Introdução à Ciência de Dados

Prof. Dr. Fernando Maciano de Paula Neto
cin.ufpe.br/~fernando

Centro de Informática UFPE
Universidade Federal de Pernambuco

Centro de Informática UFPE
Universidade Federal de Pernambuco

Agenda

- Pandas
  - Acessar e filtrar o DataFrame
  - Atualização de DataFrame
  - Operações entre DataFrames
  - Pré-processamento de DataFrame

Centro de Informática UFPE | Universidade Federal de Pernambuco

Pandas

Acessar e Filtrar

cin.ufpe.br

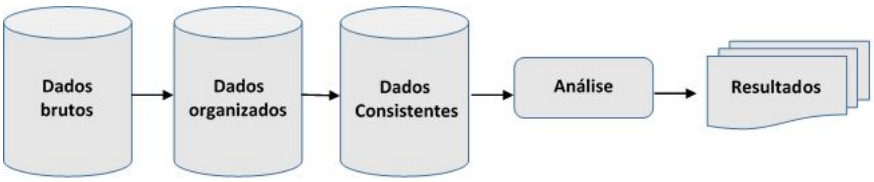
Centro de Informática UFPE | Universidade Federal de Pernambuco

Pandas é uma biblioteca de Python para *data wrangling* (processamento de dados) e análise.

A estrutura de dados é chamada de **DataFrame**. Basicamente um **DataFrame** é uma tabela, similar a uma planilha do Excel.

Pandas permite um conjunto diversificado de operações para manipular e alterar esta tabela.

Permite comandos/consultas que se parecem com o SQL.



cin.ufpe.br

Ao contrário do numpy que requer que os dados sejam do mesmo tipo, **DataFrame** possuem dados separados por colunas que possuem tipos específicos.

É possível carregar um **DataFrame** a partir de arquivos de diferentes formatos: CSV, Excel, bases de dados.

Centro de Informática UFPE
Universidade Federal de Pernambuco

Base de Dados do GROUPLENS

Grupo de Ciência de Dados (Computação Social) de Minnesota

https://grouplens.org

- Pessoas preencheram avaliações de filmes

- Dados das pessoas (exemplo: Pedro, João, Maria)
- Dados dos filmes (Titanic, Matrix)
- Dados das avaliações (Pedro avaliou Titanic com nota 4; João avaliou Matrix com nota 5)

Base de dados - Grouplens

Base de pessoas

Link: http://files.grouplens.org/datasets/movielens/ml-100k/u.user

Atributos: idade/gênero/ocupação/CEP de 943 pessoas

```
1|24|M|technician|85711
2|53|F|other|94043
3|23|M|writer|32067
4|24|M|technician|43537
5|33|F|other|15213
6|42|M|executive|98101
7|57|M|administrator|91344
8|36|M|administrator|05201
9|29|M|student|01002
10|53|M|lawyer|90703
11|39|F|other|30329
12|28|F|other|06405
13|47|M|educator|29206
14|45|M|scientist|55106
15|49|F|educator|97301
16|21|M|entertainment|10309
17|30|M|programmer|06355
18|35|F|other|37212
19|40|M|librarian|02138
20|42|F|homemaker|95660
21|26|M|writer|30068
22|25|M|writer|40206
23|30|F|artist|48197
24|21|F|artist|94533
25|39|M|engineer|55107
26|49|M|engineer|21044
27|40|F|librarian|30030
28|32|M|writer|55369
29|41|M|programmer|94043
30|7|M|student|55436
31|24|M|artist|10003
32|28|F|student|78741
33|23|M|student|27510
34|38|F|administrator|42141
35|20|F|homemaker|42459
36|19|F|student|93117
37|23|M|student|55105
38|28|F|other|54467
```


Exemplo de carregamento

caminho do arquivo ou URL link

```
import pandas as pd
usuarios = pd.read_csv(
    "http://files.grouplens.org/datasets/movielens/ml-100k/u.user",
    sep='|',header=None, names=["user_id", "age", "gender", "occupation", "zip_code"])
print("Tipo da variavel usuarios", type(usuarios))
print("Shape", usuarios.shape)
usuarios.head()
```

```
Tipo da variavel usuarios <class 'pandas.core.frame.DataFrame'>
Shape (943, 5)
```

Out[1]:

	user_id	age	gender	occupation	zip_code
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213

head é uma função muito utilizada para visualizar as primeiras linhas do DataFrame. Para "dar uma conferida".

Base de filmes:

Link: http://files.grouplens.org/datasets/mov

Atributos:
Nome do filme

Data de lançamento

Data de lan;camento em video

Url da IMDb

Tipo do filme:

desconhecido, ação, aventura,

animação, infantil, comédia, ...

```
1|Toy Story (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Toy%20Story%20(1995)|0|0|0|1|1|1|0|0|0|0|0|0|0|0|0|0|0|0
2|GoldenEye (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?GoldenEye%20(1995)|0|1|1|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
3|Four Rooms (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Four%20Rooms%20(1995)|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
4|Get Shorty (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Get%20Shorty%20(1995)|0|1|0|0|0|1|0|0|1|0|0|0|0|0|0|0|0|0
5|Copycat (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Copycat%20(1995)|0|0|0|0|0|0|1|0|1|0|0|0|0|0|0|1|0|0
6|Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)|01-Jan-1995||http://us.imdb.com/Title?Yao+a+yao+yao+dao+waipo+qiao+(1995)|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
7|Twelve Monkeys (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Twelve%20Monkeys%20(1995)|0|0|0|0|0|0|0|0|0|1|1|0|0|0|0|0|0|1|0|0|0|0
8|Babe (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Babe%20(1995)|0|0|0|0|1|1|0|0|1|0|0|0|0|0|0|0|0|0|0|0
9|Dead Man Walking (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Dead%20Man%20Walking%20(1995)|0|0|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0
10|Richard III (1995)|22-Jan-1996||http://us.imdb.com/M/title-exact?Richard%20III%20(1995)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|1|0
11|Seven (Se7en) (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Se7en%20(1995)|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|1|0|0
12|Usual Suspects, The (1995)|14-Aug-1995||http://us.imdb.com/M/title-exact?Usual%20Suspects,%20The%20(1995)|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0|1|0
13|Mighty Aphrodite (1995)|30-Oct-1995||http://us.imdb.com/M/title-exact?Mighty%20Aphrodite%20(1995)|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0|0
14|Postino, Il (1994)|01-Jan-1994||http://us.imdb.com/M/title-exact?Postino,%20Il%20(1994)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|1|0|0|0|0|0
15|Mr. Holland's Opus (1995)|29-Jan-1996||http://us.imdb.com/M/title-exact?Mr.%20Holland's%20Opus%20(1995)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0
16|French Twist (Gazon maudit) (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Gazon%20maudit%20(1995)|0|0|0|0|0|1|0|0|0|0|0|0|0|0|1|0|0|0|
17|From Dusk Till Dawn (1996)|05-Feb-1996||http://us.imdb.com/M/title-exact?From%20Dusk%20Till%20Dawn%20(1996)|0|1|0|0|0|1|1|0|0|0|0|1|0|0|0|0|0|1
18|White Balloon, The (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Badkonake%20Sefid%20(1995)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0
19|Antonia's Line (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Antonia%20(1995)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0
20|Angels and Insects (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Angels%20and%20Insects%20(1995)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|1|0|0|0|0
21|Muppet Treasure Island (1996)|16-Feb-1996||http://us.imdb.com/M/title-exact?Muppet%20Treasure%20Island%20(1996)|0|1|1|0|0|1|0|0|0|0|0|0|1|0|0|0
22|Braveheart (1995)|16-Feb-1996||http://us.imdb.com/M/title-exact?Braveheart%20(1995)|0|1|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|1|0
23|Taxi Driver (1976)|16-Feb-1996||http://us.imdb.com/M/title-exact?Taxi%20Driver%20(1976)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|1|0|0
24|Rumble in the Bronx (1995)|23-Feb-1996||http://us.imdb.com/M/title-exact?Hong%20Faan%20Kui%20(1995)|0|1|1|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0
25|Birdcage, The (1996)|08-Mar-1996||http://us.imdb.com/M/title-exact?Birdcage,%20The%20(1996)|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0|0
26|Brothers McMullen, The (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Brothers%20McMullen,%20The%20(1995)|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0|0
27|Bad Boys (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Bad%20Boys%20(1995)|0|1|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
28|Apollo 13 (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Apollo%2013%20(1995)|0|1|0|0|0|0|0|0|1|0|0|0|0|0|0|0|1|0|0
29|Batman Forever (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Batman%20Forever%20(1995)|0|1|1|0|0|1|1|0|0|0|0|0|0|0|0|0|0|0
30|Belle de jour (1967)|01-Jan-1967||http://us.imdb.com/M/title-exact?Belle%20de%20jour%20(1967)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0
31|Crimson Tide (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Crimson%20Tide%20(1995)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|1|1|0
32|Crumb (1994)|01-Jan-1994||http://us.imdb.com/M/title-exact?Crumb%20(1994)|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0
33|Desperado (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Desperado%20(1995)|0|1|0|0|0|0|0|0|0|0|0|0|0|0|1|0|1|0|0
34|Doom Generation, The (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Doom%20Generation,%20The%20(1995)|0|0|0|0|0|1|0|0|1|0|0|0|0|0|0|0|0|0|0
35|Free Willy 2: The Adventure Home (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?
Free%20Willy%202:%20The%20Adventure%20Home%20(1995)|0|0|1|0|1|0|0|0|1|0|0|0|0|0|0|0|0|0|0
36|Mad Love (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Mad%20Love%20(1995)|0|0|0|0|0|0|0|1|0|0|0|0|0|1|0|0|0|0
37|Nadja (1994)|01-Jan-1994||http://us.imdb.com/M/title-exact?Nadja%20(1994)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
38|Net, The (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Net,%20The%20(1995)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|1|1|0|0
39|Strange Days (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Strange%20Days%20(1995)|0|1|0|0|0|0|1|0|0|0|0|0|0|0|0|1|0|0|0
40|To Wong Foo, Thanks for Everything! Julie Newmar (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?
To%20Wong%20Foo,%20Thanks%20for%20Everything!%20Julie%20Newmar%20(1995)|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0|0|0
41|Billy Madison (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Billy%20Madison%20(1995)|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0|0
42|Clerks (1994)|01-Jan-1994||http://us.imdb.com/M/title-exact?Clerks%20(1994)|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0|0|0
43|Disclosure (1994)|01-Jan-1994||http://us.imdb.com/M/title-exact?Disclosure%20(1994)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|1|0|0
44|Dolores Claiborne (1994)|01-Jan-1994||http://us.imdb.com/M/title-exact?Dolores%20Claiborne%20(1994)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|1|0|0
45|Eat Drink Man Woman (1994)|01-Jan-1994||http://us.imdb.com/M/title-exact?Yinshi%20Nan%20Nu%20(1994)|0|0|0|0|0|1|0|0|1|0|0|0|0|0|0|0|0|0|0
```

Centro de Informática UFPE — Universidade Federal de Pernambuco

```
filmes = pd.read_csv(

    "http://files.grouplens.org/datasets/movielens/ml-100k/u.item",

    sep='|',header=None, names=["movie_id", "movie_title",  "release_date",
"video_release_date", "IMDb_URL", "unknown", "Action", "Adventure", "Animation",

        "Children", "Comedy", "Crime", "Documentary", "Drama", "Fantasy","FilmNoir",
"Horror", "Musical", "Mystery", "Romance", "SciFi","Thriller", "War", "Western"],

    encoding='latin1')

print(filmes.shape)

filmes.head()
```

	movie_id	movie_title	release_date	video_release_date	IMDb_URL	unknown	Action	Adventure	Animation	Children	...	Fantasy	FilmNoir
0	1	Toy Story (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%2...	0	0	0	1	1	...	0	0
1	2	GoldenEye (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?GoldenEye%20(...	0	1	1	0	0	...	0	0
2	3	Four Rooms (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Four%20Rooms%...	0	0	0	0	0	...	0	0
3	4	Get Shorty (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Get%20Shorty%...	0	1	0	0	0	...	0	0
4	5	Copycat (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Copycat%20(1995)	0	0	0	0	0	...	0	0

Base de avaliações

Link: http://files.grouplens.org/datasets/movielens/ml-100k/u.data

Atributos:

Id do usuário,

Id do filme,

Avaliação (1,2,3,4,5)

Timestamp (marca temporal)

O *timestamp do unix* corresponde ao número de segundos desde a meia-noite do dia 01/01/1970 no fuso horário UTC sem considerar os segundos bissextos (tal como mencionado na resposta do ctgPi). Para simplificar, vamos denominar este momento no tempo de ponto zero. Assim, o *timestamp do unix* é o número de segundos desde o ponto zero.

```
196	242	3	881250949
186	302	3	891717742
22	377	1	878887116
244	51	2	880606923
166	346	1	886397596
298	474	4	884182806
115	265	2	881171488
253	465	5	891628467
305	451	3	886324817
6	86	3	883603013
62	257	2	879372434
286	1014	5	879781125
200	222	5	876042340
210	40	3	891035994
224	29	3	888104457
303	785	3	879485318
122	387	5	879270459
194	274	2	879539794
291	1042	4	874834944
234	1184	2	892079237
119	392	4	886176814
167	486	4	892738452
299	144	4	877881320
291	118	2	874833878
308	1	4	887736532
95	546	2	879196566
38	95	5	892430094
102	768	2	883748450
63	277	4	875747401
160	234	5	876861185
50	246	3	877052329
301	98	4	882075827
225	193	4	879539727
290	88	4	880731963
97	194	3	884238860
157	274	4	886890835
181	1081	1	878962623
```



```
avaliacoes = pd.read_csv(
    "http://files.grouplens.org/datasets/movielens/ml-100k/u.data",
    sep='\t',header=None, names=["user_id", "movie_id", "rating", "timestamp"])
print(avaliacoes.shape)
avaliacoes.head()
```

	user_id	movie_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

Estatística descritiva dos dados

```
print(usuarios.describe(),"\n")
```

```
          user_id         age
count  943.000000  943.000000
mean   472.000000   34.051962
std    272.364951   12.192740
min      1.000000    7.000000
25%    236.500000   25.000000
50%    472.000000   31.000000
75%    707.500000   43.000000
max    943.000000   73.000000
```

Nessa etapa, checamos "visualmente" se os valores fazem sentido.

O valor mínimo e máximo fazem sentido?
A média faz sentido?

Tipos dos dados

```
print(usuarios.dtypes)
```

```
user_id       int64
age           int64
gender       object
occupation   object
zip_code     object
dtype: object
```

Tipos dos dados

print(filmes.**dtypes**)

```
movie_id               int64
movie_title           object
release_date          object
video_release_date   float64
IMDb_URL              object
unknown                int64
Action                 int64
Adventure              int64
Animation              int64
Children               int64
Comedy                 int64
Crime                  int64
Documentary            int64
Drama                  int64
Fantasy                int64
FilmNoir               int64
Horror                 int64
Musical                int64
Mystery                int64
Romance                int64
SciFi                  int64
Thriller               int64
War                    int64
Western                int64
dtype: object
```

Em algumas colunas, o Pandas consegue identificar automaticamente o tipo.
Em outras, será preciso editar o tipo.
Faremos isso mais adiante, na etapa de Pré-processamento.

Acessando colunas

 # Selecionando uma coluna

print(usuarios['age'].**head**())

print(usuarios.age.**head**())

```
0    24
1    53
2    23
3    24
4    33
Name: age, dtype: int64
0    24
1    53
2    23
3    24
4    33
Name: age, dtype: int64
```

	user_id	age	gender	occupation	zip_code
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213

 # Selecionando multiplas colunas

usuarios[['age','gender']].head()

	age	gender
0	24	M
1	53	F
2	23	M
3	24	M
4	33	F

 # Selecionando linhas com slice

avaliacoes[10:15]

	user_id	movie_id	rating	timestamp
10	62	257	2	879372434
11	286	1014	5	879781125
12	200	222	5	876042340
13	210	40	3	891035994
14	224	29	3	888104457

 # Nao e possivel combinar selecao de linha e coluna diretamente via []
avaliacoes[1:5,'rating']

isso aqui dá erro!

filmes[15:20]

	movie_id	movie_title	release_date	video_release_date	IMDb_URL	unknown	Action	Adventure	Animation
15	16	French Twist (Gazon maudit) (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Gazon%20maudi...	0	0	0	0
16	17	From Dusk Till Dawn (1996)	05-Feb-1996	NaN	http://us.imdb.com/M/title-exact?From%20Dusk%2...	0	1	0	0
17	18	White Balloon, The (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Badkonake%20S...	0	0	0	0
18	19	Antonia's Line (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Antonia%20(1995)	0	0	0	0
19	20	Angels and Insects (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Angels%20and%...	0	0	0	0

5 rows × 24 columns

Centro de Informática UFPE
Universidade Federal de Pernambuco

 # A selecao de porcoes especificas sao feitas atraves dos atributos
 # loc permite selecao por labels

```
print(avaliacoes.loc[10:15,'rating'])
```

```
10    2
11    5
12    5
13    3
14    3
15    3
Name: rating, dtype: int64
```

```
print(avaliacoes.loc[10:15, [movie_id,'rating']])
```

	movie_id	rating
10	257	2
11	1014	5
12	222	5
13	40	3
14	29	3
15	785	3

 # iloc permite a selecao por indice

```
display(filmes.iloc[:5,[1,2,4]])
```

	movie_title	release_date	IMDb_URL
0	Toy Story (1995)	01-Jan-1995	http://us.imdb.com/M/title-exact?Toy%20Story%2...
1	GoldenEye (1995)	01-Jan-1995	http://us.imdb.com/M/title-exact?GoldenEye%20(...
2	Four Rooms (1995)	01-Jan-1995	http://us.imdb.com/M/title-exact?Four%20Rooms%...
3	Get Shorty (1995)	01-Jan-1995	http://us.imdb.com/M/title-exact?Get%20Shorty%...
4	Copycat (1995)	01-Jan-1995	http://us.imdb.com/M/title-exact?Copycat%20(1995)

cin.ufpe.br

linhas

```
avaliacoes.loc[10:15,'rating'])
```

coluna

```
10    2
11    5
12    5
13    3
14    3
15    3
Name: rating, dtype: int64
```

linhas

```
avaliacoes.loc[10:15,['rating','timestamp']]
```

lista das colunas

	rating	timestamp
10	2	879372434
11	5	879781125
12	5	876042340
13	3	891035994
14	3	888104457
15	3	879485318

 # iloc permite a selecao por indice

```
display(filmes.iloc[2:10,[1,2,4]])
```

linhas

colunas

	movie_title	release_date	IMDb_URL
2	Four Rooms (1995)	01-Jan-1995	http://us.imdb.com/M/title-exact?Four%20Rooms%...
3	Get Shorty (1995)	01-Jan-1995	http://us.imdb.com/M/title-exact?Get%20Shorty%...
4	Copycat (1995)	01-Jan-1995	http://us.imdb.com/M/title-exact?Copycat%20(1995)
5	Shanghai Triad (Yao a yao yao dao waipo qiao) ...	01-Jan-1995	http://us.imdb.com/Title?Yao+a+yao+yao+dao+wai...
6	Twelve Monkeys (1995)	01-Jan-1995	http://us.imdb.com/M/title-exact?Twelve%20Monk...
7	Babe (1995)	01-Jan-1995	http://us.imdb.com/M/title-exact?Babe%20(1995)
8	Dead Man Walking (1995)	01-Jan-1995	http://us.imdb.com/M/title-exact?Dead%20Man%20...
9	Richard III (1995)	22-Jan-1996	http://us.imdb.com/M/title-exact?Richard%20III...

x = avaliacoes.loc[10:15, ['movie_id','rating']]

	movie_id	rating
10	257	2
11	1014	5
12	222	5
13	40	3
14	29	3
15	785	3

Podemos acessar o conteúdo do Dataframe pelo índice:

```
In [31]: 1 x.iloc[0,1]
Out[31]: 2

In [34]: 1 x.iloc[3,0]
Out[34]: 40
```


Centro de Informática UFPE
Universidade Federal de Pernambuco

Filtros

 # Ainda podemos selecionar os dados com mascaras

 # booleanas como em NumPy

usuarios[usuarios.age > 40].head()

	user_id	age	gender	occupation	zip_code
1	2	53	F	other	94043
5	6	42	M	executive	98101
6	7	57	M	administrator	91344
9	10	53	M	lawyer	90703
12	13	47	M	educator	29206

Filtros

Entendendo um pouco mais a fundo...

usuarios['age'] > 40

```
usuarios['age'] > 40
```

```
0      False
1       True
2      False
3      False
4      False
       ...
938    False
939    False
940    False
941     True
942    False
Name: age, Length: 943, dtype: bool
```

usuarios[usuarios['age'] > 40]

	user_id	age	gender	occupation	zip_code
1	2	53	F	other	94043
5	6	42	M	executive	98101
6	7	57	M	administrator	91344
9	10	53	M	lawyer	90703
12	13	47	M	educator	29206
...	...	...	...	...	...
931	932	58	M	educator	06437
933	934	61	M	engineer	22902
934	935	42	M	doctor	66221
936	937	48	M	educator	98072
941	942	48	F	librarian	78209

272 rows × 5 columns

Filtros

 # Ainda podemos selecionar os dados com mascaras

 # booleanas como em NumPy

usuarios[usuarios.occupation.isin(['executive','educator'])].head()

	user_id	age	gender	occupation	zip_code
5	6	42	M	executive	98101
12	13	47	M	educator	29206
14	15	49	F	educator	97301
50	51	28	M	educator	16509
53	54	22	M	executive	66315

seleciona usuários que possuem na coluna de ocupação o conteúdo "executive" ou "educator"

Filtros

 # Ainda podemos selecionar os dados com mascaras

 # booleanas como em NumPy

```
usuarios[(usuarios.age > 40) & ~(usuarios.occupation.isin(['none','other']))].head()
```

	user_id	age	gender	occupation	zip_code
5	6	42	M	executive	98101
6	7	57	M	administrator	91344
9	10	53	M	lawyer	90703
12	13	47	M	educator	29206
13	14	45	M	scientist	55106

Filtros

Como saber a média das mulheres cientistas que são usuárias?

```
cientistas = usuarios[(usuarios.gender=='F') &
(usuarios.occupation=='scientist')]

print(cientistas.age.mean())
```

```
28.333333333333332
```

```
     user_id  age gender occupation zip_code
174      175   26      F  scientist    21911
729      730   31      F  scientist    32114
929      930   28      F  scientist    07310
28.333333333333332
```

In [ ]:

Quantos filmes de animação foram lançados em 1995?

```
filmes[(filmes.Animation==1)& filmes.release_date.str.contains('1995')].head()
```

	movie_id	movie_title	release_date	video_release_date	IMDb_URL	unknown	Action	Adventure	Animation	Children	...	Fantasy	FilmNo
0	1	Toy Story (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%2...	0	0	0	1	1	...	0	
541	542	Pocahontas (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Pocahontas%20...	0	0	0	1	1	...	0	
1065	1066	Balto (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Balto%20(1995)	0	0	0	1	1	...	0	
1218	1219	Goofy Movie, A (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Goofy%20Movie...	0	0	0	1	1	...	0	
1411	1412	Land Before Time III: The Time of the Great Gi...	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Land%20Before...	0	0	0	1	1	...	0	
1469	1470	Gumby: The Movie (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Gumby:%20The%...	0	0	0	1	1	...	0	

shape retorna a quantidade de linhas e colunas do DataFrame

```
filmes[(filmes.Animation==1)& filmes.release_date.str.contains('1995')].shape
(6,24)
filmes[(filmes.Animation==1)& filmes.release_date.str.contains('1995')].shape[0]
6
```


Pandas

Atualização de DataFrame

cin.ufpe.br

Alterando valores do DataFrame

```
c2 = cientistas.copy()

c2.iloc[0,1] = -1

c2.loc[:,'zip_code'] = None

c2.head()
```

Linha 0 , Coluna 1

todas as linhas, da coluna "zip_code"

	user_id	age	gender	occupation	zip_code
174	175	-1	F	scientist	None
729	730	31	F	scientist	None
929	930	28	F	scientist	None


```
import re

re.search("\d+\-(\w+)\-\d+",  "12-Jan-1990").group(0)

>>'12-Jan-1990'

re.search("\d+\-(\w+)\-\d+",  "12-Jan-1990").group(1)

>>'Jan'

re.search("\d+\-\w+\-(\d+)",  "12-Jan-1990").group(1)

>>'1990'
```

```
def funcaoPegaAno(data):
    if re.search("\d+\-\w+\-(\d+)",str(data)) == None:
        return None
    else:
        return (re.search("\d+\-\w+\-(\d+)",str(data)).group(1))

filmes['release_year'] = filmes.release_date.apply(funcaoPegaAno)
```


Centro de Informática UFPE
Universidade Federal de Pernambuco

Funções lambda

```
#Normal python function
def a_name(x):
    return x+x
#Lambda function
lambda x: x+x
```

(lambda x: x+x)(2)
>>4

```
import re
import numpy as np

filmes['release_year'] = filmes.release_date.apply(
        lambda x: not x is np.nan and
    re.search("\d+\-\w+\-(\d+)",str(x)).group(1) or None)
```

	movie_title	release_date	release_year
0	Toy Story (1995)	01-Jan-1995	1995
1	GoldenEye (1995)	01-Jan-1995	1995
2	Four Rooms (1995)	01-Jan-1995	1995
3	Get Shorty (1995)	01-Jan-1995	1995
4	Copycat (1995)	01-Jan-1995	1995

Centro de Informática UFPE
Universidade Federal de Pernambuco

Operações básicas

Média e Mediana de uma determinada coluna:

print("Mediana de idade dos usuarios ", usuarios.age.median())

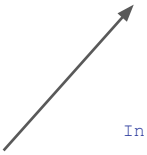
>>Mediana de idade dos usuarios 31.0

print("Mediana de idade dos usuarios ", usuarios.age.mean())
>>Mediana de idade dos usuarios 34.05196182396607

Seleção

```
piorAvaliacao = avaliacoes.rating.argmin()
print(filmes[filmes.movie_id == avaliacoes.movie_id.iloc[piorAvaliacao]].movie_title.iloc[0])
```

>> Heavyweights (1994)



In [ ]:

imprimir somente o nome do filme.
se tirar o iloc[0] vai imprimir informações da coluna movie_title como o tipo dela.


```
avaliacoes.rating.value_counts()
```

```
4    34174
3    27145
5    21201
2    11370
1     6110
Name: rating, dtype: int64
```

Pandas

Operações entre DataFrames

cin.ufpe.br

Concat

```
# A forma mais simples de juntar diferentes dados e concatenacao

A = pd.Series(["A{}".format(a) for a in range(4)],
              index=range(4), name="A")

B = pd.Series(["B{}".format(a) for a in range(4)],
              index=range(4), name= "B")

C = pd.Series(["C{}".format(a) for a in range(5)],
              index=range(5), name= "C")
```

```
1 print(A)
```

```
0    A0
1    A1
2    A2
3    A3
Name: A, dtype: object
```

```
1 print(B)
```

```
0    B0
1    B1
2    B2
3    B3
Name: B, dtype: object
```

```
1 print(C)
```

```
0    C0
1    C1
2    C2
3    C3
4    C4
Name: C, dtype: object
```

cin.ufpe.br

```
pd.concat([A,B,C],axis=1)
```

	A	B	C
0	A0	B0	C0
1	A1	B1	C1
2	A2	B2	C2
3	A3	B3	C3
4	NaN	NaN	C4

Centro de Informática UFPE
Universidade Federal de Pernambuco

Merge

pd.merge(left, right, how='inner',

on=**None**, left_on=**None**, right_on=**None**,

left_index=**False**, right_index=**False**, sort=**True**,

suffixes=('_x', '_y'),copy=**True**,indicator=**False**)

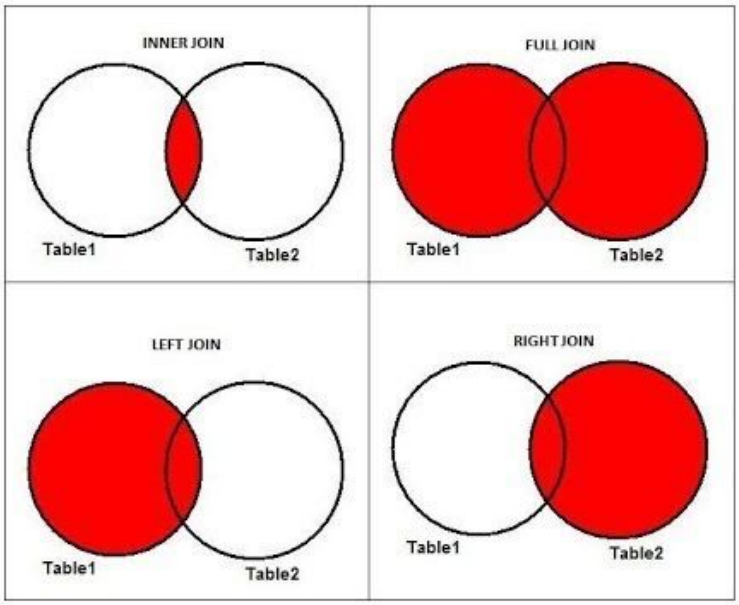
left: tabela à esquerda
right: tabela à direita
how: inner join
on: a coluna em comum

Filmes

	movie_id	movie_title	release_date	video_release_date	IMDb_URL	unknown	Action	Adventure	Animation	Children	...	Fantasy	FilmNoir
0	1	Toy Story (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%2...	0	0	0	1	1	...	0	0
1	2	GoldenEye (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?GoldenEye%20(...	0	1	1	0	0	...	0	0
2	3	Four Rooms (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Four%20Rooms%...	0	0	0	0	0	...	0	0
3	4	Get Shorty (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Get%20Shorty%...	0	1	0	0	0	...	0	0
4	5	Copycat (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Copycat%20(1995)	0	0	0	0	0	...	0	0

Avaliações

	user_id	movie_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596




```
avaliacaoFilmes =
pd.merge(avaliacoes,filmes[["movie_id","movie_title","release_year"]],

                 on="movie_id")

avaliacaoFilmes.head()
```

	user_id	movie_id	rating	timestamp	movie_title	release_year
0	196	242	3	881250949	Kolya (1996)	1997
1	63	242	3	875747190	Kolya (1996)	1997
2	226	242	5	883888671	Kolya (1996)	1997
3	154	242	3	879138235	Kolya (1996)	1997
4	306	242	5	876503793	Kolya (1996)	1997

Split-apply-combine

Agrupar dados a partir de um dado critério antes de colocar um filtro.

Por exemplo: qual a média das avaliações por gênero de filme?

- O procedimento é executado da seguinte forma:
    1. Split: dividir dados conforme um critério (e.g. coluna)
    2. Apply: aplicar uma determinada função a cada um dos grupos
    3. Combine: reunir resultados em uma única estrutura de dados

Centro de Informática UFPE
Universidade Federal de Pernambuco

- A função que será aplicada aos grupos pode ter como objetivo:
    - Sumarizar informações: computar estatísticas descritivas dos grupos (média, desvios, mínimo/máximo, contar valores, etc.)
    - Transformar valores: normalizar dados nos grupos; gerar novos atributos; preencher dados ausentes
    - Filtrar dados: eliminar grupos a partir de estatísticas computadas para o grupo

Agrupar por

```
usuarioPorProfissao = usuarios.groupby(usuarios.occupation)

usuarioPorProfissao.age.mean()
```

```
occupation
administrator    38.746835
artist           31.392857
doctor           43.571429
educator         42.010526
engineer         36.388060
entertainment    29.222222
executive        38.718750
healthcare       41.562500
homemaker        32.571429
lawyer           36.750000
librarian        40.000000
marketing        37.615385
none             26.555556
```

usuarios.head()

	user_id	age	gender	occupation	zip_code
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213

usuarioPorProfissao.mean()

occupation	user_id	age
administrator	430.949367	38.746835
artist	451.892857	31.392857
doctor	533.714286	43.571429
educator	466.905263	42.010526

Pandas

Pré-processamento de DataFrame

Pré-processamento de dados

Tipos de atributos

- nominal
- binário
- ordinal
- numérico

cin.ufpe.br

Atributos nominais

Os atributos nominais ou categóricos são descrições **qualitativas** de objetos. Eles são nomes ou categorias que são atribuídas a um objeto. Por exemplo, a *cor_de_cabelo* é um atributo categórico para uma pessoa, podendo assumir os valores *ruivo*, *louro*, *preto*, *castanho*, *grisalho*, *branco*.

Atributos binários

Atributos binários são um caso particular de atributos nominais em que existem apenas duas categorias. Em geral representamos tais atributos por **0** e **1**. Também frequentemente associamos 0 com a ausência do atributo e 1 com a presença. Exemplos de atributos binários são: *fumante* (numa base de dados médicos, indicando se a pessoa é ou não fumante),

Atributos ordinais

Atributos ordinais são aqueles em que existe uma ordem (total) definida entre suas possíveis categorias. Por exemplo, tamanho de roupas divididos P, M, e G. Sabemos que existe uma ordem entre as possíveis categorias, mas não sabemos quantificar tal diferença. Não sabemos quão grande é a diferença de tamanho entre o P e o G.

cin.ufpe.br

Atributos numéricos

Esses atributos representam quantidades mensuradas. Os atributos numéricos podem ser grandezas reais ou inteiras. Eles estão subdivididos em dados **intervalares** e de **razão**. Nos dados intervalares existe uma relação de ordem entre os diversos valores, e além disso a diferença entre dois valores possui significado. O exemplo clássico é o atributo temperatura. Sabemos que 40 graus é mais quente que 30 (uma diferença de 10 graus). O mesmo pode ser dito em relação a 20 e 10 graus (novamente uma diferença de 10 graus). Contudo, não podemos dizer que um dia em que foi registrado 40 graus é 2x mais quente que um dia que foi registrado 20 graus. Isso é bastante evidente se mudarmos a escala de Celsius para Fahrenheit. 40 graus C é equivalente a 104 F, e 20C equivalente a 68F. Nesse caso, temos uma diferença de aproximadamente 1,5x. O mesmo ocorre quando comparamos anos (em datas). A razão é simples: não sabemos a localização exata do zero. O que definimos como zero tanto na escala de temperatura quanto no calendário é uma mera convenção. Mesmo assim, podemos calcular médias, desvios e outras estatísticas de dados intervalares.

Os dados de razão são aqueles em que o zero absoluto é conhecido. Esse ponto representa a total ausência do atributo. A temperatura medida em Kelvin, ao contrário da escala de Celsius e Fahrenheit, é uma escala de razão; 20K é duas vezes mais quente que 10K. O mesmo ocorre, por exemplo, quando comparamos uma pessoa que recebe um salário de R\$20\.000,00, recebe cinco vezes mais que alguém que receba R\$4.000,00.

Centro de Informática UFPE
Universidade Federal de Pernambuco

Ajustando as colunas do DataFrame

```
print(usuarios.dtypes)
```

```
usuarios.head()
```

```
user_id        int64
age            int64
gender        object
occupation    object
zip_code      object
dtype: object
```

	user_id	age	gender	occupation	zip_code
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213

user_id e age estão ok: são inteiros!

Mas gênero, ocupação e zip_code não.

```
usuarios['gender'] = usuarios.gender.astype('category')
```

Centro de Informática UFPE — Universidade Federal de Pernambuco

Atualizar o tipo do atributo para category permite realizar várias funções próprias de atributos do tipo categórico.

Centro de Informática UFPE — Universidade Federal de Pernambuco

usuarios.gender.cat.categories = ['Feminino', 'Masculino']

usuarios.gender

```
0      Masculino
1       Feminino
2      Masculino
3      Masculino
4       Feminino
5      Masculino
6      Masculino
7      Masculino
8      Masculino
9      Masculino
10      Feminino
11      Feminino
12     Masculino
13     Masculino
14      Feminino
15     Masculino
16     Masculino
17      Feminino
18     Masculino
19      Feminino
20     Masculino
21     Masculino
22      Feminino
```

cin.ufpe.br

```
usuarios.occupation = usuarios.occupation.astype('category')
usuarios.zip_code = usuarios.zip_code.astype('category')
filmes.iloc[:,5:] = filmes.iloc[:,5:].astype('bool')
filmes.release_date = pd.to_datetime(filmes.release_date, format="%d-%b-%Y")
```

```
print(filmes.release_date.describe())
```

antes do tipo categórico

```
count            1681
unique            240
top       01-Jan-1995
freq              215
Name: release_date, dtype: object
```

depois do tipo categórico

```
count                    1681
unique                    240
top       1995-01-01 00:00:00
freq                      215
first     1922-01-01 00:00:00
last      1998-10-23 00:00:00
Name: release_date, dtype: object
```



```
filmes.iloc[:,5:].apply(pd.value_counts)
```

	unknown	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	FilmNoir	Horror	Musical	Mystery	Romance	SciF
False	1680	1431	1547	1640	1560	1177	1573	1632	957	1660	1658	1590	1626	1621	1435	158
True	2	251	135	42	122	505	109	50	725	22	24	92	56	61	247	10

Avaliação (rating) pode ser tratada como atributo numérico. Poderia ser como categórico já que possui valores discretos possíveis 0, 1, 2, 3, 4 ou 5.

Vamos tratar como atributo ordinal.

CategoricalDtype: Type for categorical data with the categories and orderedness.

```
from pandas.api.types import CategoricalDtype

avaliacoes.rating =
avaliacoes.rating.astype(CategoricalDtype(categories=avaliacoes.rating.unique().sort(), ordered=True))
avaliacoes.rating.describe()
```

antes

```
count    100000.000000
mean          3.529860
std           1.125674
min           1.000000
25%           3.000000
50%           4.000000
75%           4.000000
max           5.000000
Name: rating, dtype: float64
```

depois

```
count     100000
unique         5
top            4
freq       34174
Name: rating, dtype: int64
```



```
avaliacoes.timestamp = pd.to_datetime(avaliacoes.timestamp,unit='s')

avaliacoes.timestamp.describe()
```

```
count                  100000
unique                  49282
top       1998-03-27 21:20:06
freq                       12
first     1997-09-20 03:05:10
last      1998-04-22 23:10:38
Name: timestamp, dtype: object
```

Centro de Informática UFPE — Universidade Federal de Pernambuco

Colunas com dados ausentes:

- Ignorar as linhas com dados ausentes (apagá-las)
- Preencher manualmente o valor ausente
- Preencher com um valor global para todos
- Usar alguma medida de tendência para o valor ausente (a média, mediana, etc)
- Usar alguma medida de tendência para o valor ausente a depender do grupo que pertença aquela linha
- Usar estatística inferencial ou aprendizagem de máquina para inferir o valor

Centro de Informática UFPE — Universidade Federal de Pernambuco

Apagando linhas ou colunas com dados faltantes

DataFrame.**dropna**(*axis=0, how='any', thresh=None, subset=None, inplace=False*)[source]

axis: 0 para linhas 1 para colunas

how: any: se qualquer valor ausente está presente, apague aquela linha ou coluna

all: se todos os valores ausentes naquela linha ou coluna, apague

subset: rótulos das linhas ou colunas a procurar de valores faltantes

```
df = pd.DataFrame({"Nome": ['Maria Betania', 'Caetano Veloso', 'Djavan'],
                   "Cidade Natal": ["Salvador", np.nan, 'Maceio'],
                   "Nascimento": [pd.Timestamp("1946-08-18"), pd.NaT,
                            pd.NaT],
                   "Melhor lugar":[np.nan, np.nan, np.nan]})
```

	Nome	Cidade Natal	Nascimento	Melhor lugar
0	Maria Betania	Salvador	1946-08-18	NaN
1	Caetano Veloso	NaN	NaT	NaN
2	Djavan	Maceio	NaT	NaN

Centro de Informática UFPE
Universidade Federal de Pernambuco

df.isnull().sum()

```
Nome            0
Cidade Natal    1
Nascimento      2
Melhor lugar    3
dtype: int64
```

	Nome	Cidade Natal	Nascimento	Melhor lugar
0	Maria Betania	Salvador	1946-08-18	NaN
1	Caetano Veloso	NaN	NaT	NaN
2	Djavan	Maceio	NaT	NaN

df.dropna()

Tabela vazia

df.dropna(subset=["Cidade Natal", "Nascimento"])

	Nome	Cidade Natal	Nascimento	Melhor lugar
0	Maria Betania	Salvador	1946-08-18	NaN

cin.ufpe.br

df.isnull().sum()

```
Nome            0
Cidade Natal    1
Nascimento      2
Melhor lugar    3
dtype: int64
```

	Nome	Cidade Natal	Nascimento	Melhor lugar
0	Maria Betania	Salvador	1946-08-18	NaN
1	Caetano Veloso	NaN	NaT	NaN
2	Djavan	Maceio	NaT	NaN

df.dropna(axis=1)

	Nome
0	Maria Betania
1	Caetano Veloso
2	Djavan

df.dropna(axis=1, how="all")

	Nome	Cidade Natal	Nascimento
0	Maria Betania	Salvador	1946-08-18
1	Caetano Veloso	NaN	NaT
2	Djavan	Maceio	NaT

Centro de Informática UFPE
Universidade Federal de Pernambuco

Encontrar as linhas que possuem dados faltantes

np.where(df["Melhor lugar"].isnull())

>>(array([0, 1, 2]),)

cin.ufpe.br

Quais linhas possuem Data de Lançamento nula?

np.where(filmes.release_date.isnull())

>>(array([266]),)

print(filmes.iloc[266])

inconsistente: precisamos apagar.

filmes = **filmes.drop(266)**

movie_id	267
movie_title	unknown
release_date	NaT
video_release_date	NaN
IMDb_URL	NaN
unknown	True
Action	False
Adventure	False
Animation	False
Children	False
Comedy	False
Crime	False
Documentary	False
Drama	False
Fantasy	False
FilmNoir	False
Horror	False
Musical	False
Mystery	False
Romance	False
SciFi	False
Thriller	False
War	False
Western	False

Preenchimento de valores ausentes

Podemos preencher todos os dados ausentes com um único valor.

Isso é feito em Pandas através da função fillna, tal como havíamos feito anteriormente quando extraímos o ano da data de lançamento e o convertemos para inteiro.

A função espera um valor que será assinalado ao atributo. Esse valor pode ser arbitrário ou uma medida de tendência computada sobre os dados presentes. Por exemplo, podemos preencher a data de lançamento do filme descoberto acima com a moda.

DataFrame.**fillna**(*value=None, method=None, axis=None, inplace=False, limit=None, downcast=None*)

Colocando valores em determinada linha

Se não tivéssemos apagado a linha anterior 266...

```
filmes.release_date = filmes.release_date.fillna(value=filmes.release_date.mode()[0])
```


Colocando valores em todas as colunas

```
filmes.video_release_date.fillna(value="1990-10-10", inplace=True)
```

	movie_id	movie_title	release_date	video_release_date	IMDb_URL	unknown	Action	Adventure	Animation	Children	...	Fantasy	Fil
0	1	Toy Story (1995)	01-Jan-1995	1990-10-10	http://us.imdb.com/M/title-exact?Toy%20Story%2...	0	0	0	1	1	...	0	
1	2	GoldenEye (1995)	01-Jan-1995	1990-10-10	http://us.imdb.com/M/title-exact?GoldenEye%20(...	0	1	1	0	0	...	0	
2	3	Four Rooms (1995)	01-Jan-1995	1990-10-10	http://us.imdb.com/M/title-exact?Four%20Rooms%...	0	0	0	0	0	...	0	
3	4	Get Shorty (1995)	01-Jan-1995	1990-10-10	http://us.imdb.com/M/title-exact?Get%20Shorty%...	0	1	0	0	0	...	0	
4	5	Copycat (1995)	01-Jan-1995	1990-10-10	http://us.imdb.com/M/title-exact?Copycat%20(1995)	0	0	0	0	0	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1677	1678	Mat' i syn	06-Feb-1998	1990-10-10	http://us.imdb.com/M/title-	0	0	0	0	0		0	

Base de dados de vinhos:

```
wine =
pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv",
                    sep=';')
```

A	B	C	D	E	F	G	H	I	J	K	L
fixed acidity	volatile acidi	citric acid	residual suga	chlorides	free sulfur di	total sulfur d	density	pH	sulphates	alcohol	quality
7	0.27	0.36	20.7	0.045	45	170	1.001	3	0.45	8.8	6
6.3	0.3	0.34	1.6	0.049	14	132	0.994	3.3	0.49	9.5	6
8.1	0.28	0.4	6.9	0.05	30	97	0.9951	3.26	0.44	10.1	6
7.2	0.23	0.32	8.5	0.058	47	186	0.9956	3.19	0.4	9.9	6
7.2	0.23	0.32	8.5	0.058	47	186	0.9956	3.19	0.4	9.9	6
8.1	0.28	0.4	6.9	0.05	30	97	0.9951	3.26	0.44	10.1	6
6.2	0.32	0.16	7	0.045	30	136	0.9949	3.18	0.47	9.6	6
7	0.27	0.36	20.7	0.045	45	170	1.001	3	0.45	8.8	6
6.3	0.3	0.34	1.6	0.049	14	132	0.994	3.3	0.49	9.5	6
8.1	0.22	0.43	1.5	0.044	28	129	0.9938	3.22	0.45	11	6
8.1	0.27	0.41	1.45	0.033	11	63	0.9908	2.99	0.56	12	5
8.6	0.23	0.4	4.2	0.035	17	109	0.9947	3.14	0.53	9.7	5
7.9	0.18	0.37	1.2	0.04	16	75	0.992	3.18	0.63	10.8	5
6.6	0.16	0.4	1.5	0.044	48	143	0.9912	3.54	0.52	12.4	7
8.3	0.42	0.62	19.25	0.04	41	172	10.002	2.98	0.67	9.7	5
6.6	0.17	0.38	1.5	0.032	28	112	0.9914	3.25	0.55	11.4	7
6.3	0.48	0.04	1.1	0.046	30	99	0.9928	3.24	0.36	9.6	6
6.2	0.66	0.48	1.2	0.029	29	75	0.9892	3.33	0.39	12.8	8
7.4	0.34	0.42	1.1	0.033	17	171	0.9917	3.12	0.53	11.3	6
6.5	0.31	0.14	7.5	0.044	34	133	0.9955	3.22	0.5	9.5	5
6.2	0.66	0.48	1.2	0.029	29	75	0.9892	3.33	0.39	12.8	8


```
print(wine.info())
wine.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4898 entries, 0 to 4897
Data columns (total 12 columns):
fixed acidity           4898 non-null float64
volatile acidity        4898 non-null float64
citric acid             4898 non-null float64
residual sugar          4898 non-null float64
chlorides               4898 non-null float64
free sulfur dioxide     4898 non-null float64
total sulfur dioxide    4898 non-null float64
density                 4898 non-null float64
pH                      4898 non-null float64
sulphates               4898 non-null float64
alcohol                 4898 non-null float64
quality                 4898 non-null int64
dtypes: float64(11), int64(1)
memory usage: 459.3 KB
None
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

Normalização

Aplicar normalização min-max em todas as colunas:

$$v[i] = \frac{v[i] - min(v)}{max(v) - min(v)}(b - a) + a$$

```
wine.drop('quality',axis=1).apply(lambda x: (x-x.min())/(x.max()-x.min())).head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	0.307692	0.186275	0.216867	0.308282	0.106825	0.149826	0.373550	0.267785	0.254545	0.267442	0.129032
1	0.240385	0.215686	0.204819	0.015337	0.118694	0.041812	0.285383	0.132832	0.527273	0.313953	0.241935
2	0.413462	0.196078	0.240964	0.096626	0.121662	0.097561	0.204176	0.154039	0.490909	0.255814	0.338710
3	0.326923	0.147059	0.192771	0.121166	0.145401	0.156794	0.410673	0.163678	0.427273	0.209302	0.306452
4	0.326923	0.147059	0.192771	0.121166	0.145401	0.156794	0.410673	0.163678	0.427273	0.209302	0.306452

Centro de Informática UFPE
Universidade Federal de Pernambuco

Normalização

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler

scaler = MinMaxScaler()
pd.DataFrame(scaler.fit_transform(wine.drop('quality',axis=1)),
columns=wine.drop('quality',axis=1).columns)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	0.307692	0.186275	0.216867	0.308282	0.106825	0.149826	0.373550	0.267785	0.254545	0.267442	0.129032
1	0.240385	0.215686	0.204819	0.015337	0.118694	0.041812	0.285383	0.132832	0.527273	0.313953	0.241935
2	0.413462	0.196078	0.240964	0.096626	0.121662	0.097561	0.204176	0.154039	0.490909	0.255814	0.338710
3	0.326923	0.147059	0.192771	0.121166	0.145401	0.156794	0.410673	0.163678	0.427273	0.209302	0.306452
4	0.326923	0.147059	0.192771	0.121166	0.145401	0.156794	0.410673	0.163678	0.427273	0.209302	0.306452
5	0.413462	0.196078	0.240964	0.096626	0.121662	0.097561	0.204176	0.154039	0.490909	0.255814	0.338710
6	0.230769	0.235294	0.096386	0.098160	0.106825	0.097561	0.294664	0.150183	0.418182	0.290698	0.258065

```
wineCopy = wine.copy().drop('quality',axis=1)

print(wineCopy.head())

wineCopy.iloc[:,:] = scaler.fit_transform(wineCopy.drop('quality',axis=1)) #vetor do numpy como resultado
```

wineCopy.head()

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.0              0.27         0.36            20.7      0.045
1            6.3              0.30         0.34             1.6      0.049
2            8.1              0.28         0.40             6.9      0.050
3            7.2              0.23         0.32             8.5      0.058
4            7.2              0.23         0.32             8.5      0.058

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 45.0                 170.0   1.0010  3.00       0.45
1                 14.0                 132.0   0.9940  3.30       0.49
2                 30.0                  97.0   0.9951  3.26       0.44
3                 47.0                 186.0   0.9956  3.19       0.40
4                 47.0                 186.0   0.9956  3.19       0.40

   alcohol
0      8.8
1      9.5
2     10.1
3      9.9
4      9.9
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	0.307692	0.186275	0.216867	0.308282	0.106825	0.149826	0.373550	0.267785	0.254545	0.267442	0.129032
1	0.240385	0.215686	0.204819	0.015337	0.118694	0.041812	0.285383	0.132832	0.527273	0.313953	0.241935
2	0.413462	0.196078	0.240964	0.096626	0.121662	0.097561	0.204176	0.154039	0.490909	0.255814	0.338710
3	0.326923	0.147059	0.192771	0.121166	0.145401	0.156794	0.410673	0.163678	0.427273	0.209302	0.306452
4	0.326923	0.147059	0.192771	0.121166	0.145401	0.156794	0.410673	0.163678	0.427273	0.209302	0.306452

Normalização Z-Score

$$v[i] = \frac{v[i] - \bar{v}}{\sigma_v}$$

```
#Solucao 1

wine.drop('quality',axis=1).apply(lambda x: (x-x.mean())/x.std()).describe()

#Solucao 2

wineCopy = wine.copy().drop('quality',axis=1)

scaler = StandardScaler().fit(wineCopy)

wineCopy.iloc[:,:] = scaler.fit_transform(wineCopy)

wineCopy.describe()
```

Salvar o Dataframe

DataFrame.**to_csv**(*path_or_buf=None, sep=',', na_rep='', float_format=None, columns=None, header=True, index=True, index_label=None, mode='w', encoding=None, compression='infer', quoting=None, quotechar='"', line_terminator=None, chunksize=None, date_format=None, doublequote=True, escapechar=None, decimal='.', errors='strict', storage_options=None*)

- **Salvando...**
  filmes.to_csv("filmes_aulas.csv",sep="-",header=False)
- **Lendo...**

```
filmes2 = pd.read_csv("filmes_aulas.csv",

    sep='-',header=None, names=["movie_id", "movie_title",  "release_date", "video_release_date", "IMDb_URL",
"unknown", "Action", "Adventure", "Animation",

        "Children", "Comedy", "Crime", "Documentary", "Drama", "Fantasy","FilmNoir", "Horror", "Musical", "Mystery",
"Romance", "SciFi","Thriller", "War", "Western"],

    encoding='latin1')
```

- **Carregando...**

```
filmes.to_csv("filmes_aulas.csv",sep="-")
```

- **Lendo...**

```
filmes2 = pd.read_csv("filmes_aulas.csv",
    sep='-',header=0,
    encoding='latin1')
```


Centro de Informática UFPE
Universidade Federal de Pernambuco

Curiosidade

```
#Podemos ler as 20 primeiras linhas de um arquivo com o Pandas
import pandas as pd
z = pd.read_csv("nome_arquivo.csv", nrows=20)
```

cin.ufpe.br