Mundo Tech

ANÁLISE DE DADOS EM PYTHON COM PANDAS

SENAI <LAB365>

SUMÁRIO

Análise de dados em Python com Pandas **3**

Dicas do Google Colab **3**

Preparando o Ambiente **4**

Leitura de arquivos de dados **4**

Primeiros contatos com os dados **5**

Obtendo informações de um *Data Frame* 6

Outras formas de selecionar uma pequena amostra de dados 7

Invertendo linhas por colunas 8

Selecionando linhas 9

Slicing 9

Selecionando colunas 10

Resposta do desafio 11

Referências **11**

ANÁLISE DE DADOS EM PYTHON COM PANDAS

O Pandas é uma biblioteca para manipulação e análise de dados, escrita em Python. Essa biblioteca possui um conjunto de ferramentas com funções para nossa análise exploratória dos dados. Com ela, podemos ler, manipular e apresentar os dados sem muita complicação.

O Pandas é uma das bibliotecas mais utilizadas, pois é de fácil uso e aprendizagem.

Neste material, você conhecerá alguns comandos, componentes e sintaxe da Biblioteca Pandas. E, é importante destacar que este documento servirá de guia para o futuro.

DICAS DO GOOGLE COLAB

O Google Colab manipula arquivo do Jupyter Notebook, que também chamamos de arquivo Notebook. Ele é um interpretador de Python interativo. Assim, podemos digitar um comando de Python e apertar *Shift* + *Enter,* e esse comando será executado e seu resultado impresso na tela.

Existem alguns atalhos bastante úteis para trabalhar com Jupyter Notebook:

0. ‘Esc’ - Saí do modo edição da célula atual
1. ‘Enter’ - Edita a célula selecionada
2. ‘Shift + Enter’ - Executa a célula selecionada
3. ‘A’ - Adiciona uma nova célula acima da célula atual (não pode estar em modo de edição)
4. ‘B’ - Adiciona uma nova célula abaixo da célula atual (não pode estar em modo de edição) x
5. ‘Ctrl + Z’ - Desfaz as últimas edições
6. ‘Ctrl + Shift + Z’ - Refaz o último comando de desfazer
7. ‘?nome_função’ - Mostra a documentação de uma dada função

PREPARANDO O AMBIENTE

Para iniciar o uso da Biblioteca Pandas, precisamos, antes de tudo, importar as bibliotecas. Para isso, utilizamos o comando indicado na figura, a seguir. Neste comando, utilizamos a função *import* e chamamos a biblioteca pandas. Ao realizar este comando, criamos um objeto com nome "pd", que assume o comportamento da biblioteca e o utilizaremos em nossos comandos futuros.

```
[ ]  import pandas as pd
```

LEITURA DE ARQUIVOS DE DADOS

Existem muitas funções nativas para ler dados e, da mesma forma, podemos realizar a leitura de diferentes tipos de dados.

Para utilizar um arquivo no formato CSV, utilizamos a função read_csv. Para isso, iremos utilizar a função read_csv do Pandas, com o qual acabamos de criar um objeto chamado "pd". Além disso, precisaremos informar o caminho do arquivo e que tipo de separador ele está utilizando (vírgula ou ponto e vírgula).

Após a leitura desse arquivo, salvamos esses dados em um Data Frame, que chamamos aqui de "df".

Sobre o *encoding*, é importante você saber que ele trata do tipo de código que cada letra e símbolo do meu DataSet está utilizando. Por exemplo, o encoding ISSO-8859-1 suporta palavras com acentos e caracteres característicos da língua portuguesa.

```
df = pd.read_csv('caminho_arquivo.csv', encoding = 'ISO-8859-1', delimiter='caractere_delimitador')
df = pd.read_csv('Dataset_ENEM.csv', encoding = 'ISO-8859-1', delimiter=',')
```

Para a leitura de um arquivo do Excel, utilizamos a mesma lógica, porém, neste caso, com a função read_excel. Além disso, devemos aqui informar qual das planilhas do arquivo Excel desejamos carregar.

```
df = pd.read_excel('caminho_arquivo.xlsx', sheet_name='guia')
df = pd.read_excel('vendas_202005.xlsx', sheet_name='Jan')
```

PRIMEIROS CONTATOS COM OS DADOS

A primeira tarefa a ser feita quando carregamos um conjunto de dados é uma rápida visualização a fim de entender melhor o que consta nele. Assim, basta simplesmente informar o nome do Data Frame no qual acabamos de carregar o arquivo, neste caso o “df”, e executar o código.

A partir daí, o Pandas irá nos retornar os dados. Neste caso, podemos observar que eles estão em formato de uma tabela, em que cada linha representa um aluno e, nas colunas, estão os dados, conforme podem ser detalhados no arquivo de Dicionário de Dados.

Podemos observar que a tabela de dados do ENEM possui 909.194 linhas e 18 colunas.

```
df
```

	NOTA_CN	NOTA_CH	NOTA_LC	NOTA_MT	NOTA_REDACAO	NOTA_MEDIA	TP_ESCOLA	IDADE	SEXO	ESTADO_CIVIL	COR_RACA	TP_ENSINO
0	574.2	538.7	582.6	571.5	640.0	566.750	Privada	18	F	1	1	1.0
1	369.7	398.2	395.6	473.8	480.0	409.325	Pública	26	F	0	3	1.0
2	423.0	372.8	488.7	516.5	540.0	450.250	Pública	17	F	1	3	1.0
3	412.5	417.2	485.1	446.2	580.0	440.250	Pública	18	F	1	3	1.0
4	460.7	570.5	541.8	631.3	640.0	551.075	Pública	17	M	1	3	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...
909189	533.3	485.8	530.2	595.5	540.0	536.200	Pública	17	F	1	1	1.0
909190	486.2	543.4	560.3	535.2	740.0	531.275	Pública	18	F	1	1	1.0
909191	500.8	563.2	555.3	442.8	780.0	515.525	Pública	18	F	1	2	1.0
909192	491.6	549.3	586.8	633.5	640.0	565.300	Pública	18	F	1	3	1.0
909193	425.7	525.0	511.5	621.8	620.0	521.000	Pública	17	F	1	3	1.0

909194 rows × 18 columns

Para vermos apenas uma pequena parte dos dados, temos o comando .head(), que irá mostrar por padrão as 5 primeiras linhas do que existe em um conjunto de dados dentro de um objeto do Pandas.

```
df.head()
```

	NOTA_CN	NOTA_CH	NOTA_LC	NOTA_MT	NOTA_REDACAO
0	574.2	538.7	582.6	571.5	640.0
1	369.7	398.2	395.6	473.8	480.0
2	423.0	372.8	488.7	516.5	540.0
3	412.5	417.2	485.1	446.2	580.0
4	460.7	570.5	541.8	631.3	640.0

Da mesma forma, podemos também sinalizar quantas linhas queremos que o comando .head() retorne. Por exemplo, a seguir, solicitamos 10 linhas.

```
df.head(10)
```

	NOTA_CN	NOTA_CH	NOTA_LC	NOTA_MT	NOTA_REDACAO	NOTA_MEDIA
0	574.2	538.7	582.6	571.5	640.0	566.750
1	369.7	398.2	395.6	473.8	480.0	409.325
2	423.0	372.8	488.7	516.5	540.0	450.250
3	412.5	417.2	485.1	446.2	580.0	440.250
4	460.7	570.5	541.8	631.3	640.0	551.075
5	509.8	494.9	540.5	564.3	940.0	527.375
6	422.1	534.4	516.3	585.2	480.0	514.500
7	450.9	436.6	489.1	408.4	580.0	446.250
8	501.3	534.3	539.7	545.1	740.0	530.100
9	407.4	364.8	487.2	377.9	560.0	409.325

Obtendo informações de um *Data Frame*

O comando **.shape** nos apresenta a quantidade de linhas e colunas de um *Data Frame*. Veja o exemplo a seguir.

```
df.shape
```

```
(909194, 18)
```

Podemos também obter os tipos de cada coluna do *Data Frame* através do comando .dtypes.

```
df.dtypes
```

```
NOTA_CN          float64
NOTA_CH          float64
NOTA_LC          float64
NOTA_MT          float64
NOTA_REDACAO     float64
NOTA_MEDIA       float64
TP_ESCOLA         object
```

Já o comando **.info()** apresenta mais detalhes de informações, como apresentado no exemplo, a seguir.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 909194 entries, 0 to 909193
Data columns (total 18 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   NOTA_CN             909194 non-null  float64
 1   NOTA_CH             909194 non-null  float64
 2   NOTA_LC             909194 non-null  float64
 3   NOTA_MT             909194 non-null  float64
 4   NOTA_REDACAO        909194 non-null  float64
 5   NOTA_MEDIA          909194 non-null  float64
 6   TP_ESCOLA           909194 non-null  object
 7   IDADE               909194 non-null  int64
 8   SEXO                909194 non-null  object
 9   ESTADO_CIVIL        909194 non-null  int64
 10  COR_RACA            909194 non-null  int64
 11  TP_ENSINO           909194 non-null  float64
 12  TREINEIRO           909194 non-null  int64
 13  UF_ESC              909194 non-null  object
 14  ESTUDO_PAI          909194 non-null  object
 15  ESTUDO_MAE          909194 non-null  object
 16  PESSOAS_RESIDENCIA  909194 non-null  int64
 17  RENDA_MENSAL        909194 non-null  object
dtypes: float64(7), int64(5), object(6)
memory usage: 124.9+ MB
```

a) Destacado em Verde: número de linhas não nula

b) Azul: tipo de dado de cada coluna, normalmente quando ele identifica como *object* variáveis do tipo *string* (ou texto).

c) Vermelho: memória utilizada por este *Data Frame*.

Além de outras informações, explore o conjunto de dados.

O comando **.columns** nos retorna a lista de todas as colunas existentes no *DataSet*.

```
df.columns
```

```
Index(['NOTA_CN', 'NOTA_CH', 'NOTA_LC', 'NOTA_MT', 'NOTA_REDACAO',
       'NOTA_MEDIA', 'TP_ESCOLA', 'IDADE', 'SEXO', 'ESTADO_CIVIL', 'COR_RACA',
       'TP_ENSINO', 'TREINEIRO', 'UF_ESC', 'ESTUDO_PAI', 'ESTUDO_MAE',
       'PESSOAS_RESIDENCIA', 'RENDA_MENSAL'],
      dtype='object')
```

Outras formas de selecionar uma pequena amostra de dados

O comando inverso ao **head()** é o **tail(),** que irá mostrar as 5 últimas linhas do *Data Frame* por padrão.

```
df.tail()
```

	NOTA_CN	NOTA_CH	NOTA_LC	NOTA_MT	NOTA_REDACAO
909189	533.3	485.8	530.2	595.5	540.0
909190	486.2	543.4	560.3	535.2	740.0
909191	500.8	563.2	555.3	442.8	780.0
909192	491.6	549.3	586.8	633.5	640.0
909193	425.7	525.0	511.5	621.8	620.0

Apesar da convenção no Python ser utilizar o .head() ou o .tail() para olhar um *Data Frame* novo, algo muito mais inteligente de se fazer para ter uma ideia representativa do *Data Frame* é utilizar o sample(5), que seleciona 5 linhas aleatórias do seu *Data Frame*. Se você executar o comando algumas vezes, irá obter resultados diferentes a cada iteração. Observe que, na imagem, a seguir, executamos duas vezes o comando e, desta forma, ele nos trouxe linhas aleatórias.

```
df.sample(5)
```

	NOTA_CN	NOTA_CH	NOTA_LC	NOTA_MT	NOTA_REDACAO
618259	394.0	499.8	524.1	623.2	420.0
152258	518.5	670.8	596.7	672.4	900.0
255114	578.1	589.1	598.3	670.1	640.0
395960	373.8	501.1	551.1	648.8	920.0
863337	568.7	606.3	558.5	637.6	680.0

```
df.sample(5)
```

	NOTA_CN	NOTA_CH	NOTA_LC	NOTA_MT	NOTA_REDACAO
782185	548.2	605.8	586.7	542.5	560.0
2351	403.8	363.8	482.5	438.7	600.0
169193	492.4	378.5	368.1	457.5	280.0
6263	543.3	617.9	614.8	580.8	880.0
728219	529.7	594.7	589.7	565.6	680.0

Invertendo linhas por colunas

Uma outra forma de visualização de dados é realizando a transposição dos dados. Para isso, basta colocar o comando **.T** depois do comando que gera uma visualização de um Data Frame. Acompanhe o exemplo a seguir.

```
df.sample(5).T
```

	402912	666006	769640	576054	167858
NOTA_CN	525.8	444.0	432.9	570.6	586.4
NOTA_CH	602.0	500.5	361.1	580.7	646.3
NOTA_LC	582.5	538.1	486.7	568.0	603.6
NOTA_MT	598.5	414.4	466.5	593.2	721.3
NOTA_REDACAO	700.0	540.0	340.0	880.0	780.0
NOTA_MEDIA	577.2	474.25	436.8	578.125	639.4
TP_ESCOLA	Privada	Pública	Pública	Privada	Pública
IDADE	18	17	18	17	18
SEXO	F	F	F	F	M

Assim, podemos utilizar diversos comando na mesma linha. No exemplo anterior, utilizamos o comando de visualização **.sample(5)** com a transposição .T. Essas funcionalidades irão atuar de forma encadeada: primeiro será executado o .sample(5), que irá buscar cinco registros aleatórios e, posteriormente, realizará a transposição, invertendo essa visualização de linhas para colunas.

Selecionando linhas

Outra forma de visualizar os cinco primeiros registros é utilizando o comando a seguir, informando o início e fim da seleção. E, se utilizarmos a função **.T**, iremos inverter a forma de apresentação dos dados.

```
df[0:5]
```

	NOTA_CN	NOTA_CH	NOTA_LC	NOTA_MT	NOTA_REDACAO	NOTA_MEDIA
0	574.2	538.7	582.6	571.5	640.0	566.750
1	369.7	398.2	395.6	473.8	480.0	409.325
2	423.0	372.8	488.7	516.5	540.0	450.250
3	412.5	417.2	485.1	446.2	580.0	440.250
4	460.7	570.5	541.8	631.3	640.0	551.075

```
df[0:5].T
```

	0	1	2	3	4
NOTA_CN	574.2	369.7	423.0	412.5	460.7
NOTA_CH	538.7	398.2	372.8	417.2	570.5
NOTA_LC	582.6	395.6	488.7	485.1	541.8
NOTA_MT	571.5	473.8	516.5	446.2	631.3
NOTA_REDACAO	640.0	480.0	540.0	580.0	640.0
NOTA_MEDIA	566.75	409.325	450.25	440.25	551.075

Podemos realizar a visualização de qualquer intervalo de dados, bastando informar quais linhas queremos. Veja o comando a seguir. Perceba que ele sempre irá retornar uma linha a menos do que informamos, isso pelo fato de que a primeira linha inicia no índice 0 (zero).

```
df[30:35]
```

	NOTA_CN	NOTA_CH	NOTA_LC	NOTA_MT	NOTA_REDACAO	NOTA_MEDIA
30	492.2	518.9	571.4	427.0	540.0	502.375
31	496.1	619.6	563.5	540.6	920.0	554.950
32	559.6	618.7	607.5	615.6	900.0	600.350
33	508.9	551.0	564.9	565.2	520.0	547.500
34	487.9	518.7	494.3	563.1	500.0	516.000

Slicing

O *Slicing* é a operação de colocar algo entre colchetes. Isso significa a seleção de um subconjunto de dados, o que foi utilizado anteriormente.

Assim, [20:30] vai selecionar exatamente 10 elementos (30 - 20). O número da direita menos o número da esquerda é sempre o total de elementos que serão selecionados, e isso é uma convenção no Python.

O número 20 é selecionado por ser um intervalo fechado na esquerda, e o número 30 não é selecionado por ser um intervalo aberto na direita. Essa é outra convenção do Python.

Selecionando colunas

Até agora, os comandos utilizados nos retornavam todas as colunas do *DataSet*, porém, se desejamos selecionar apenas uma coluna, devemos informar o nome do *DataSet* e, entre colchetes, o nome da coluna desejada. No exemplo, a seguir, solicitamos apenas a coluna NOTA_MEDIA dos alunos.

```
df['NOTA_MEDIA']
```

```
0    566.750
1    409.325
2    450.250
3    440.250
4    551.075
```

E, se você deseja mais de uma coluna? Para isso, basta utilizar o mesmo comando, porém entre colchetes, separado por vírgula, como no exemplo, a seguir.

```
df[['NOTA_MEDIA', 'IDADE', 'SEXO']]
```

	NOTA_MEDIA	IDADE	SEXO
0	566.750	18	F
1	409.325	26	F
2	450.250	17	F
3	440.250	18	F

Atenção

Neste caso, é necessário utilizar dois colchetes!

Vamos pensar um pouco? Que comando é necessário para selecionar as 10 primeiras linhas mostrando apenas as colunas NOTA_REDACAO e NOTA_MEDIA? Consegue imaginar?

Achou que iria ficar sem a resposta? Será que você acertou?

Basta utilizar primeiro o comando para selecionar as 10 primeiras linhas e posteriormente o comando de selecionar as colunas específicas. Gostou?

```
df[0:10][['NOTA_REDACAO', 'NOTA_MEDIA']]
```

Neste material, foi possível iniciar com os comandos básicos do Pandas. Porém, ainda há muito pela frente. Este é somente o começo!

REFERÊNCIAS

WES MCKINNEY AND THE PANDAS DEVELOPMENT TEAM. **Pandas**: powerful Python data analysis toolkit Release 1.4.4. Disponível em: https://pandas.pydata.org/pandas-docs/stable/pandas.pdf. Acesso em: 08 de ago. 2022.

SENAI <LAB365>