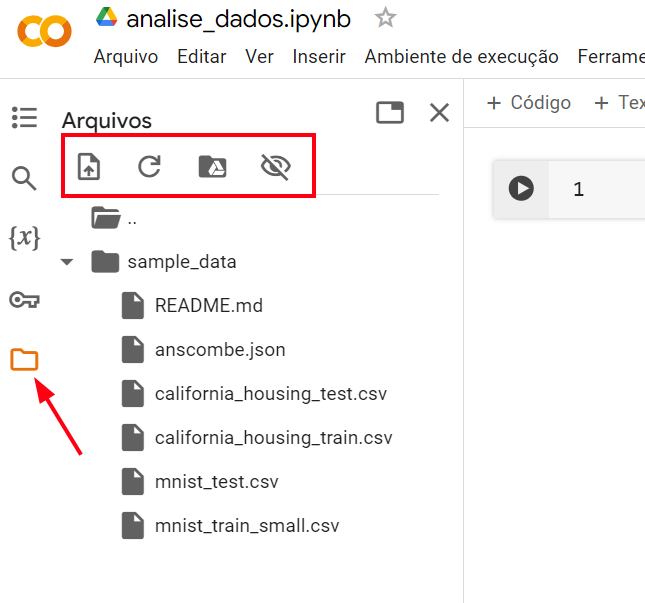
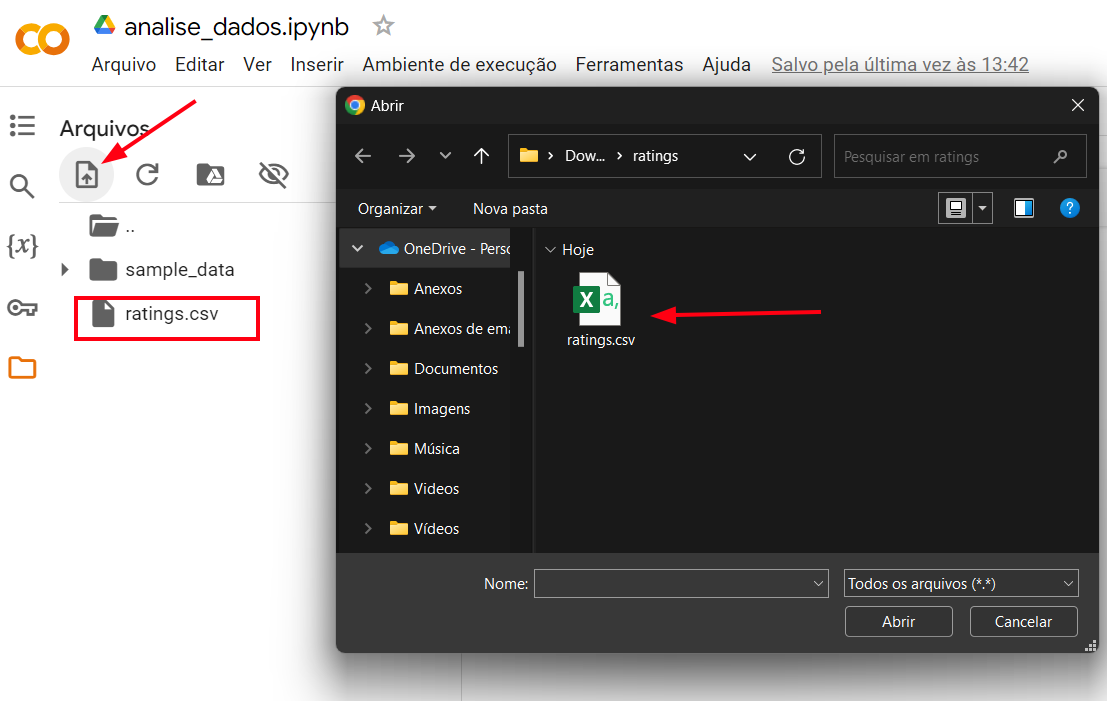
**Para saber mais: fazendo upload de arquivos no Colab**

O Google Collaboratory é uma ferramenta completa para análise de dados, fornecendo recursos em nuvem de forma gratuita com suporte a linguagem Python e várias bibliotecas instaladas. Naturalmente, para fazer a análise de dados, é preciso utilizar uma base de dados, e o Colab oferece um sistema de armazenamento de arquivos que permite carregar os dados para dentro do ambiente de desenvolvimento.

O sistema de arquivos pode ser acessado na 5ª opção do menu lateral esquerdo, a partir de um ícone de pasta. A partir da seleção, surgirá um explorador de arquivos, com uma pasta sample\_data contendo arquivos de dados para praticar e um novo menu de 4 ícones. O primeiro destes ícones é o de fazer upload de arquivos, o segundo atualiza o sistema de arquivos, o terceiro permite conectar o sistema de arquivos ao Google Drive a partir da conta do gmail e por fim, o quarto ícone habilita a visualização de arquivos ocultos.

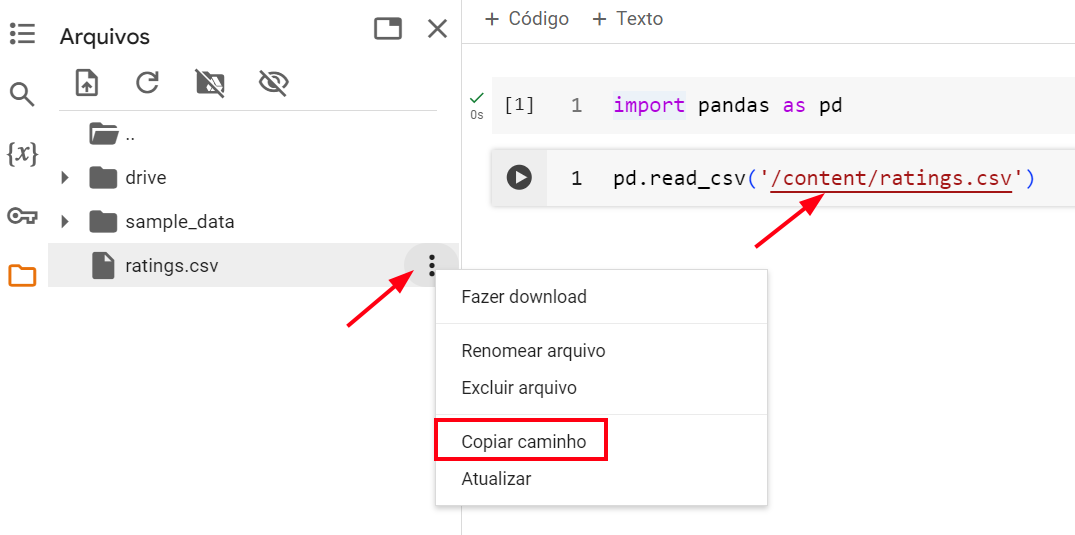


A partir daí, temos duas opções para fazer a leitura de arquivos, seja fazendo o upload diretamente do computador ou a partir de um arquivo que esteja no Google Drive. A primeira opção é bem direta, vamos selecionar o ícone de upload do arquivo e o explorador de arquivos do sistema operacional irá se abrir, podemos selecionar o arquivo da nossa escolha e enviar para dentro do ambiente do Colab. O arquivo surgirá dentro do explorador de arquivos do Colab. Caso não apareça, podemos selecionar no botão de atualizar.



Se for o desejo acessar o arquivo pelo Google Drive, podemos fazer o upload do arquivo para dentro do Google Drive, da forma usual e depois selecionamos a 3ª opção do menu de arquivos do Colab, chamada "Montar Drive", que permite fazer um acesso direto aos arquivos da conta de gmail que está acessando o Colab. Ao acessar essa opção, o Google solicitará uma permissão e caso seja permitido, o explorador de arquivos apresentará uma nova pasta, chamada "drive", contendo todos os arquivos do Google Drive.

Depois de fazer o upload do arquivo em formato csv, precisamos fazer a leitura com a biblioteca pandas. Para isso, podemos usar o método read\_csv(), informando o local do arquivo dentro do sistema de arquivos do Colab. Podemos navegar pelo sistema de pastas de arquivos, seja no arquivo de upload ou na pasta do Drive, e ao lado direito do arquivo selecionar o ícone de mais opções, escolhendo a opção "Copiar caminho". Feito isso, basta colar o caminho do arquivo entre os parênteses da função read\_csv().



**Para saber mais: Pandas Series e DataFrame**

Series e DataFrame são estruturas de dados fundamentais no ecossistema do Pandas. Nós cientistas de dados nos debruçamos diariamente com essas estruturas e, por isso, é importante aprender sobre o que representam e como manipulá-las.

**DataFrame**

Um DataFrame é uma estrutura tabular bidimensional, similar a uma planilha ou tabela de banco de dados, onde os dados são organizados em linhas e colunas. Vamos criar um simples DataFrame e verificar sua estrutura:

**import** pandas **as** pd

# Criando um DataFrame simples

dados = { 'Nome': ['Mirla', 'Ana', 'Daniel'],

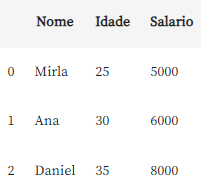
'Idade': [25, 30, 35],

'Salario': [5000, 6000, 8000]}

df = pd.DataFrame(dados)

df

**Saída:**



Explorando brevemente a sua estrutura, no lado esquerdo temos o índice (de 0 a 2) indicando a quantidade de registros (observações) que possuímos no DataFrame, em negrito no topo as colunas, que identificam a natureza dos dados e os valores que são a combinação entre o tipo do dado e a observação que ele representa, por exemplo, Mirla é um nome no índice de número 0.

Em um DataFrame, podemos aplicar funções para explorar, tratar e manipular os dados tanto em toda a sua estrutura como em uma ou mais colunas (Series). Podemos observar, logo abaixo, dois exemplos de como explorar um DataFrame por inteiro e por apenas uma coluna:

df.info()

**Saída:**

<**class** 'pandas.core.frame.DataFrame'>

RangeIndex: 3 entries, 0 to 2

Data columns (total 3 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Nome 3 non-null object

1 Idade 3 non-null int64

2 Salário 3 non-null int64

dtypes: int64(2), object(1)

memory usage: 200.0+ bytes

Aqui, é possível notar algumas observações sobre o DataFrame, como quantidade de registros e colunas, o tipo das variáveis da coluna (object e int64), se possuem dados nulos e o uso de memória daqueles dados.

df['Salario'].mean()

# ou df.Salario.mean()

**Saída:**

6333.333333333333

No código acima, calculamos apenas a média de uma coluna do df trazendo a média salarial das pessoas cadastradas.

### Series

Um Series por sua vez é uma estrutura unidimensional que armazena uma sequência de valores indexados. Podemos dizer que um DataFrame nada mais é que uma sequência de vários Series nomeados organizados coluna por coluna e ordenados pelos índices.

**import** pandas **as** pd

# Criando um Series simples

idades = pd.Series([45, 30, 45, 40, 25])

idades

**Saída:**

0 45

1 30

2 45

3 40

4 25

dtype: int64

Explorando brevemente a sua estrutura, no lado esquerdo temos o índice (de 0 a 4) indicando a quantidade de registros (observações) que possuímos no Series e no lado direito os registros dos dados. Na última linha, podemos observar o tipo do dado (int64). A diferença entre este Series e o Series extraído de uma coluna do DataFrame é que o que vem do DataFrame também apresenta na última linha o nome (Name) do Series.

Num Series, podemos aplicar funções para explorar, tratar e manipular seus dados, como podemos observar, logo abaixo, em dois exemplos:

idades.index

**Saída:**

RangeIndex(start=0, stop=5, step=1)

No código acima lemos os índices do Series. Como estão numerados de 0 até 4, o código representa que o índice é gerado pelo RangeIndex()partindo de 0 até 5 de um a um.

idades.sort\_values()

**Saída:**

4 25

1 30

3 40

0 45

2 45

dtype: int64

No código acima, ordenamos apenas os valores de idades em ordem crescente, sem alterar os índices.

Observamos nestes exemplos, como podemos trabalhar com conjuntos de dados estruturados, seja em formato de DataFrame ou Series, compreendendo suas diferenças e como utilizá-las no contexto de ciência de dados.