



FUNDAMENTOS E TÉCNICAS EM CIÊNCIAS DE DADOS

PROF. JOSENALDE OLIVEIRA

josenalde@eaj.ufrn.br

<https://github.com/josenalde/datascience>

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - UFRN

PANDAS #3 – TRATAMENTO/LIMPEZA DE DADOS

- LER ARQUIVOS .CSV COM DIFERENTES CODIFICAÇÕES
- LIMPAR/ALTERAR NOME DE COLUNAS
- CONVERTER UMA COLUNA STRING PARA NUMÉRICA
- EXTRAIR VALORES DOS EXTREMOS DE UMA STRING
- CORRIGIR VALORES INCORRETOS E DELETAR VALORES FALTANTES

Vamos utilizar como exemplo, um notebook de descrição de Laptops, que possui a característica geral abaixo:

```
1 df.dtypes
Manufacturer      object
Model Name        object
Category          object
Screen Size       object
Screen            object
CPU               object
RAM              object
Storage           object
GPU              object
Operating System  object
Operating System Version object
Weight            object
Price (Euros)     object
dtype: object
```

	Manufacturer	Model Name	Category	Screen Size	Screen	CPU	RAM	Storage	GPU	Operating System	Operating System Version	Weight	Price (Euros)
0	Apple	MacBook Pro	Ultrabook	13.3"	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	NaN	1.37kg	1339,69
1	Apple	Macbook Air	Ultrabook	13.3"	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	NaN	1.34kg	898,94
2	HP	250 G6	Notebook	15.6"	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	NaN	1.86kg	575,00
3	Apple	MacBook Pro	Ultrabook	15.4"	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	NaN	1.83kg	2537,45
4	Apple	MacBook Pro	Ultrabook	13.3"	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS	NaN	1.37kg	1803,60

PANDAS #3 – TRATAMENTO/LIMPEZA DE DADOS

Embora seja possível definir o tipo do dado quando da leitura E o indicador de separador de decimais, os dados possuem caracteres de unidade/medida:

```
1 def mod_cols(col):
2     #remove espaços em branco ao redor
3     col = col.strip()
4     #remove parenteses
5     col = col.replace('(', '')
6     col = col.replace(')', '')
7     #passa para minúsculo
8     col = col.lower()
9     #preenche espaço entre palavras com _
10    col = col.replace(' ', '_')
11    return col
12
13 df.columns = [mod_cols(c) for c in df.columns]
14 df.columns
15 #df.columns.tolist()
```

```
Index(['manufacturer', 'model_name', 'category', 'screen_size', 'screen',
      'cpu', 'ram', 'storage', 'gpu', 'operating_system',
      'operating_system_version', 'weight', 'price_euros'],
      dtype='object')
```

1	df.dtypes
	Manufacturer object
	Model Name object
	Category object
→	Screen Size object
	Screen object
	CPU object
	RAM object
→	Storage object
	GPU object
	Operating System object
	Operating System Version object
→	Weight object
→	Price (Euros) object
	dtype: object

Algumas ações sugeridas:

- Remover espaços em branco ‘ao redor’ do nome da coluna
- Remover os parênteses ao redor de **Euros**
- Remover a unidade kg de Weight
- Remover a unidade polegadas de Screen Size

PANDAS #3 – TRATAMENTO/LIMPEZA DE DADOS

Algumas ações sugeridas:

- ~~Remover espaços em branco ‘ao redor’ do nome da coluna~~
- ~~Remover os parênteses ao redor de Euros~~
- Remover a unidade kg de Weight e transformar em número float
- Remover a unidade polegadas de Screen Size

```
1 df['screen_size']
```

0	13.3
1	13.3
2	15.6
3	15.4
4	13.3
	...
1298	14.0
1299	13.3
1300	14.0
1301	15.6
1302	15.6

Name: screen_size, Length: 1303, dtype: float64

```
df['screen_size'] = (df['screen_size'].str.replace(' ', ''))
```

```
df['screen_size'] = df['screen_size'].astype(float)
```

```
df['screen_size'] = (df['screen_size'].str.replace(' ', '').astype(float))
```



```
1 df['weight'] = df['weight'].str.replace('kg', '')
2 df['weight'] = df['weight'].str.replace('s', '').astype(float)
3 df['weight']
```

```
1 def mod_weight(w):
2     w = w.strip()
3     #alguns lugares tem kg, noutros kgs
4     w = w.replace('kg', '')
5     w = w.replace('s', '')
6     w = float(w)
7     return w
8
9 df['weight'] = [mod_weight(w) for w in df['weight']]
10 df['weight']
```

```
0    1.37
1    1.34
2    1.86
3    1.83
4    1.37
```

PANDAS #3 – TRATAMENTO STRINGS - SEPARAR

Pode ser necessário separar partes de strings no início, fim ou meio dos dados, inclusive para criar outras colunas com a parte separada. Exemplo, se quisermos separar o tamanho da mídia de armazenamento não volátil, do seu tipo (storage)

```
df['storage'].str.split(n=1)
```

Se não colocar o parâmetro n, separa por padrão por espaço em branco, todas as palavras

```
df['storage'].str.split(n=1, expand=True)
```

```
df_sto = df['storage'].str.split(n=1, expand=True)
```

```
df_sto.columns = ['A', 'B']
```

```
df.insert(8, 'storage_type', df_sto['B'])
```

Uma solução é criar novo DataFrame com esta separação, atribuir rótulos a estas colunas para facilitar a manipulação, e criar uma nova coluna do DataFrame original com esta informação separada

Screen	CPU	RAM	Storage
IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD
1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage
Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD
IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD
IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD

```
0      [128GB, SSD]
1  [128GB, Flash Storage]
2      [256GB, SSD]
3      [512GB, SSD]
4      [256GB, SSD]
...
1298      [128GB, SSD]
1299      [512GB, SSD]
1300  [64GB, Flash Storage]
1301      [1TB, HDD]
1302      [500GB, HDD]
Name: storage, Length: 1303, dtype: object
```

Expand

	0	1
0	128GB	SSD
1	128GB	Flash Storage
2	256GB	SSD
3	512GB	SSD
4	256GB	SSD
...
1298	128GB	SSD
1299	512GB	SSD
1300	64GB	Flash Storage
1301	1TB	HDD
1302	500GB	HDD

1303 rows × 2 columns

PANDAS #3 – TRATAMENTO STRINGS - SEPARAR

Exercício 1: com o uso do `rsplit`, criar nova coluna `screen_resolution`, a partir da coluna `screen`

Exercício 2: na coluna `storage` existem algumas linhas com dois tipos de armazenamento. Deseja-se separar, criando colunas `storage_capacity1`, `storage_type1`, `storage_capacity2`, `storage_type2`. Por exemplo, entre as linhas de índice 76 e 81, temos:

```
1 df.loc[76:81, 'storage']
```

76	2TB HDD
77	128GB SSD + 1TB HDD
78	1TB HDD
79	128GB SSD + 1TB HDD
80	256GB SSD
81	512GB SSD

Name: storage, dtype: object

Screen	CPU	RAM	Storage
IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD
1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage
Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD
IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD
IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD

PANDAS #3 – STRINGS

Funções de strings vetorizadas em Pandas

Datasets normalmente possuem colunas string despadronizadas, e também com valores nulos. Funções de mapeamento falham com valores ausentes. Vamos ver métodos de Series que são robustas à valores NaN

```
1 df = pd.Series(data=data)
```

```
1 df
```

```
pedro                pedro@gmail.com
josenalde  josenalde.oliveira@ufrn.br
david                david3@gmail.com
paulo                      NaN
dtype: object
```

```
1 df.str.contains('gmail')
```

```
pedro      True
josenalde  False
david      True
paulo      NaN
dtype: object
```

correspondência, na forma de uma lista

pad Adiciona espaços em branco à esquerda, à direita ou nos dois lados das strings

center Equivalente a `pad(side='both')`

repeat Duplica valores (por exemplo, `s.str.repeat(3)` é equivalente a `x * 3` para cada string)

replace Substitui ocorrências do padrão/da regex por outra string

slice Fatia cada string da Series

split Separa as string no delimitador ou na expressão regular

strip Remove espaços em branco de ambos os lados, incluindo quebras de linha

rstrip Remove espaços em branco do lado direito

lstrip Remove espaços em branco do lado esquerdo

```
import numpy as np
import pandas as pd
data = {'pedro': 'pedro@gmail.com', 'josenalde':
'josenalde.oliveira@ufrn.br', 'david': 'david3@gmail.com',
'paulo': np.nan}
```

Método	Descrição
cat	Concatena strings em todos os elementos, com um delimitador opcional
contains	Devolve um array booleano se cada string contiver um padrão/uma regex
count	Conta as ocorrências do padrão
extract	Utiliza uma expressão regular com grupos para extrair uma ou mais strings de uma Series de strings; o resultado será um DataFrame com uma coluna por grupo

endswith Equivalente a `x.endswith(pattern)` para cada elemento

startswith Equivalente a `x.startswith(pattern)` para cada elemento

findall Calcula uma lista com todas as ocorrências de um padrão/uma regex para cada string

get Indexa cada elemento (obtem o *i*-ésimo elemento)

isalnum Equivalente ao `str.alnum` embutido

isalpha Equivalente ao `str.isalpha` embutido

isdecimal Equivalente ao `str.isdecimal` embutido

isdigit Equivalente ao `str.isdigit` embutido

islower Equivalente ao `str.islower` embutido

isnumeric Equivalente ao `str.isnumeric` embutido

isupper Equivalente ao `str.isupper` embutido

join Junta strings em cada elemento da Series utilizando o separador especificado

len Calcula o tamanho de cada string

lower, upper Converte para letras minúsculas ou maiúsculas; equivalente a `x.lower()` ou a `x.upper()` para cada elemento

match Usa `re.match` com a expressão regular especificada em cada elemento, devolvendo os grupos com os quais houve um

PANDAS #3 – TRATAMENTO DE DADOS NAN (MISSING)

Um dos métodos mais utilizados é o `fillna`, onde podem ser passados valores constantes, operações entre os valores, mapeamentos etc. sendo bastante versátil

```
df2.fillna({1:0.5, 2: -1})
```

	0	1	2
0	0.667995	0.500000	-1.000000
1	2.033357	0.500000	-1.000000
2	-0.286754	0.500000	0.484688
3	-1.851530	0.500000	2.399583
4	-0.294887	-0.566266	2.266865
5	-1.610551	0.548917	-0.475550
6	-1.801883	-0.680614	-0.833749

```
1 df2.iloc[:4,1] = np.nan
2 df2.iloc[:2,2] = np.nan
3 df2
```

```
df2.fillna(0)
```

	0	1	2
0	0.667995	0.000000	0.000000
1	2.033357	0.000000	0.000000
2	-0.286754	0.000000	0.484688
3	-1.851530	0.000000	2.399583
4	-0.294887	-0.566266	2.266865
5	-1.610551	0.548917	-0.475550
6	-1.801883	-0.680614	-0.833749

```
1 df2.fillna(df2.loc[:,0].mean())
```

	0	1	2
0	0.667995	-0.449179	-0.449179
1	2.033357	-0.449179	-0.449179
2	-0.286754	-0.449179	0.484688
3	-1.851530	-0.449179	2.399583
4	-0.294887	-0.566266	2.266865
5	-1.610551	0.548917	-0.475550
6	-1.801883	-0.680614	-0.833749

PANDAS #3 – TRATAMENTO DE DADOS NAN (MISSING)

Algumas estratégias:

- Ignorar os registros (eliminar)
- Completar manualmente (não assegura padrão, nem sempre é fácil)
- Constante global para todos
- Usar medidas de tendência central (como médias) do atributo ou de outros atributos
- Usar o valor mais provável (regressão, inferência)

Podem trazer viés – tendencioso - drift