FUNDAMENTOS E TÉCNICAS EM CIÊNCIAS DE DADOS

PROF. JOSENALDE OLIVEIRA

josenalde.oliveira@eaj.ufrn.br
https://github.com/josenalde/datascience

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - UFRN

PANDAS #6 – UNINDO DATASETS

Combinando datasets com CONCAT. Por exemplo, os datasets do World Happiness de 2015, 2016, 2017

```
df_15 = pd.read_csv('../datasets/wh2015.csv')
df_16 = pd.read_csv('../datasets/wh2016.csv')
df_17 = pd.read_csv('../datasets/wh2017.csv')
```

Vamos supor nomes das colunas iguais nos datasets, mas também precisamos considerar nomes diferentes e como irá se comportar ao fazer o CONCAT. Para simplificar, iremos extrair apenas as 5 primeiras linhas de cada dataset, e criar uma coluna YEAR em cada um

	Country	Region	Happiness Rank	Happiness Score	Standard Error
0	Switzerland	Western Europe	1	7.587	0.03411
1	Iceland	Western Europe	2	7.561	0.04884
2	Denmark	Western Europe	3	7.527	0.03328
3	Norway	Western Europe	4	7.522	0.03880
4	Canada	North America	5	7.427	0.03553

```
df_16.head()
```

Country	Region	Happiness Rank	Happiness Score	Lower Confidence Interval

```
1 df_17.head()
```

	Country	Happiness.Rank	Happiness.Score	Whisker.high
0	Norway	1	7.537	7.594445

PANDAS #6 – UNINDO DATASETS

Combinando datasets com CONCAT. Por exemplo, os datasets do World Happiness de 2015, 2016, 2017

```
df_15.head()
head_15 = df_15.head()
head_15 = head_15.iloc[:,0:5]
head_15['year'] = 2015
head_15
```

	Country	Region	Happiness Rank	Happiness Score	Standard Error	year
0	Switzerland	Western Europe	1	7.587	0.03411	2015
1	Iceland	Western Europe	2	7.561	0.04884	2015
2	Denmark	Western Europe	3	7.527	0.03328	2015
3	Norway	Western Europe	4	7.522	0.03880	2015
4	Canada	North America	5	7.427	0.03553	2015

```
head_16 = df_16.head()
head_16 = head_16.iloc[:,0:5]
head_16['year'] = 2016
head_16
```

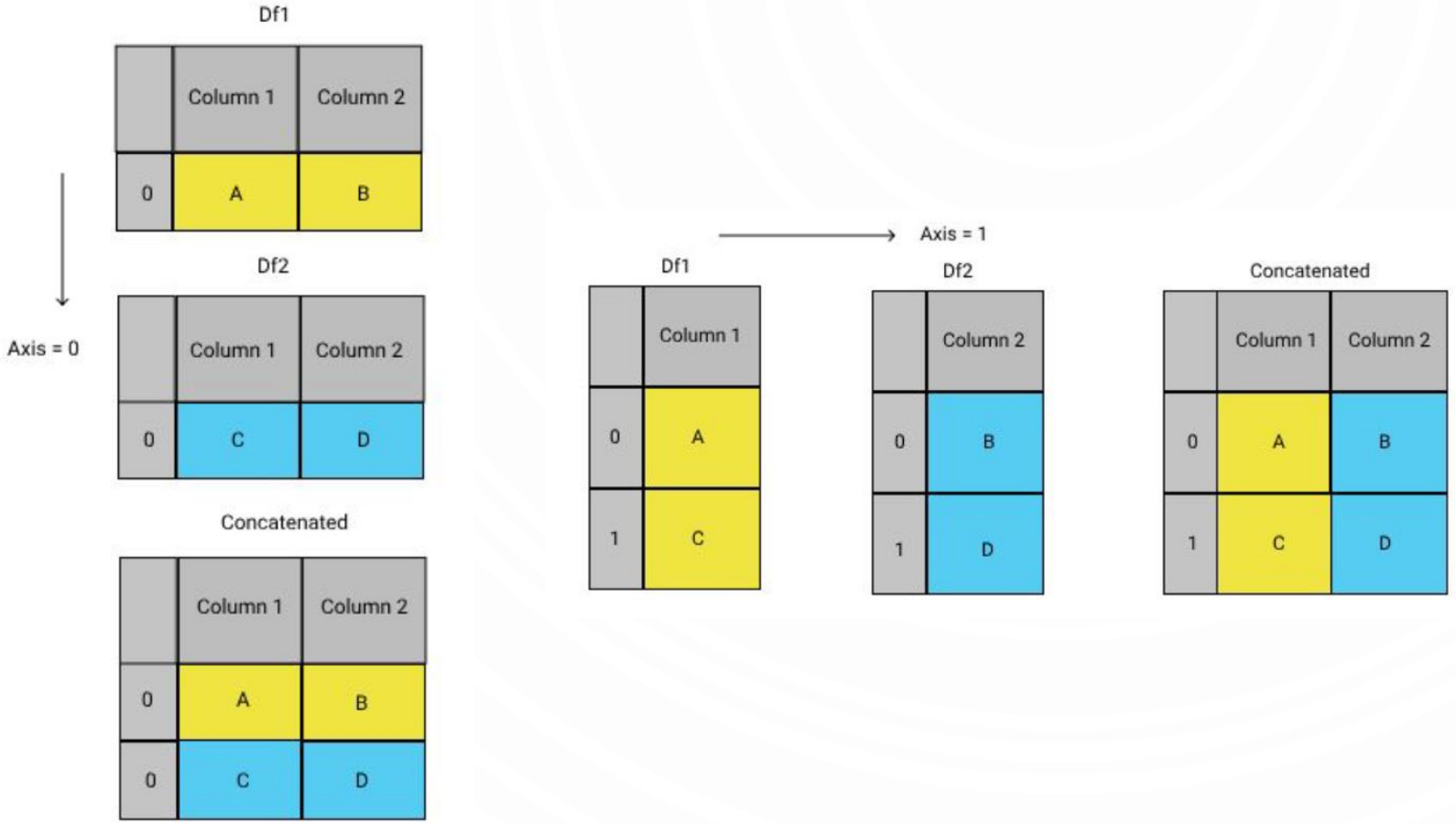
	Country	Region	Happiness Rank	Happiness Score	Lower Confidence Interval	year
0	Denmark	Western Europe	1	7.526	7.460	2016
1	Switzerland	Western Europe	2	7.509	7.428	2016
2	Iceland	Western Europe	3	7.501	7.333	2016
3	Norway	Western Europe	4	7.498	7.421	2016
4	Finland	Western Europe	5	7.413	7.351	2016

```
head_17 = df_17.head()
head_17 = head_17.iloc[:,0:5]
head_17['year'] = 2017
head_17
```

	Country	Happiness.Rank	Happiness.Score	Whisker.high	Whisker.low	year
0	Norway	1	7.537	7.594445	7.479556	2017
1	Denmark	2	7.522	7.581728	7.462272	2017
2	Iceland	3	7.504	7.622030	7.385970	2017
3	Switzerland	4	7.494	7.561772	7.426227	2017
4	Finland	5	7.469	7.527542	7.410458	2017

PANDAS #6 – UNINDO DATASETS

Combinando datasets com CONCAT. Pode ser registros (linhas) juntas (o mais comum), ou 'lado a lado'



PANDAS #6 – UNINDO DATASETS

Combinando datasets com CONCAT. Pode ser registros (linhas) juntas (o mais comum), ou 'lado a lado'

```
df_15_16 = pd.concat([head_15.iloc[:,[0,3,5]], head_16.iloc[:,[0,3,5]]], axis=0)
df_15_16
```

	Country	Happiness Score	year
0	Switzerland	7.587	2015
1	Iceland	7.561	2015
2	Denmark	7.527	2015
3	Norway	7.522	2015
4	Canada	7.427	2015
0	Denmark	7.526	2016
1	Switzerland	7.509	2016
2	Iceland	7.501	2016
3	Norway	7.498	2016
4	Finland	7.413	2016

```
df_15_16 = pd.concat([head_15.iloc[:,[0,3,5]], head_16.iloc[:,[0,3,5]]], axis=0, ignore_index=True)
df_15_16
```

	Country	Happiness Score	year
0	Switzerland	7.587	2015
1	Iceland	7.561	2015
2	Denmark	7.527	2015
3	Norway	7.522	2015
4	Canada	7.427	2015
5	Denmark	7.526	2016
6	Switzerland	7.509	2016
7	Iceland	7.501	2016
8	Norway	7.498	2016
9	Finland	7.413	2016

```
df_15_16_h = pd.concat([head_15.iloc[:,[0,3,5]], head_16.iloc[:,[0,3,5]]], axis=1)
df_15_16_h
```

	Country	Happiness Score	year	Country	Happiness Score	year
0	Switzerland	7.587	2015	Denmark	7.526	2016
1	Iceland	7.561	2015	Switzerland	7.509	2016
2	Denmark	7.527	2015	Iceland	7.501	2016
3	Norway	7.522	2015	Norway	7.498	2016
4	Canada	7.427	2015	Finland	7.413	2016

PANDAS #6 – UNINDO DATASETS

Caso haja a junção vertical com colunas não presentes – o Pandas atribui NaN

```
df_15_16_v = pd.concat([head_15.iloc[:,[0,3,4,5]], head_16.iloc[:,[0,3,5]]], axis=0, ignore_index=True)
df_15_16_v
```

	Country	Happiness Score	Standard Error	year
0	Switzerland	7.587	0.03411	2015
1	Iceland	7.561	0.04884	2015
2	Denmark	7.527	0.03328	2015
3	Norway	7.522	0.03880	2015
4	Canada	7.427	0.03553	2015
5	Denmark	7.526	NaN	2016
6	Switzerland	7.509	NaN	2016
7	Iceland	7.501	NaN	2016
8	Norway	7.498	NaN	2016
9	Finland	7.413	NaN	2016

	Year	Country	Happiness Score	Standard Error
0	2015	Switzerland	7.587	0.03411
1	2015	Iceland	7.561	0.04884
2	2015	Denmark	7.527	0.03328
3	2015	Norway	7.522	0.03880

	Country	Happiness Score	Year
0	Denmark	7.526	2016
1	Switzerland	7.509	2016
2	Iceland	7.501	2016

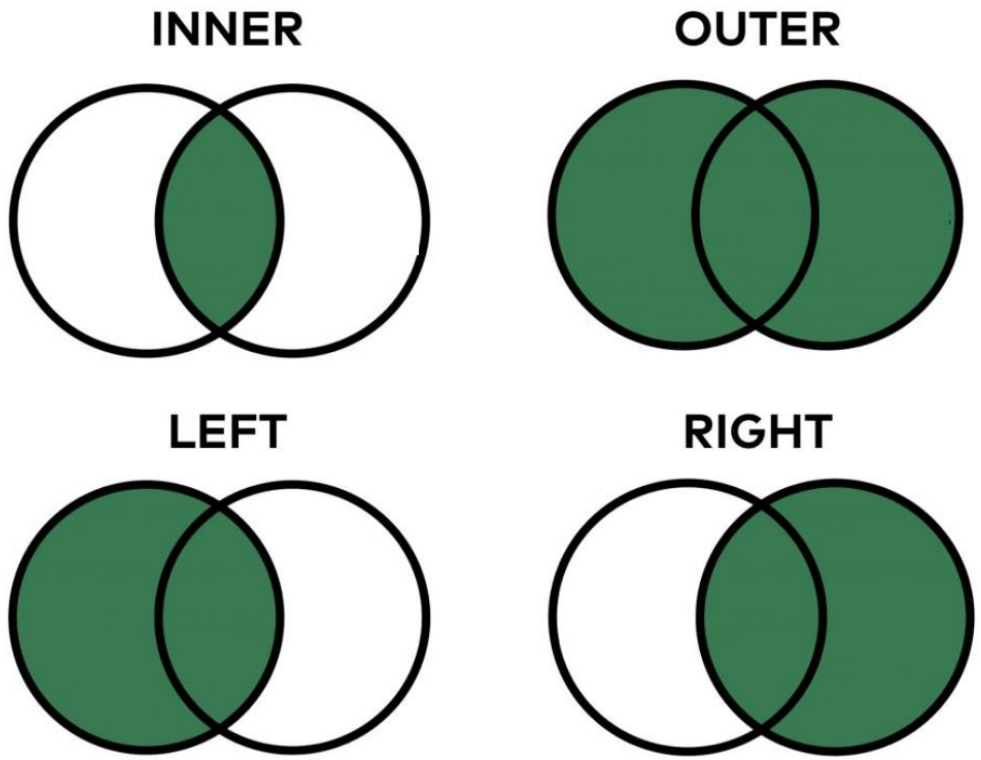
	Year	Country	Happiness Score	Standard Error	Country	Happiness Score	Year
0	2015	Switzerland	7.587	0.03411	Denmark	7.526	2016.0
1	2015	Iceland	7.561	0.04884	Switzerland	7.509	2016.0
2	2015	Denmark	7.527	0.03328	Iceland	7.501	2016.0
3	2015	Norway	7.522	0.03880	NaN	NaN	NaN

axis=1

PANDAS #6 – UNINDO DATASETS

Outro método interessante é o MERGE, que atua apenas horizontalmente (axis=1), com 2 dataframes por vez
É eficiente, interessante para grandes datasets – define flexibilidade em como os dados são combinados
Outer, Inner, Left, Right

	pd.concat()	pd.merge()
Default Join Type	Outer	Inner
Can Combine More Than Two Dataframes at a Time?	Yes	No
Can Combine Dataframes Vertically (axis=0) or Horizontally (axis=1)?	Both	Horizontally
Syntax	**Concat (Vertically)** concat([df1,df2,df3]) **Concat (Horizontally)** concat([df1,df2,df3], axis = 1)	**Merge (Join on Columns)** merge(left = df1, right = df2, how = 'join_type', on = 'Col') **Merge (Join on Index)** merge(left = df1, right = df2, how = 'join_type', left_index = True, right_index = True)



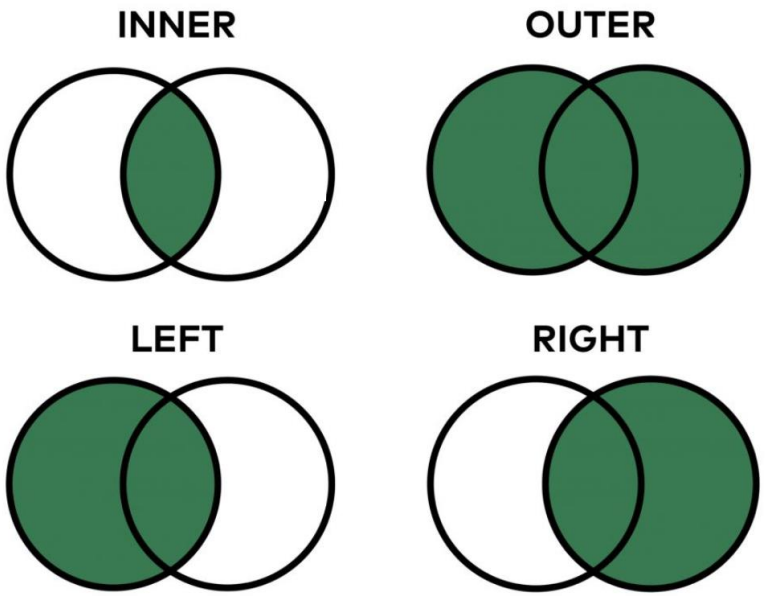
PANDAS #6 – UNINDO DATASETS

Outro método interessante é o MERGE, que atua apenas horizontalmente (axis=1), com 2 dataframes por vez
É eficiente, interessante para grandes datasets – define flexibilidade em como os dados são combinados
Outer, Inner, Left, Right

A cláusula `INNER JOIN` permite usar um operador de comparação para comparar os valores de colunas provenientes de tabelas associadas. Por meio desta cláusula, os registros de duas tabelas são usados para que sejam gerados os dados relacionados de ambas.

A cláusula `LEFT JOIN` permite obter não apenas os dados relacionados de duas tabelas, mas também os dados não relacionados encontrados na tabela à esquerda da cláusula `JOIN`. Caso não existam dados relacionados entre as tabelas à esquerda e a direita do `JOIN`, os valores resultantes de todas as colunas da lista de seleção da tabela à direita serão nulos.

Ao contrário do `LEFT JOIN`, a cláusula `RIGHT JOIN` retorna todos os dados encontrados na tabela à direita de `JOIN`. Caso não existam dados associados entre as tabelas à esquerda e à direita de `JOIN`, serão retornados valores nulos.



PANDAS #6 – UNINDO DATASETS

Outro método interessante é o MERGE, que atua apenas horizontalmente (axis=1), com 2 dataframes por vez
É eficiente, interessante para grandes datasets – define flexibilidade em como os dados são combinados
Outer, Inner, Left, Right

	Country	Happiness Rank	Year
2	Denmark	3	2015
3	Norway	4	2015
4	Canada	5	2015

	Country	Happiness Rank	Year
2	Iceland	3	2016
3	Norway	4	2016
4	Finland	5	2016

```
pd.merge(left=three_2015, right=three_2016, on="Country")
```

	Country	Happiness Rank_x	Year_x	Happiness Rank_y	Year_y
0	Norway	4	2015	4	2016

PANDAS #6 – UNINDO DATASETS

Outro método interessante é o MERGE, que atua apenas horizontalmente (axis=1), com 2 dataframes por vez
É eficiente, interessante para grandes datasets – define flexibilidade em como os dados são combinados
Outer, Inner, Left, Right

	Country	Happiness Rank	Year
2	Denmark	3	2015
3	Norway	4	2015
4	Canada	5	2015

	Country	Happiness Rank	Year
2	Iceland	3	2016
3	Norway	4	2016
4	Finland	5	2016

```
pd.merge(left=three_2015, right=three_2016,
         how='left', on='Country',suffixes=('_2015','_2016'))
```

	Country	Happiness Rank_2015	Year_2015	Happiness Rank_2016	Year_2016
0	Denmark	3	2015	NaN	NaN
1	Norway	4	2015	4.0	2016.0
2	Canada	5	2015	NaN	NaN

	Country	Happiness Rank	Year
2	Denmark	3	2015
3	Norway	4	2015
4	Canada	5	2015
5	Finland	6	2015

	Country	Happiness Rank	Year
2	Iceland	3	2016
3	Norway	4	2016
4	Finland	5	2016

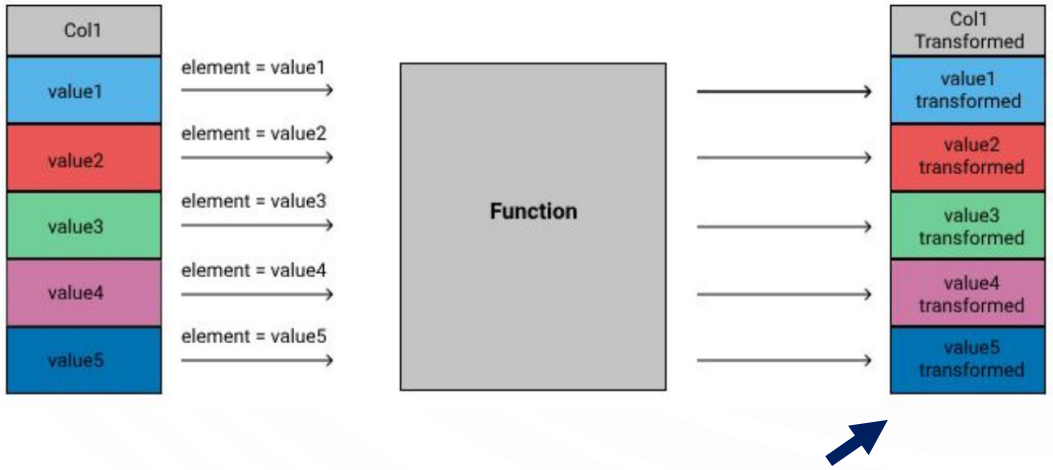
```
pd.merge(left = four_2015, right = three_2016,
         left_index = True, right_index = True,
         how='left',
         suffixes = ('_2015','_2016'))
```

	Country_2015	Happiness Rank_2015	Year_2015	Country_2016	Happiness Rank_2016	Year_2016
2	Denmark	3	2015	Iceland	3.0	2016.0
3	Norway	4	2015	Norway	4.0	2016.0
4	Canada	5	2015	Finland	5.0	2016.0
5	Finland	6	2015	NaN	NaN	NaN

PANDAS #6 – CRIANDO COLUNAS TRANSFORMADAS

Gerar transformação de colunas a partir de mapeamentos: apply, map, applymap, melt
No dataset em estudo, qual feature contribui mais para o Happiness Score (target variable)

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy	Family	Health	Freedom	Trust	Generosity
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811



PANDAS #6 – CRIANDO COLUNAS TRANSFORMADAS

Gerar transformação de colunas a partir de mapeamentos: apply, map, applymap, melt
No dataset em estudo, qual feature contribui mais para o Happiness Score (target variable)

```python
def label(element, x):
    if element > x:
        return 'High' # ou 1
    else:
        return 'Low' # ou 0

# para uma função com o valor de comparação já definido, map() funciona bem
'''
def label(element):
    if element > 1:
        return 1
    else:
        return 0
df_15['economy_impact'] = df_15['economy'].map(label)
'''

df_15['economy_impact'] = df_15['economy'].apply(label, x = 1.1)
```

```python
df_15
```

	Country	Region	Happiness Rank	Happiness Score	Standard Error	economy	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual	economy_impact
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738	High
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.70201	High

Para aplicar em múltiplas colunas, substituindo seus valores numéricos por variáveis dummy (0/1):

PANDAS #6 – CRIANDO COLUNAS TRANSFORMADAS

Gerar transformação de colunas a partir de mapeamentos: apply, map, applymap, melt
No dataset em estudo, qual feature contribui mais para o Happiness Score (target variable)

```
df_15.rename(columns={'Family': 'family', 'Health (Life Expectancy)': 'health', 'Freedom': 'freedom',
                      'Trust (Government Corruption)': 'trust', 'Generosity': 'generosity'},inplace=True)
factors = ['economy', 'family', 'health', 'freedom', 'trust', 'generosity']
```

```
factors_impact = df_15[factors].applymap(label2)
```

```
factors_impact
```

	economy	family	health	freedom	trust	generosity
0	1	1	0	0	0	0
1	1	1	0	0	0	0
2	1	1	0	0	0	0
3	1	1	0	0	0	0
4	1	1	0	0	0	0
...	...	...	...	...	...	...
153	0	0	0	0	0	0
154	0	0	0	0	0	0
155	0	0	0	0	0	0
156	0	0	0	0	0	0
157	0	0	0	0	0	0

Para aplicar em múltiplas colunas, substituindo seus valores numéricos por variáveis dummy (0/1):

```
factors_impact.apply(pd.value_counts)
```

	economy	family	health	freedom	trust	generosity
0	92	69	156	158.0	158.0	158.0
1	66	89	2	NaN	NaN	NaN

```
def v_counts(col):
    n = col.value_counts()
    d = col.size
    return n/d

factors_impact.apply(v_counts)
```

	economy	family	health	freedom	trust	generosity
0	0.582278	0.436709	0.987342	1.0	1.0	1.0
1	0.417722	0.563291	0.012658	NaN	NaN	NaN

PANDAS #6 – CRIANDO COLUNAS TRANSFORMADAS

Gerar transformação de colunas a partir de mapeamentos: apply, map, applymap, melt
No dataset em estudo, qual feature contribui mais para o Happiness Score (target variable)

```
1 top_two = df_15.head(2)
2 top_two
```

	Country	Region	Happiness Rank	Happiness Score	Standard Error	economy	family	health	freedom	trust	generosity	Dystopia Residual	economy_impact
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738	High
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.70201	High

```
1 pd.melt(top_two, id_vars=['Country'], value_vars=['economy', 'family', 'health'])
```

	Country	variable	value
0	Switzerland	economy	1.39651
1	Iceland	economy	1.30232
2	Switzerland	family	1.34951
3	Iceland	family	1.40223
4	Switzerland	health	0.94143
5	Iceland	health	0.94784