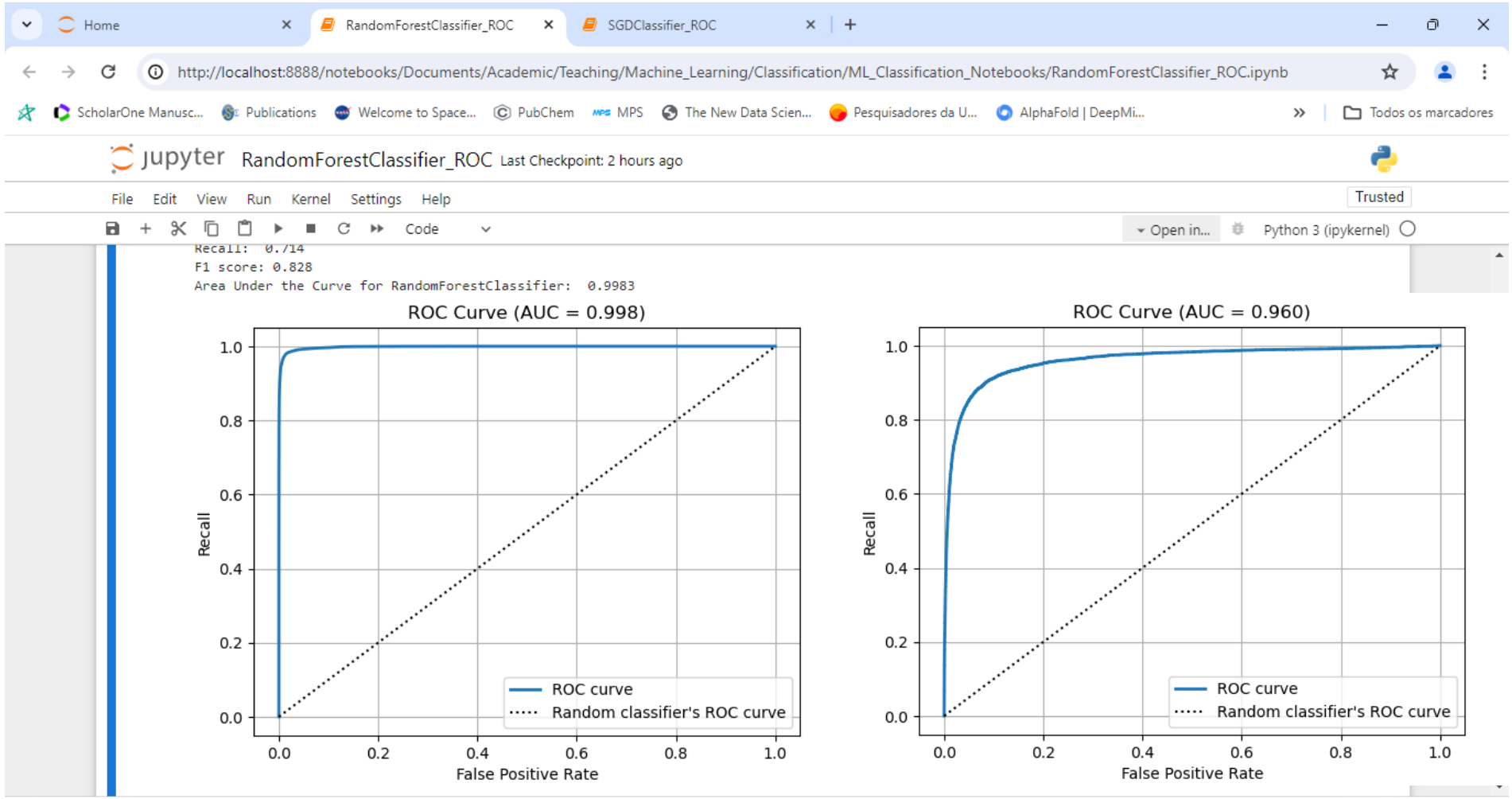
Como no classificador anterior, temos as métricas e a curva ROC. Olhando a AUC, vemos que o modelo gerado pelo classificador floresta aleatória (AUC = 0,998) é superior ao anterior (AUC = 0,960).

Indica que encerrou a execução do código no Jupyter.



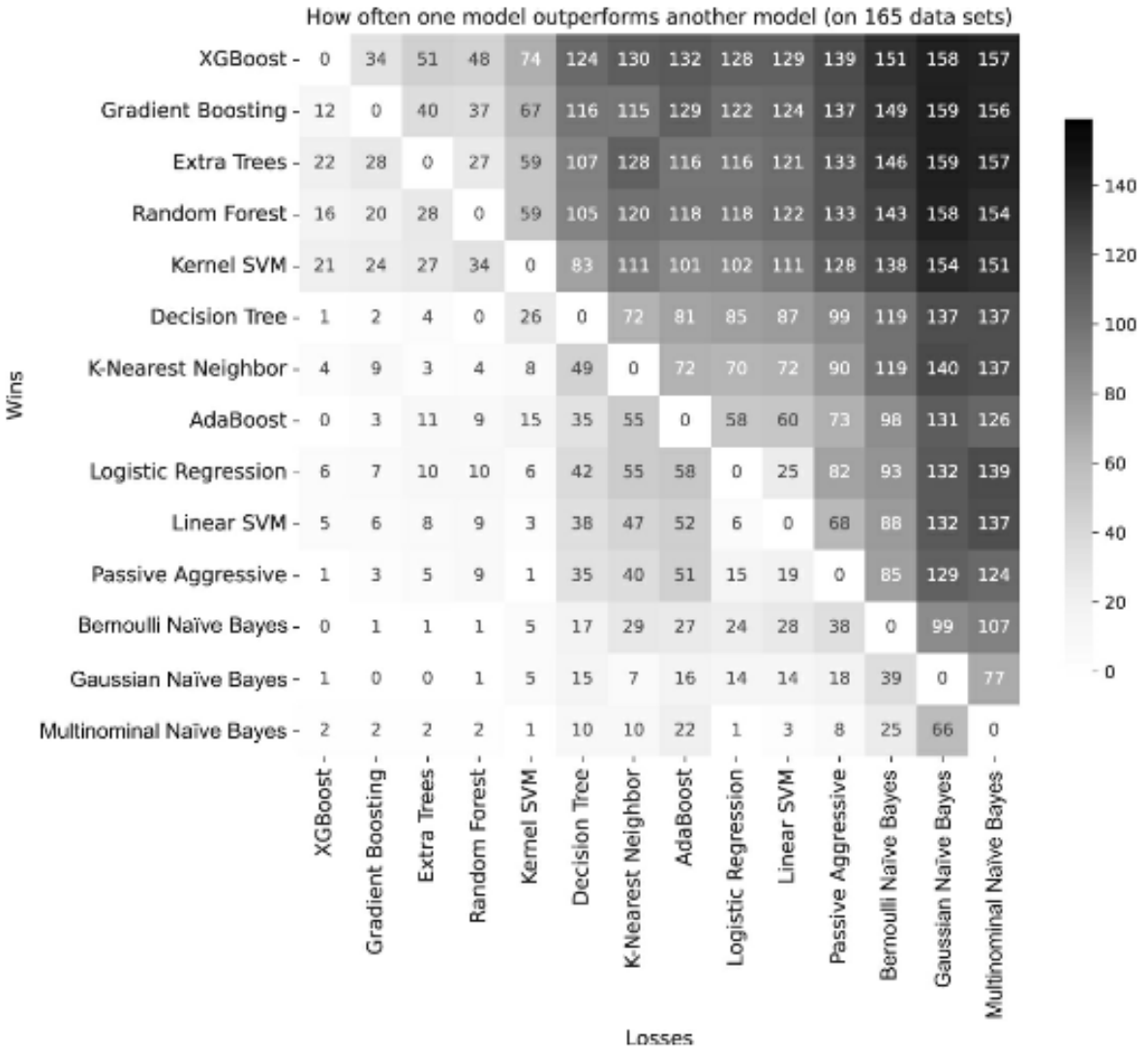
## Classificador Floresta Aleatória

Abaixo temos as curvas ROC para os dois métodos. Vemos claramente o melhor desempenho da curva da esquerda.



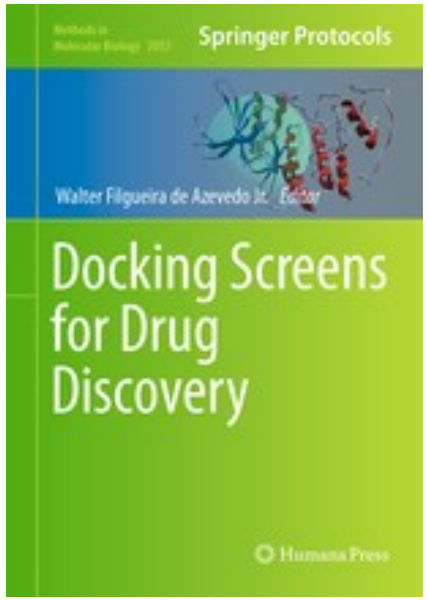
## Métodos Ensemble

De uma forma geral, tanto para problemas de classificação quanto para aqueles de regressão podemos considerar os métodos ensemble com grande chance de apresentarem desempenho superior.
Quando for considerar a elaboração de um modelo de aprendizado de máquina, sempre inclua os métodos ensemble devido ao desempenho e aplicabilidade a grandes conjuntos de dados. Além disso, bibliotecas como o Scikit-Learn apresentam a maioria dos métodos ensemble.




Fonte: Kunapuli, Gautam. Ensemble Methods for Machine Learning (p. 36). Manning. Edição do Kindle.

# Autor




Dr. Walter F. de Azevedo, Jr. earned a BSc in Physics (1990), an MSc in Applied Physics (1992), and a DSc in Applied Physics (1997) from the University of São Paulo (Brazil). In his doctoral studies, Dr. Azevedo worked under the supervision of Prof. Yvonne Primerano Mascarenhas (University of São Paulo) and Prof. Sung-Hou Kim (University of California, Berkeley) on a split Doctoral program with a fellowship from the Brazilian Research Council (CNPq). During his first two years at Berkeley, he was under a CNPq fellowship (1993-95). Due to his performance, Prof. S.-H. Kim hired him as Visiting Researcher for the Department of Chemistry, University of California at Berkeley (1995-96).

The work developed during these three years at Berkeley resulted in his thesis about the structure of Cyclin-Dependent Kinase 2 (CDK2) in complex with inhibitors (PDB access code: 2A4L) (de Azevedo et al., 1996; de Azevedo et al., 1997). Dr. Azevedo is the first author of both papers, and these publications gathered more than 1,000 citations on the Web of Science. During 1997-98 he had a postdoc position at São Paulo State University (Unesp) with a Fapesp fellowship. He holds a habilitation degree in Physics (livre-docência) from the São Paulo State University (Unesp)(2004). In 1998, Dr. Azevedo participated in a research project with NASA that sent proteins to crystallize in a microgravity environment onboard the Space Shuttle Discovery (STS-95). This research had coverage of Brazilian TV networks. He published a book entitled "Docking Screens for Drug Discovery" with Springer Nature in 2019. This book sold 46,000 copies (April 2024) with over 2 million dollars in sales (https://link.springer.com/book/10.1007/978-1-4939-9752-7). In 2020, the Journal Plos Biology ranked Dr. Azevedo among the most influential researchers in the world (Fields: Biochemistry & Molecular Biology and Biophysics).

Dr. Azevedo has vast editorial experience. He is the frontiers section editor (Bioinformatics/Biophysics) for the Current Drug Targets, section editor (Bioinformatics in Drug Design and Discovery) for the Current Medicinal Chemistry, review editor for Frontiers in Chemistry, associate editor for Exploration of Drug Science, member of the editorial boards Molecular Diversity and the Journal of Molecular Structures, and editor of Docking Screens for Drug Discovery (Methods of Molecular Biology)-Springer Nature. He is a reviewer for over 60 high-impact journals, including Nature Communications and Briefings in Bioinformatics. His research interests are interdisciplinary, with three main emphases: machine learning, complex systems, and computational systems biology. Dr. Azevedo has over 200 scientific publications about protein structures, computer models of complex systems, and simulations of protein systems. These workers have over 7300 citations on the Web of Science (h-index: 48. m-quotient: 1.7), +7800 citations in Scopus (h-index: 50), and +9700 citations on Google Scholar (h-index: 53).

# Referências

Géron, Aurélien. 2023. Hands-on Machine Learning with Scikit-Learn, Keras, and# TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. 3rd ed. CA 95472: O'Reilly.

Kunapuli, Gautam. 2023. Ensemble Methods for Machine Learning. Manning. Edição do Kindle.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Verplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. Scikitlearn: Machine Learning in Python. J. Mach. Learn. Res., 2011, 12, 2825-2830.

Walsh I, Fishman D, Garcia-Gasulla D, Titma T, Pollastri G; ELIXIR Machine Learning Focus Group, Harrow J, Psomopoulos FE, Tosatto SCE. DOME: recommendations for supervised machine learning validation in biology. Nat Methods. 2021;18(10):1122-1127.


Que a luz da ciência acabe com
as trevas do negacionismo.