Google Colab + IA

Instalação e uso de scikit-learn e TensorFlow

Ferramentas

Notebook + ML + Deep Learning

TensorFlow

Roteiro da aula

Ao final, você consegue:

- Criar/organizar um notebook no Colab
- Selecionar CPU, GPU ou TPU e verificar o hardware
- Usar IA para explicar, gerar e revisar código com cuidado
- Instalar e importar scikit-learn e TensorFlow
- Treinar 2 mini-modelos (ML clássico + deep learning)

Sequência

1) Interface e atalhos do notebook

2) Runtime: CPU/GPU/TPU + !nvidia-smi

3) IA no Colab: perguntar, pedir exemplos, revisar

4) Instalar pacotes (pip) e checar versões

5) scikit-learn: pipeline + métricas

6) TensorFlow: Keras + treino rápido

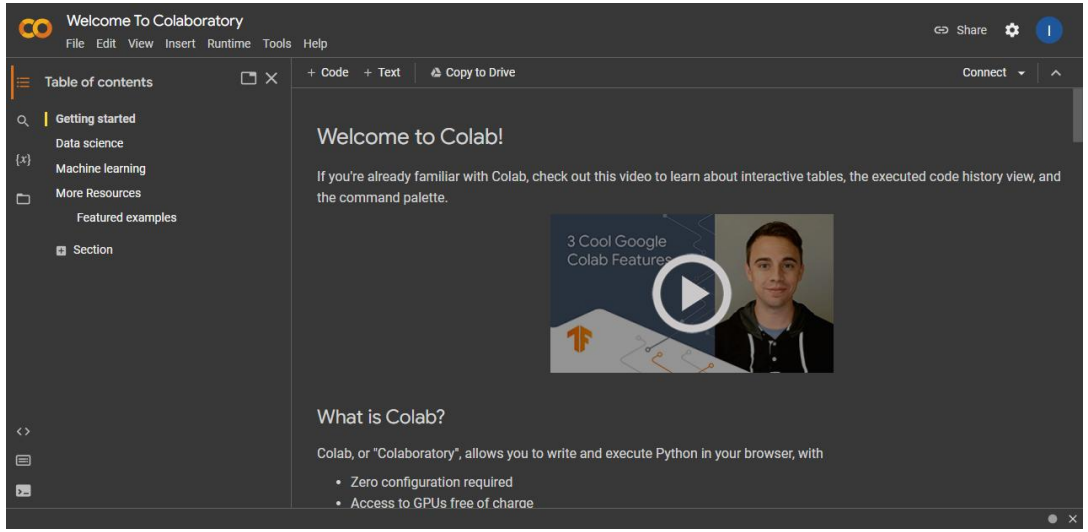
O que é o Google Colab?

Ideia central

Um ambiente estilo Jupyter Notebook no navegador, com Python pronto para rodar e com integração ao Google Drive.

- Sem instalação local
- Compartilhamento fácil (link)
- Acesso a GPU/TPU (quando disponível)
- Ótimo para aulas, protótipos e experimentos

Interface (exemplo)



Notebook por dentro: células e execução

Dois tipos de célula

Texto (Markdown)

Use para explicar o raciocínio, inserir imagens, links e fórmulas.

Código (Python)

```
1 x = 2
2 print(x**3)
```

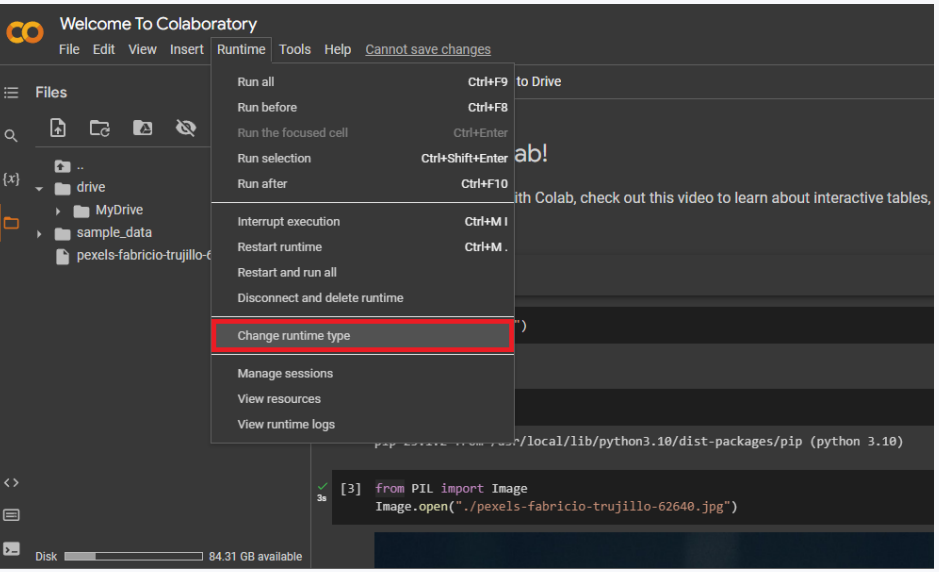
Atalhos úteis

- Shift + Enter: roda a célula e vai para a próxima
- Ctrl/Cmd + Enter: roda e fica na mesma célula
- Ctrl/Cmd + M B: nova célula abaixo
- Ctrl/Cmd + M M: transforma em Markdown
- Ctrl/Cmd + M Y: transforma em código
- !comando: roda shell (ex.: !pip, !ls)

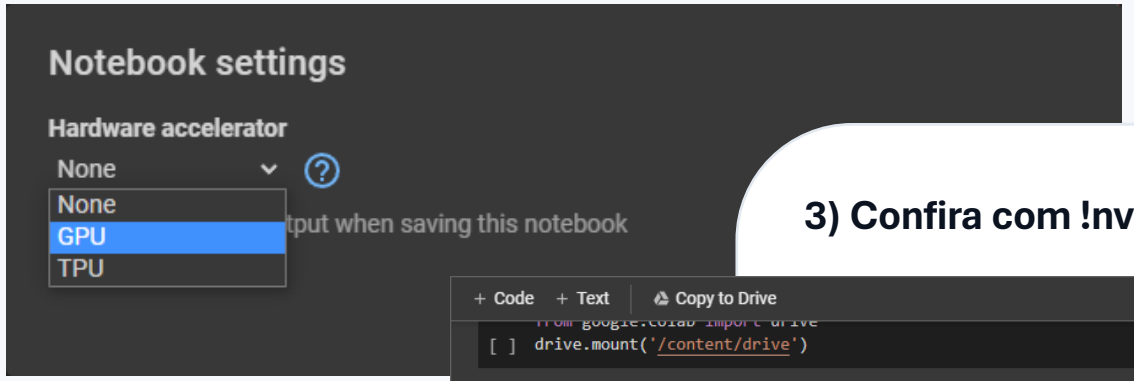
Runtime: CPU, GPU e TPU

Caminho rápido: Runtime → Change runtime type → Hardware accelerator

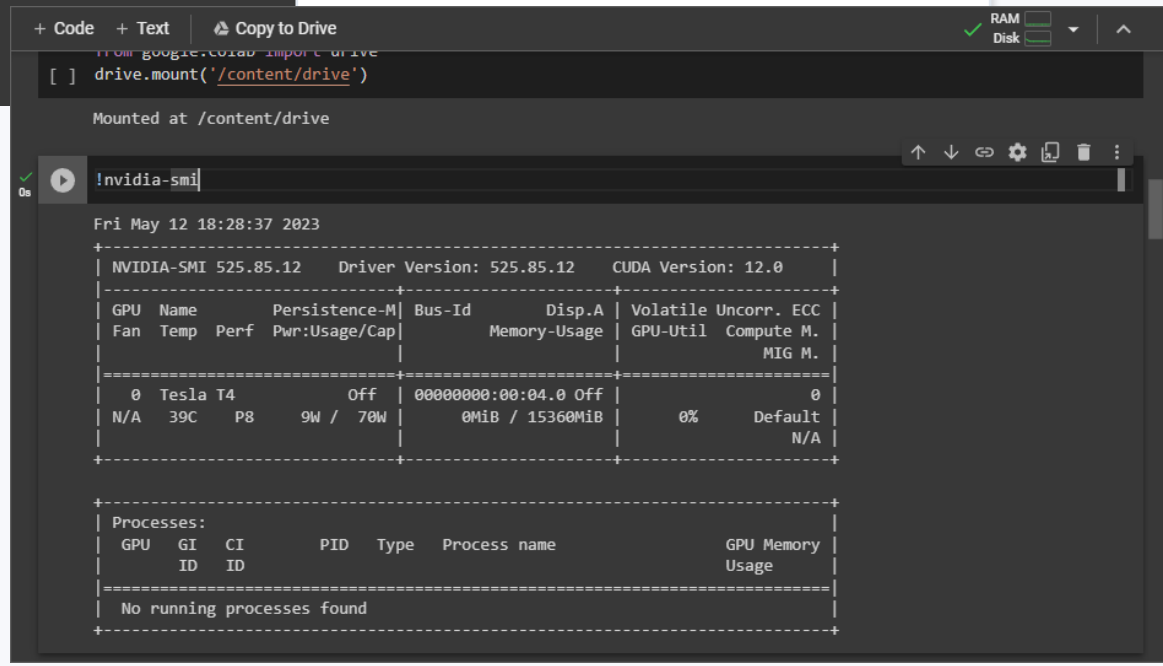
1) Abra o menu Runtime



2) Selecione GPU (ou TPU)



3) Confira com !nvidia-smi



Dica: GPU/TPU no plano gratuito pode ser limitado por tempo/uso. Ative só quando precisar.

Regra de ouro

O disco do runtime é temporário. Para não perder arquivos, use o Google Drive (ou baixe ao final).

Montar o Drive:

```
from google.colab import drive
drive.mount('/content/drive')

# depois: /content/drive/MyDrive/
```

Boas práticas

- Crie uma pasta por projeto no Drive
- Guarde datasets em /MyDrive/dados/
- Salve modelos em /MyDrive/modelos/
- Use nomes e seções (Markdown) para legibilidade
- Faça “Restart runtime” se o ambiente ficar estranho

Extra (quando útil):

- Carregar notebook do GitHub
- Baixar arquivos do runtime (Files → Download)

O que a IA ajuda a fazer ?

- Explicar código e erros (“por que isso quebrou?”)
- Gerar esboços de funções e pipelines
- Sugerir melhorias (performance, legibilidade, métricas)
- Resumir o notebook e criar comentários/Markdown

Como pedir (modelo de prompt)

```
1 Contexto: tenho um notebook de classificação
2 Tarefa: crie um pipeline scikit-learn com
validação
3 Restrições: sem vazamento de dados, com métricas
4 Saída: código + explicação curta + como testar
```

⚠️ Checklist de segurança

- Leia o código gerado antes de executar
- Cuidado com comandos “!” (shell) e downloads
- Peça para justificar escolhas e citar docs
- Se der erro: cole o stack trace e o trecho mínimo
- Trate segredos (tokens/senhas) como “nunca colar”

Instalando bibliotecas (pip) no Colab

O básico

- Algumas libs já vêm instaladas (numpy, pandas, etc.)
- Use !pip para instalar/atualizar
- Se der conflito: Runtime → Restart runtime
- Sempre confira versões antes de começar o projeto

Comandos recomendados:

```
# instalar/atualizar
!pip install -q -U scikit-learn
!pip install -q -U tensorflow

# checar
import sklearn, tensorflow as tf
print('sklearn', sklearn.__version__)
print('tf', tf.__version__)
```

Dicas rápidas

- Use “-q” para reduzir o barulho do pip
- Para pacotes do sistema: !apt-get
- Evite misturar versões muito antigas
- Para reprodutibilidade: anote versões no topo do notebook

Se o Colab já tiver TF/sklearn instalados, você pode pular o pip e só importar. É bom validar versões para evitar surpresas.

Quando usar?

- Problemas tabulares (planilhas, CSV)
- Baselines rápidos e interpretáveis
- Pré-processamento + modelo + métricas em um fluxo só
- Ótimo para começar antes do deep learning

Conceitos que aparecem no código:

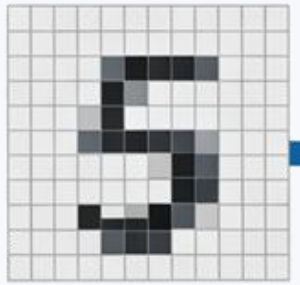
- train/test split
- padronização (scaler)
- modelo
- métrica

Exemplo (Iris + LogisticRegression)

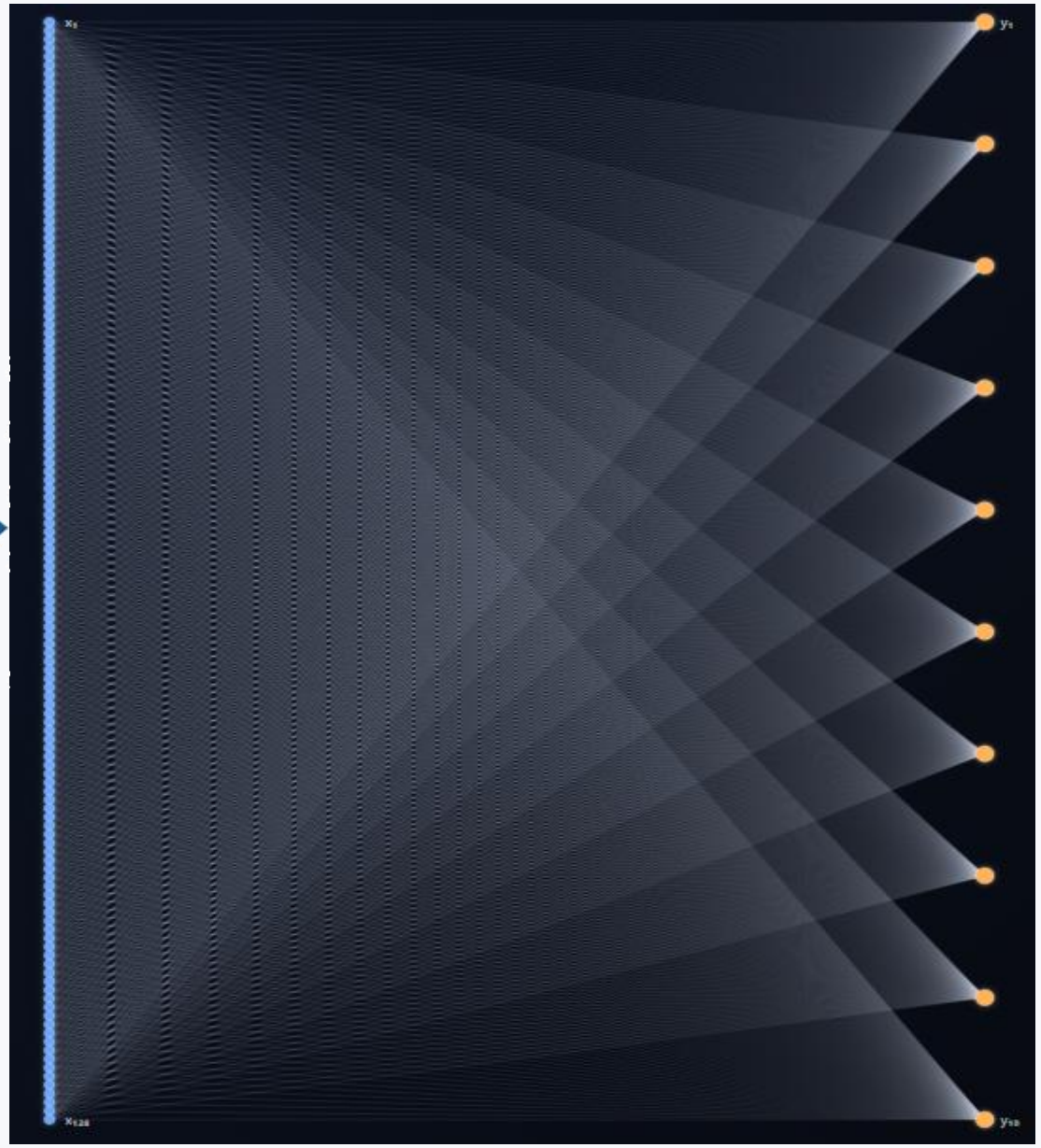
```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

X, y = load_iris(return_X_y=True)
Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

clf = Pipeline([
  ('scaler', StandardScaler()),
  ('model', LogisticRegression(max_iter=1000))
])
clf.fit(Xtr, ytr)
pred = clf.predict(Xte)
print('acc:', accuracy_score(yte, pred))
```

TensorFlow: rede neural com Keras

128 neurônios
de entrada



28×28 Pixels



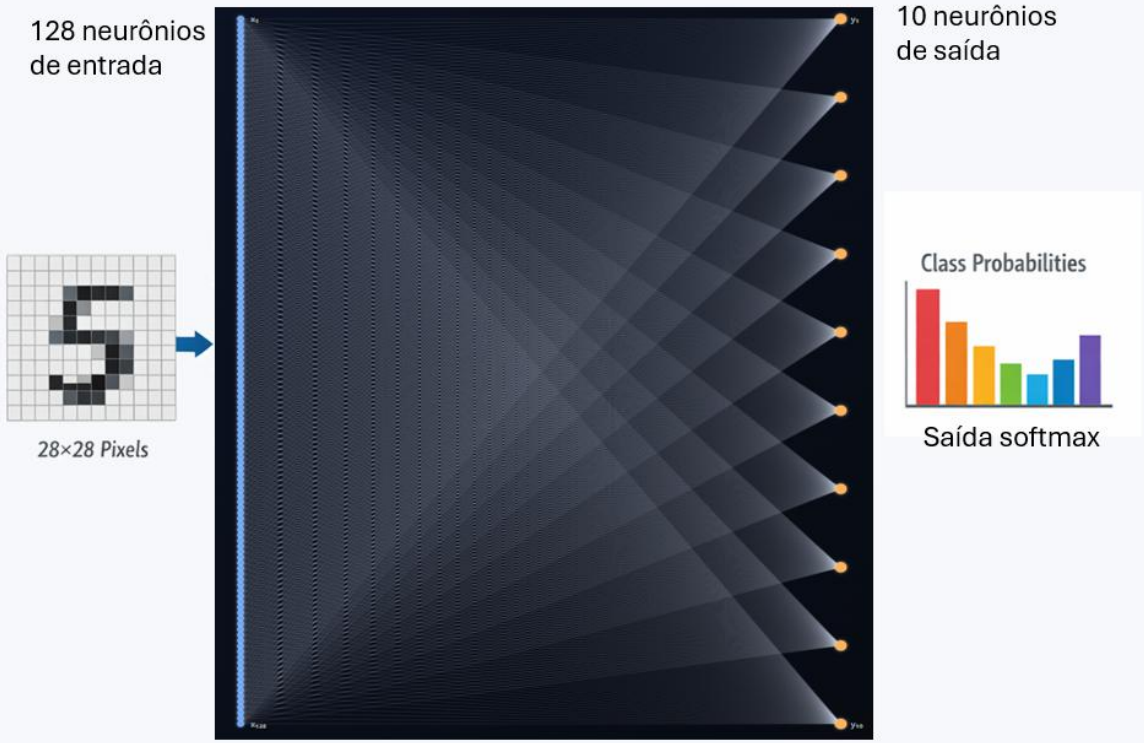
10 neurônios
de saída

Saída softmax

TensorFlow: rede neural com Keras

Exemplo (Keras)

```python
import tensorflow as tf

(xtr, ytr), (xte, yte) = tf.keras.datasets.mnist.load_data()
xtr = xtr.astype("float32") / 255.0
xte = xte.astype("float32") / 255.0

model = tf.keras.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation="relu"),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation="softmax"),
])
model.compile(optimizer="adam",
              loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])
model.fit(xtr, ytr, epochs=3, batch_size=128, validation_split=0.1)
print(model.evaluate(xte, yte, verbose=0))
```



TensorFlow: rede neural com Keras

Quando faz sentido?

- Imagens, texto, áudio e séries temporais
- Modelos com muitas camadas e não-linearidades
- Aproveitar GPU para acelerar treino
- Deploy/export com ferramentas do ecossistema TF

Dica de performance

Se você ativou GPU, confirme com: `tf.config.list_physical_devices("GPU")`

Exemplo (Keras)

```
import tensorflow as tf

(xtr, ytr), (xte, yte) = tf.keras.datasets.mnist.load_data()
xtr = xtr.astype("float32") / 255.0
xte = xte.astype("float32") / 255.0

model = tf.keras.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation="relu"),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation="softmax"),
])
model.compile(optimizer="adam",
              loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])
model.fit(xtr, ytr, epochs=3, batch_size=128, validation_split=0.1)
print(model.evaluate(xte, yte, verbose=0))
```

Salvar notebook e modelos

Salvar no Drive

- O Colab salva notebooks no Drive (pasta “Colab Notebooks”)
- Modelos e arquivos gerados devem ir para /content/drive/MyDrive/
- Você também pode baixar arquivos pelo painel Files

Exemplo: scikit-learn (joblib)

```
import joblib
joblib.dump(clf, '/content/drive/MyDrive/modelos/iris_clf.joblib')
```

Exemplo: TensorFlow (SavedModel)

```
model.save('/content/drive/MyDrive/modelos/mnist_model')
```

Checklist final do notebook

- Topo: objetivo + versões (Python, TF, sklearn)
- Seções: dados → treino → avaliação → próximos passos
- Resultados salvos (métricas, gráficos, modelos)
- Runtime desligado (se não for usar)

Dica: peça para a IA criar um “Resumo do Notebook” no final (em Markdown) com: dados, modelo, métrica, e como reproduzir.

Se algo der errado, normalmente é uma destas 6 coisas:

GPU não aparece	Runtime → Change runtime type; tente reiniciar; pode haver limitação de cota.
“ModuleNotFoundError”	Instale com !pip (ou reinicie runtime após instalar).
Conflito de versões	Evite misturar versões antigas; reinicie e reinstale “limpo”.
Notebook ficou lento	Limpe variáveis grandes; reinicie runtime; use batch menor.
Sem espaço/disco	Apague arquivos temporários; mova dados para o Drive.
Erro misterioso	Crie um “mínimo reproduzível” e peça para a IA explicar o stack trace.

Desafio A: scikit-learn

- Trocando LogisticRegression por RandomForest
- Comparando accuracy
- Peça para a IA sugerir 2 melhorias e explique o porquê

Prompt pronto

“Adapte meu código para usar RandomForest + validação cruzada.
Explique as escolhas e mostre como evitar vazamento de dados.”

Desafio B: TensorFlow

- Aumente epochs para 5 e observe overfitting
- Teste outra arquitetura (2 Dense + Dropout)
- Mostre um gráfico de loss/accuracy
- Peça para a IA sugerir um “próximo experimento”

Pergunta de reflexão

Qual foi a maior fonte de erro: dados, modelo, ou avaliação?
O que você faria para validar melhor?

Documentação (oficial / recomendada)

Colab: https://colab.research.google.com/
Gemini no Colab (Enterprise): https://docs.cloud.google.com/colab/docs/gemini-in-colab/overview
scikit-learn (instalação): https://scikit-learn.org/stable/install.html
TensorFlow (instalação): https://www.tensorflow.org/install
Tutorial Colab (interface): https://christianjmills.com/posts/google-colab-getting-started-tutorial/