

Aula: Pré-processamento de Dados

IMD1151 - Ciência de Dados

Prof. Heitor Florencio
heitorm@imd.ufrn.br

Prof. Daniel Sabino
daniel@imd.ufrn.br

Aula: Pré-processamento de Dados

IMD1151 - Ciência de Dados

Prof. Heitor Florencio

heitorm@imd.ufrn.br

- Revisão: exploração de dados
- Pré-processamento de dados
- Limpeza de dados:
 - Tratamento de valores ausentes
 - Tratamento de valores duplicados
 - Tratamento de outliers

Exploração de dataset em Python:

seleção, sumarização, agrupamento,
ordenação, indexação, amostragem

Case: dataset de jogos de futebol feminino

Women's International Football Results

An up-to-date dataset of over 7,000 international football results



Data Card Code (21) Discussion (0) Suggestions (0)

About Dataset

Context

This is a work-in-progress sister data set to the [men's international football results dataset](#). If you're interested in helping out, submit a pull request [here](#).

Usability ⓘ

10.00

License

[CC0: Public Domain](#)

Expected update frequency

Dataset no Kaggle:

<https://www.kaggle.com/datasets/martj42/womens-international-football-results>

Dataset exemplo

```
import pandas as pd
```

```
df = pd.read_csv('results.csv')
df.head()
```

	date	home_team	away_team	home_score	away_score	tournament	city	country	neutral
0	1969-11-01	Italy	France	1	0	Euro	Novara	Italy	False
1	1969-11-01	Denmark	England	4	3	Euro	Aosta	Italy	True
2	1969-11-02	England	France	2	0	Euro	Turin	Italy	True
3	1969-11-02	Italy	Denmark	3	1	Euro	Turin	Italy	False
4	1970-07-06	England	West Germany	5	1	World Cup	Genova	Italy	True



Explorar dados: seleção com `.shape`

```
df.shape
```

```
(7731, 9)
```

```
list(df)
```

```
['date',  
 'home_team',  
 'away_team',  
 'home_score',  
 'away_score',  
 'tournament',  
 'city',  
 'country',  
 'neutral']
```

Explorar dados: seleção com `.unique()`

```
df.nunique()
```

```
date          2696
home_team     229
away_team     228
home_score    23
away_score    21
tournament    76
city          1392
country       190
neutral        2
dtype: int64
```

```
df['tournament'].unique()
```

```
array(['Euro', 'World Cup', 'Friendly', 'Nordic Championship',
      'AFC Championship', 'Chunghua Cup', 'Mundialito',
      'UEFA Euro qualification', 'OFC Championship', 'UEFA Euro',
      'Southeast Asian Games', 'FIFA Women's Invitation Tournament',
      'Asian Games', 'African Championship', 'CONCACAF Championship',
      'Copa América', 'FIFA World Cup',
      'CONCACAF Invitational Tournament', 'Algarve Cup', 'Baltic Cup',
      'Olympic Games', 'Four Nations Tournament',
      'African Championship qualification',
      'CONCACAF Gold Cup qualification', 'Goodwill Games',
      'Australian Cup', 'Pan American Games', 'CONCACAF Gold Cup',
      'Island Games', 'Central American Games', 'Pacific Games',
      'OFC Olympic Qualifying Tournament',
      'AFC Olympic Qualifying Tournament', 'AFC Asian Cup qualification',
      'Arab Championship', 'AFC Asian Cup', 'Peace Queen Cup',
      'African Games', 'Cyprus Cup', 'FIFA World Cup qualification',
      'International Tournament', 'South Asian Games',
      'Central American and Caribbean Games', 'Matchworld Women's Cup',
      'Olympic Games qualification', 'Kirin Challenge Cup', 'Slavic Cup',
      'Valais Women's Cup', 'Istria Cup',
      'CONCACAF Championship qualification', 'Aphrodite Cup',
      'MS&AD Cup', 'Indian Ocean Island Games',
      'Yongchuan International Tournament', 'SheBelieves Cup',
      'African Cup of Nations qualification', 'African Cup of Nations',
      'Tournament of Nations', 'Turkish Women's Cup',
      'OFC Nations Cup qualification', 'OFC Nations Cup',
      'Gold Cup India', 'Cup of Nations', 'Tournoi de France',
      'Pinatar Cup', 'Armenia International Friendly Tournament',
      'Arab Cup', 'Aisha Buhari Cup', 'Malta International Tournament',
      'Arnold Clark Cup', 'Women's Revelations Cup',
      'SAFF Women's Friendly Tournament', 'Finalissima',
      'CAF Olympic Qualifying Tournament', 'UEFA Nations League',
      'Olympic qualification'], dtype=object)
```

Explorar dados: seleção

```
df[(df['country'] == 'Brazil') & (df['city'] == 'Rio de Janeiro')]
```

	date	home_team	away_team	home_score	away_score	tournament	city	country	neutral
2630	2007-07-12	Ecuador	Jamaica	0	1	Pan American Games	Rio de Janeiro	Brazil	True
2631	2007-07-12	Brazil	Uruguay	4	0	Pan American Games	Rio de Janeiro	Brazil	False
2632	2007-07-12	Argentina	Panama	2	0	Pan American Games	Rio de Janeiro	Brazil	True
2633	2007-07-14	Uruguay	Canada	0	7	Pan American Games	Rio de Janeiro	Brazil	True
2634	2007-07-14	Brazil	Jamaica	5	0	Pan American Games	Rio de Janeiro	Brazil	False
2635	2007-07-14	Paraguay	Mexico	0	5	Pan American Games	Rio de Janeiro	Brazil	True
2638	2007-07-16	Jamaica	Uruguay	1	1	Pan American Games	Rio de Janeiro	Brazil	True
2639	2007-07-16	Canada	Ecuador	4	0	Pan American Games	Rio de Janeiro	Brazil	True
2640	2007-07-16	Panama	Paraguay	1	1	Pan American Games	Rio de Janeiro	Brazil	True

Explorar dados: seleção com `.loc` e `.iloc`

```
df.loc[df['country'] == 'Brazil'].head()
```

	date	home_team	away_team	home_score	away_score	tournament	city	country	neutral
550	1991-04-28	Brazil	Chile	6	1	Copa América	Maringá	Brazil	False
552	1991-05-01	Chile	Venezuela	1	0	Copa América	Maringá	Brazil	True
554	1991-05-05	Brazil	Venezuela	6	11	Copa América	Maringá	Brazil	False
815	1995-01-08	Brazil	Ecuador	13	0	Copa América	Uberlândia	Brazil	False
816	1995-01-08	Chile	Bolivia	11	0	Copa América	Uberlândia	Brazil	True

Explorar dados: seleção com .slice() (em string)

```
df[(df['country'] == 'Brazil') & (df['date'].str.slice(start=0, stop=4, step=1) == '2007')]
```

	date	home_team	away_team	home_score	away_score	tournament	city	country	neutral
2630	2007-07-12	Ecuador	Jamaica	0	1	Pan American Games	Rio de Janeiro	Brazil	True
2631	2007-07-12	Brazil	Uruguay	4	0	Pan American Games	Rio de Janeiro	Brazil	False
2632	2007-07-12	Argentina	Panama	2	0	Pan American Games	Rio de Janeiro	Brazil	True
2633	2007-07-14	Uruguay	Canada	0	7	Pan American Games	Rio de Janeiro	Brazil	True
2634	2007-07-14	Brazil	Jamaica	5	0	Pan American Games	Rio de Janeiro	Brazil	False
2635	2007-07-14	Paraguay	Mexico	0	5	Pan American Games	Rio de Janeiro	Brazil	True
2638	2007-07-16	Jamaica	Uruguay	1	1	Pan American Games	Rio de Janeiro	Brazil	True



Explorar dados: seleção com `.slice()` (em string)

```
df['date'].str.slice(start=0,
                     stop=4,
                     step=1)
```

0	1969
1	1969
2	1969
3	1969
4	1970
...	
7726	2024
7727	2024
7728	2024
7729	2024
7730	2024

```
df['date']
```

0	1969-11-01
1	1969-11-01
2	1969-11-02
3	1969-11-02
4	1970-07-06
...	
7726	2024-02-28
7727	2024-02-28
7728	2024-02-28
7729	2024-02-28
7730	2024-02-28

Explorar dados: **sumarização com .count() e nunique()**

- quantos jogos que aconteceram no Brasil?

```
df[df['country'] == 'Brazil']['country'].count()
```

```
136
```

- quantas cidades já sediaram jogos no Brasil?

```
df[df['country'] == 'Brazil']['city'].nunique()
```

```
9
```

Explorar dados: **sumarização com .sum()**

- jogando em casa, quantos gols o Brasil acumulou?

```
df[(df['country'] == 'Brazil') & (df['home_team'] == 'Brazil')]['home_score'].sum()
```

216

- jogando em casa, quantos gols o Brasil marca em média e mediana?

```
df[(df['country'] == 'Brazil')
    & (df['home_team'] == 'Brazil')]['home_score'].mean()
```

3.789473684210526

```
df[(df['country'] == 'Brazil')
    & (df['home_team'] == 'Brazil')]['home_score'].median()
```

3.0

Explorar dados: agrupamento com .groupby()

- quantos jogos por país?

```
df.groupby('country')['country'].count()
```

```
country
Albania      33
Algeria      24
Andorra       3
Angola        6
Anguilla      3
..
```

```
Venezuela    15
Vietnam      95
Wales        54
Zambia       15
Zimbabwe     15
```

```
Name: country, Length: 190, dtype: int64
```

Explorar dados: agrupamento multinível com .groupby()

- quantos jogos por país/cidade?

```
df.groupby(['country', 'city'])['country'].count()
```

country	city	
Albania	Durres	1
	Durrës	1
	Elbasan	13
	Fier	2
	Shkodër	5
	..	
Zambia	Lusaka	11
	Ndola	1
	Zambia	2
Zimbabwe	Harare	14
	Zimbabwe	1

Name: country, Length: 1428, dtype: int64

Explorar dados: ordenação com `.sort_values()`

```
df.sort_values(by = 'home_score', ascending = False)
```

	date	home_team	away_team	home_score	away_score	tournament	city	country	neutral
1628	2001-12-06	North Korea	Singapore	24	0	AFC Championship	New Taipei City	Taiwan	True
888	1995-09-24	China PR	Philippines	21	0	AFC Championship	Kota Kinabalu	Malaysia	True
1107	1997-12-05	Japan	Guam	21	0	AFC Championship	Guangdong	China PR	True
1233	1998-12-09	New Zealand	Samoa	21	0	OFC Championship	Auckland	New Zealand	False
4155	2013-06-07	Jordan	Kuwait	21	0	AFC Asian Cup qualification	Amman	Jordan	False
...

Explorar dados: indexação com .set_index()

```
df.set_index(['date'], inplace=True) # inplace é para persistir no dataframe
df.head()
```

	home_team	away_team	home_score	away_score	tournament	city	country	neutral
date								
1969-11-01	Italy	France	1	0	Euro	Novara	Italy	False
1969-11-01	Denmark	England	4	3	Euro	Aosta	Italy	True
1969-11-02	England	France	2	0	Euro	Turin	Italy	True
1969-11-02	Italy	Denmark	3	1	Euro	Turin	Italy	False
1970-07-06	England	West Germany	5	1	World Cup	Genova	Italy	True



Explorar dados: indexação com `.reset_index()`

```
df.reset_index(inplace=True)
df.head()
```

	date	home_team	away_team	home_score	away_score	tournament	city	country	neutral
0	1969-11-01	Italy	France	1	0	Euro	Novara	Italy	False
1	1969-11-01	Denmark	England	4	3	Euro	Aosta	Italy	True
2	1969-11-02	England	France	2	0	Euro	Turin	Italy	True
3	1969-11-02	Italy	Denmark	3	1	Euro	Turin	Italy	False
4	1970-07-06	England	West Germany	5	1	World Cup	Genova	Italy	True

Explorar dados: amostragem com `.sample()`

```
df.sample(frac=0.1).shape
```

(773, 9)

```
df.sample(frac=0.1)
```

[illegible]

Series

DataFrame ▾

- [pandas.DataFrame](#)
- [pandas.DataFrame.index](#)
- [pandas.DataFrame.columns](#)
- [pandas.DataFrame.dtypes](#)
- [pandas.DataFrame.info](#)
- [pandas.DataFrame.select_dtypes](#)
- [pandas.DataFrame.values](#)
- [pandas.DataFrame.axes](#)
- [pandas.DataFrame.ndim](#)
- [pandas.DataFrame.size](#)
- [pandas.DataFrame.shape](#)
- [pandas.DataFrame.memory_usage](#)
- [pandas.DataFrame.empty](#)
- [pandas.DataFrame.set_flags](#)
- [pandas.DataFrame.astype](#)
- [pandas.DataFrame.convert_dtypes](#)

🏠 > API reference > **DataFrame**

DataFrame

Constructor

`DataFrame` `((data, index, columns, dtype, copy))` Two-dimensional, size-mutable, potentially heterogeneous tabular data.

Attributes and underlying data

Axes

`DataFrame.index` The index (row labels) of the DataFrame.

`DataFrame.columns` The column labels of the DataFrame.

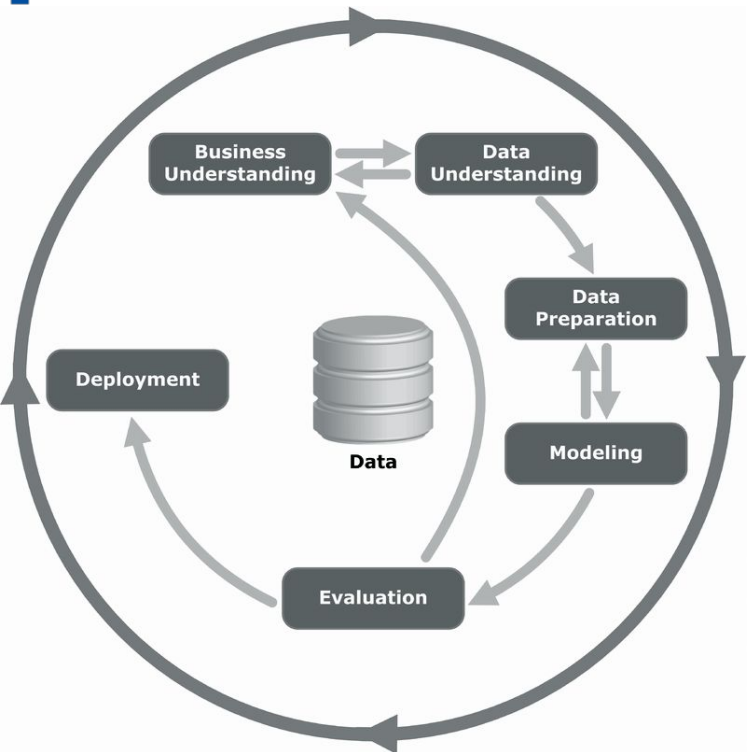
☰ On this page

Constructor

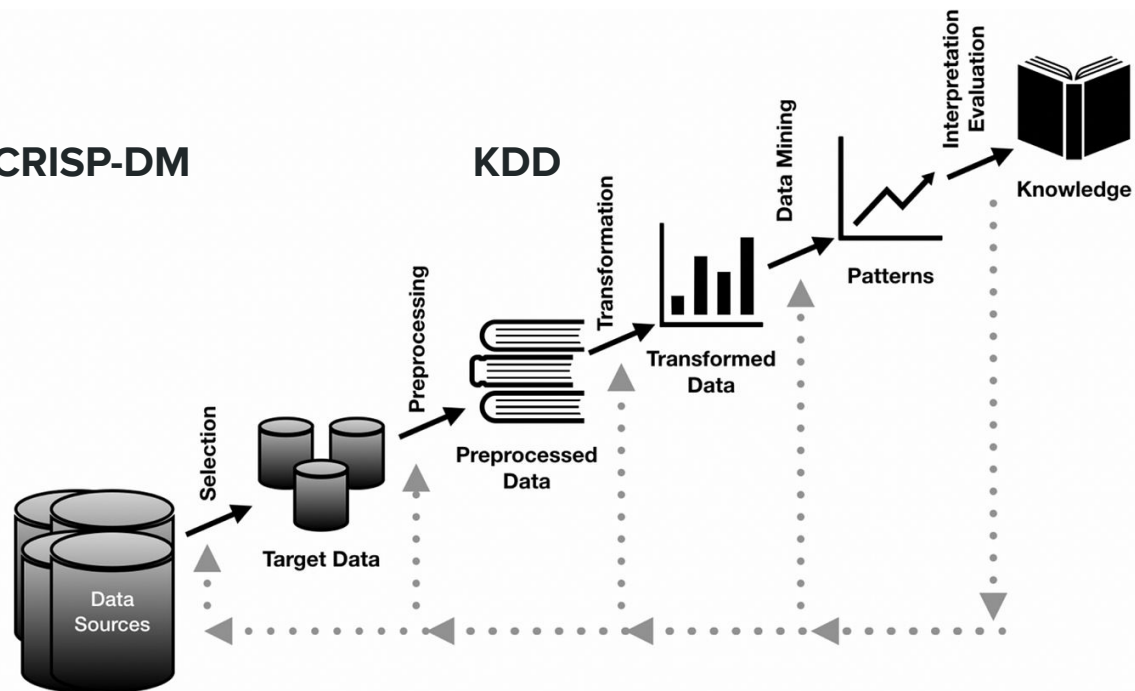
- Attributes and underlying data
- Conversion
- Indexing, iteration
- Binary operator functions
- Function application, GroupBy & window
- Computations / descriptive stats
- Reindexing / selection / label manipulation
- Missing data handling
- Reshaping, sorting, transposing
- Combining / comparing / joining / merging
- Time Series-related
- Flags
- Metadata

Pré-processamento de Dados

Modelos de Processos em Ciência de Dados



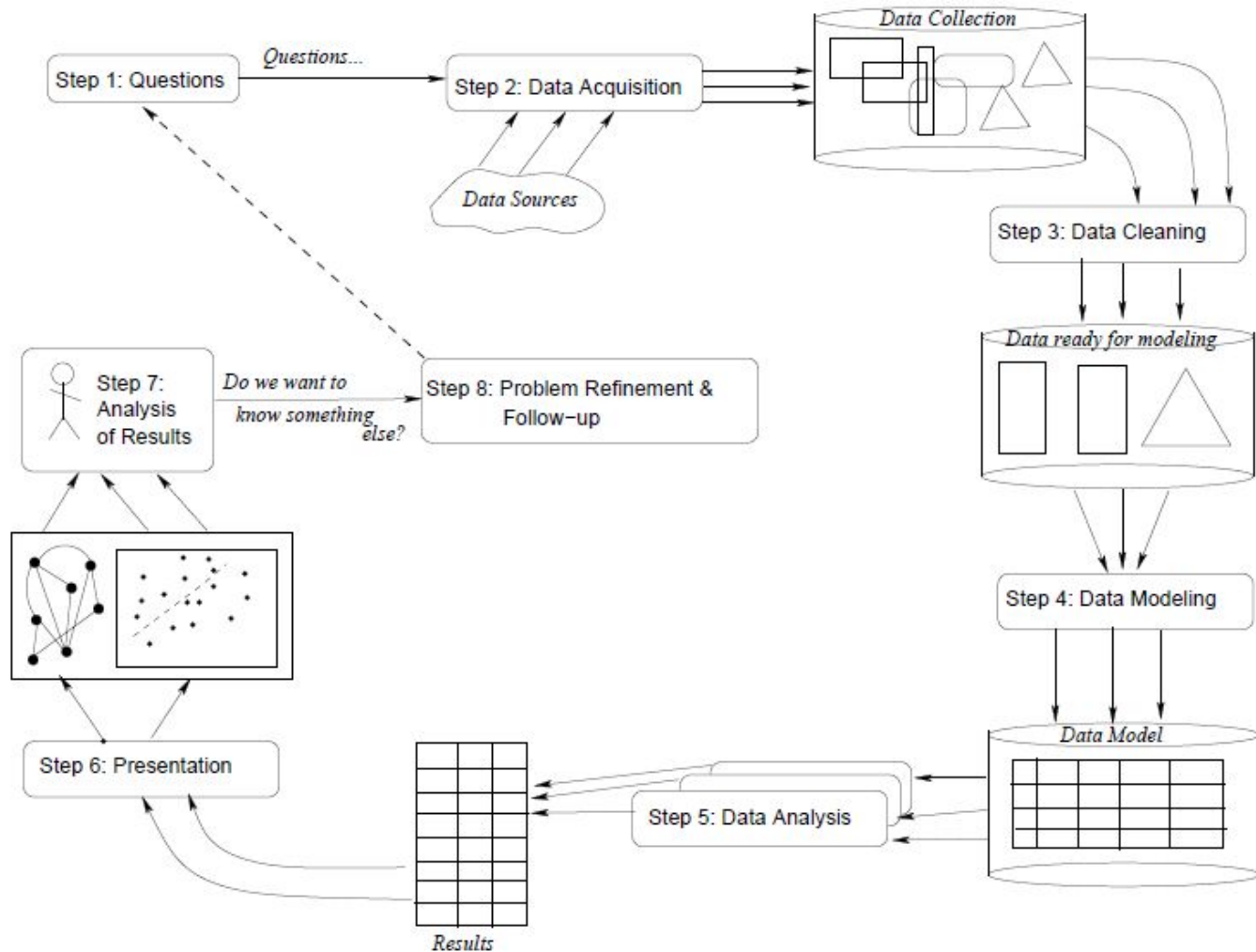
CRISP-DM

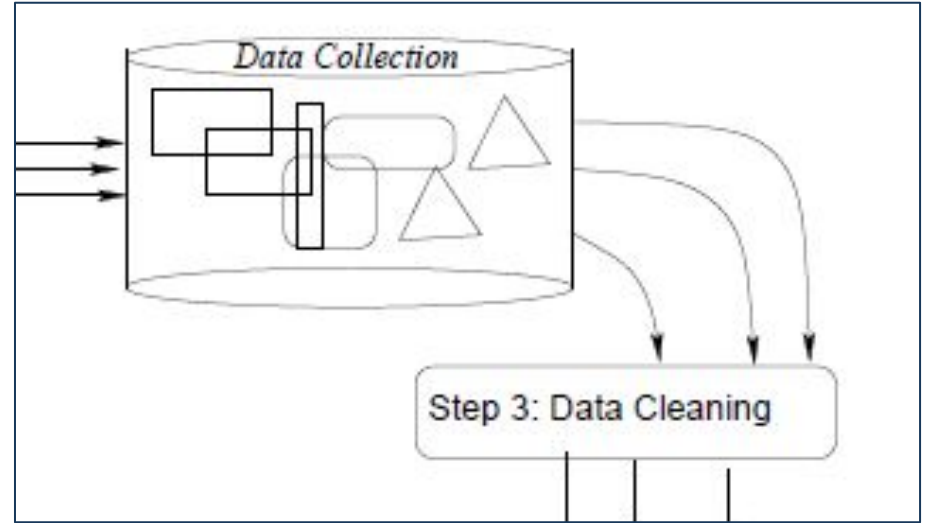
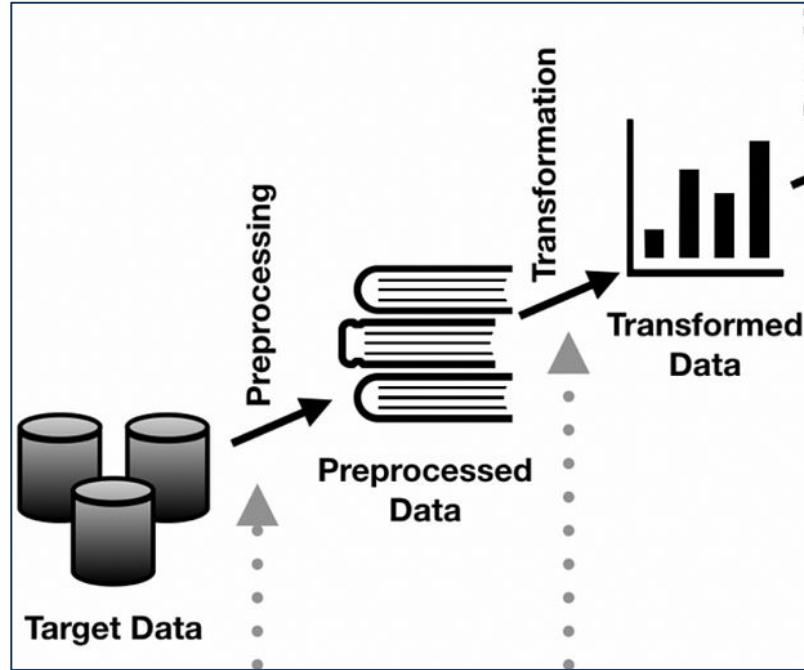


KDD

Processo de Ciência de Dados como um Ciclo

por Alexander
Dekhtyar





Por que é
essencial
Pré-processar ?

Preprocessing Breast Cancer Data to Improve the Data Quality, Diagnosis Procedure, and Medical Care Services

Zeinab Sajjadnia¹ , Raof Khayami¹ and Mohammad Reza Moosavi²

¹Department of Computer and IT Engineering, Shiraz University of Technology, Shiraz, Iran.

²Department of Computer Science and Engineering and IT, Shiraz University, Shiraz, Iran.

Cancer Informatics

Volume 19: 1–16

© The Author(s) 2020

Article reuse guidelines:

sagepub.com/journals-permissions

DOI: 10.1177/1176935120917955



ABSTRACT: In recent years, due to an increase in the incidence of different cancers, various data sources are available in this field. Consequently, many researchers have become interested in the discovery of useful knowledge from available data to assist faster decision-making by doctors and reduce the negative consequences of such diseases. Data mining includes a set of useful techniques in the discovery of knowledge from the data: detecting hidden patterns and finding unknown relations. However, these techniques face several challenges with real-world data. Particularly, dealing with inconsistencies, errors, noise, and missing values requires appropriate preprocessing and data preparation procedures. In this article, we investigate the impact of preprocessing to provide high-quality data for classification techniques. A wide range of preprocessing and data preparation methods are studied, and a set of preprocessing steps was leveraged to obtain appropriate classification results. The preprocessing is done on a real-world breast cancer dataset of the Reza Radiation Oncology Center in Mashhad with various features and a great percentage of null values, and the results are reported in this article. To evaluate the impact of the preprocessing steps on the results of classification algorithms, this case study was divided into the following 3 experiments:

Breast cancer recurrence prediction without data preprocessing

Breast cancer recurrence prediction by error removal

Breast cancer recurrence prediction by error removal and filling null values

Then, in each experiment, dimensionality reduction techniques are used to select a suitable subset of features for the problem at hand.

Preprocessing Breast Cancer Data to Improve the Data Quality, Diagnosis Procedure, and Medical Care Services

Zeinab Sajjadnia¹ , Raof Khayami¹ and Mohammad Reza Moosavi²

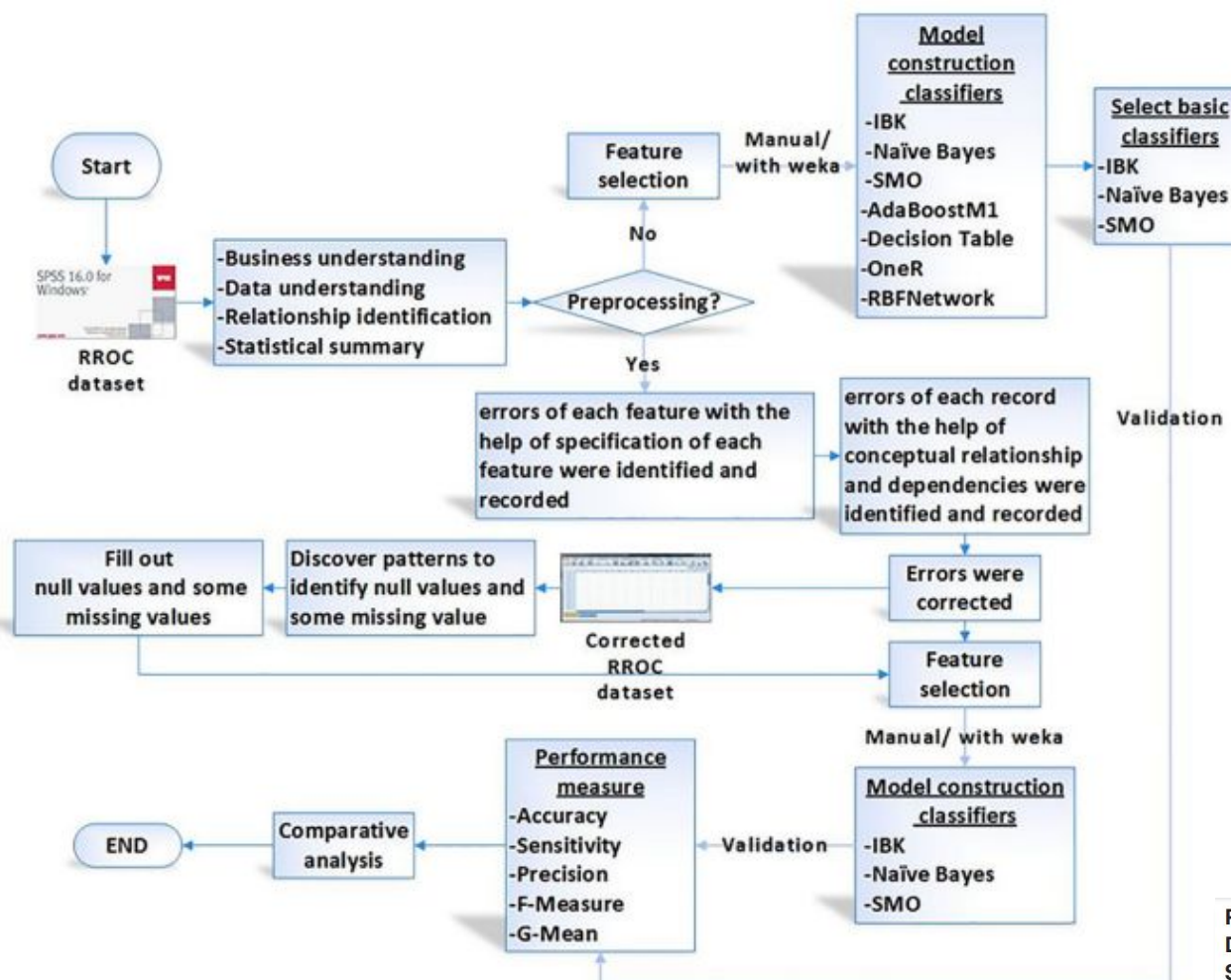
Cancer Informatics
Volume 19: 1–16
© The Author(s) 2020
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/1176935120917955



- Os autores buscaram avaliar como diferentes técnicas de pré-processamento de dados melhoram o desempenho de modelos preditivos para a predição de câncer recorrente de mama.
- Sobre o conjunto de dados:
 - 101 features para cada registro (paciente) podem ter sido coletadas;
 - 1923 registros de pacientes diagnosticados com câncer de mama (n=1923);
 - 78 apresentaram recorrência de câncer e 1742 não

$$78 + 1742 = 1820$$

103 restantes??



Preprocessing Breast Cancer Data to Improve the Data Quality, Diagnosis Procedure, and Medical Care Services

Zeinab Sajjadnia¹, Raof Khayami¹ and Mohammad Reza Moosavi²

Figure 1. The steps of case study in RROC.
RBF indicates radial basis function; RROC, Reza Radiation Oncology Center; SMO, sequential minimal optimization.

Preprocessing Breast Cancer Data to Improve the Data Quality, Diagnosis Procedure, and Medical Care Services

Zeinab Sajjadnia¹ , Raof Khayami¹ and Mohammad Reza Moosavi²

Cancer Informatics
Volume 19: 1–16
© The Author(s) 2020
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/1176935120917955



- Técnicas de pré-processamento realizadas:
 - Identificação de valores inconsistente (qualidade dados)
 - Tratamento de outliers
 - Padronização dos dados (z-score)
 - Tratamento de missing data
 - Balanceamento de dados (problema comum em conjunto de dados da área da saúde)
 - Redução da dimensionalidade

Table 4. Evaluating classification algorithms before data preprocessing.

CLASSIFIER	DIMENSION	ACCURACY	SENSITIVITY	PRECISION	F-MEASURE	G-MEAN
IBK (<i>k</i> -nearest neighbor)	(manual)25	92.66	9.74	10.75	10.22	30.64
	(manual)59	92.26	8.97	9.11	9.04	29.35
	(manual)79	91.56	27.82	18.24	22.03	51.25
	(manual)93	89.87	39.49	18.34	25.05	60.32
	(weka)22	94.85	3.85	13.76	6.01	19.51
	(weka)93	89.87	39.49	18.34	25.05	60.32

Table 6. Evaluating classification algorithms after filling null values.

CLASSIFIER	DIMENSION	ACCURACY	SENSITIVITY	PRECISION	F-MEASURE	G-MEAN
IBK (<i>k</i> -nearest neighbor)	(manual)25	98.06	70.64	81.75	75.79	83.75
	(manual)59	96.37	34.10	64.41	44.59	58.15
	(manual)79	95.92	42.95	52.92	47.42	64.97
	(manual)93	95.60	49.36	48.67	49.01	69.43
	(weka)22	97.51	58.08	78.24	66.67	75.93
	(weka)93	95.63	48.97	49.04	49.01	69.18

Pré-processamento

Pré-processamento de dados

- **Limpeza de dados:**
 - Tratamento de valores ausentes (missing values)
 - Tratamento de valores duplicados
 - Tratamento de outliers
- **Transformação de dados:**
 - Conversão de tipos
 - Normalização
 - Discretização
 - Binarização
 - Codificação
 - Transposição (pivotagem)
- **Seleção de features:**
 - Redução de features

Problemas na qualidade dos dados

- **Dados incompletos (ausentes):** ausência de valores para alguns dos atributos em parte dos dados.
- **Dados redundantes:**
 - quando dois ou mais objetos têm os mesmo valores para todos os atributos; ou
 - quando dois ou mais atributos têm os mesmos valores para dois ou mais objetos.
- **Dados inconsistentes:** não combinam ou contradizem valores de outros atributos do mesmo objeto.
- **Dados ruidosos (outliers):** possuem erros ou valores que são diferentes do esperado.

Dados ausentes (*missing values*)

Exemplo:

Tabela 3.2 *Conjunto de dados com atributos com valores ausentes*

Idade	Sexo	Peso	Manchas	Temp.	# Int.	Diagnóstico
—	M	79	—	38,0	—	Doente
18	F	67	Inexistentes	39,5	4	Doente
49	M	92	Espalhadas	38,0	2	Saudável
18	—	43	Inexistentes	38,5	8	Doente
21	F	52	Uniformes	37,6	1	Saudável
22	F	72	Inexistentes	38,0	3	Doente
—	F	87	Espalhadas	39,0	6	Doente
34	M	67	Uniformes	38,4	2	Saudável

Dados redundantes

Exemplo:

Tabela 3.5 *Conjunto de dados com objetos redundantes*

Idade	Sexo	Peso	Manchas	Temp.	# Int.	Diagnóstico
28	M	79	Concentradas	38,0	2	Doente
18	F	67	Inexistentes	39,5	4	Doente
49	M	92	Espalhadas	38,0	2	Saudável
18	F	67	Inexistentes	39,5	4	Doente
18	M	43	Inexistentes	38,5	8	Doente
21	F	52	Uniformes	37,6	1	Saudável
22	F	72	Inexistentes	38,0	3	Doente
19	F	87	Espalhadas	39,0	6	Doente
34	M	67	Uniformes	38,4	2	Saudável

É mesmo redundante ou são duas amostras?

Dados inconsistentes

Exemplo: valores iguais dos atributos [idade,sexo,peso,manchas,temp.,int] e valores diferentes para o atributo [diagnóstico]

Tabela 3.4 *Conjunto de dados com objetos inconsistentes*

Idade	Sexo	Peso	Manchas	Temp.	# Int.	Diagnóstico
28	M	79	Concentradas	38,0	2	Doente
18	F	67	Inexistentes	39,5	4	Doente
49	M	92	Espalhadas	38,0	2	Saudável
18	M	43	Inexistentes	38,5	8	Doente
21	F	52	Uniformes	37,6	1	Saudável
22	F	72	Inexistentes	38,0	3	Doente
19	F	87	Espalhadas	39,0	6	Doente
22	F	72	Inexistentes	38,0	3	Saudável

Dados com ruídos ou outlier

Exemplo:

Tabela 3.7 *Conjunto de dados com ruído*

Idade	Sexo	Peso	Manchas	Temp.	# Int.	Diagnóstico
28	M	79	Concentradas	38,0	2	Doente
18	F	300	Inexistentes	39,5	4	Doente
49	M	92	Espalhadas	38,0	2	Saudável
18	M	43	Inexistentes	38,5	8	Doente
21	F	52	Uniformes	37,6	1	Saudável
22	F	72	Inexistentes	38,0	3	Doente
19	F	87	Espalhadas	39,0	6	Doente
34	M	67	Uniformes	38,4	2	Saudável

Tratamento de dados ausentes

Pré-processamento de dados

Dados ausentes: Como identificar?

- Funções:
 - `.isna()`
 - `.info()`
- Imprime um resumo sobre o **DataFrame**, incluindo valores nulos e tipos dos dados

`pandas.DataFrame.isna`

`DataFrame.isna()`

`df.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 5 columns):
#   Column  Non-Null Count  Dtype
---  -
0   COL_1   3 non-null      float64
1   COL_2   1 non-null      float64
2   COL_3   5 non-null      int64
3   COL_4   5 non-null      int64
4   COL_5   4 non-null      float64
dtypes: float64(3), int64(2)
memory usage: 328.0 bytes
```

Dados ausentes: Como tratar?

1. Remover os registros com dados ausentes
2. Preencher os valores ausentes (imputação):
 - a. atribuir uma constante
 - b. atribuir um valor médio/mediana/moda
 - c. atribuir um valor estimado



Dataset de exemplo:

```
df = pd.DataFrame({'COL_1':[212, 434, np.nan, 44, np.nan],
                  'COL_2':[43, np.nan, np.nan, np.nan, np.nan],
                  'COL_3':[555, 603, 102, 77, 809],
                  'COL_4':[567, 560, 614, 88, 128],
                  'COL_5':[555, 603, 102, 64, np.nan]})
```

df

	COL_1	COL_2	COL_3	COL_4	COL_5
0	212.0	43.0	555	567	555.0
1	434.0	NaN	603	560	603.0
2	NaN	NaN	102	614	102.0
3	44.0	NaN	77	88	64.0
4	NaN	NaN	809	128	NaN



Tipos de dados: NaN (Not a Number) e None

- **NaN**: é um valor especial definido no padrão de ponto flutuante para representar valor indefinido.
 - **None**: é um valor do tipo NoneType que representa a ausência de um valor ou a ausência de dados em variáveis.
- O NumPy oferece funções específicas que permitem lidar com esses valores de maneira eficiente, permitindo, por exemplo, filtrar, substituir ou ignorar.

numpy.isnan

```
numpy.isnan(x, /, out=None, *, where=True, casting='same_kind', order='K', dtype=None, subok=True[, signature, extobj]) = <ufunc 'isnan'>
```

```
np.nan == np.nan
```

```
False
```

numpy.nanmean

```
numpy.nanmean(a, axis=None, dtype=None, out=None, keepdims=<no value>, *, where=<no value>)
```

[\[source\]](#)

Dados ausentes: Como tratar?

- **Tratamento 01:** remoção de registros.
- Uso da função **dropna()** para remover **linhas** ou colunas com NaN

pandas.DataFrame.dropna

```
DataFrame.dropna(*, axis=0, how=_NoDefault.no_default,
thresh=_NoDefault.no_default, subset=None, inplace=False,
ignore_index=False) #
```

	COL_1	COL_2	COL_3	COL_4	COL_5
0	212.0	43.0	555	567	555.0
1	434.0	NaN	603	560	603.0
2	NaN	NaN	102	614	102.0
3	44.0	NaN	77	88	64.0
4	NaN	NaN	809	128	NaN

```
df.dropna(axis=0) # axis: 0 or 'index', 1 or 'columns'
```

	COL_1	COL_2	COL_3	COL_4	COL_5
0	212.0	43.0	555	567	555.0



Dados ausentes: Como tratar?

- **Tratamento 01:** remoção de registros.
- Uso da função **dropna()** para remover linhas ou **colunas** com NaN

	COL_1	COL_2	COL_3	COL_4	COL_5
0	212.0	43.0	555	567	555.0
1	434.0	NaN	603	560	603.0
2	NaN	NaN	102	614	102.0
3	44.0	NaN	77	88	64.0
4	NaN	NaN	809	128	NaN

```
df.dropna(axis=1)
```

	COL_3	COL_4
0	555	567
1	603	560
2	102	614
3	77	88
4	809	128



	COL_1	COL_2	COL_3	COL_4	COL_5
0	212.0	43.0	555	567	555.0
1	434.0	NaN	603	560	603.0
2	NaN	NaN	102	614	102.0
3	44.0	NaN	77	88	64.0
4	NaN	NaN	809	128	NaN

```
df.dropna(axis=0) # axis: 0 or 'index', 1 or 'columns'
```

	COL_1	COL_2	COL_3	COL_4	COL_5
0	212.0	43.0	555	567	555.0

```
df.dropna(axis=1)
```

	COL_3	COL_4
0	555	567
1	603	560
2	102	614
3	77	88
4	809	128

perda de registros
+
desbalanceamento

Dados ausentes: Como tratar?

- **Tratamento 01:** remoção de registros.
- Uso da função **dropna()** para remover linhas ou colunas com NaN
- Definir um limiar (%) de valores com o parâmetro **thresh**

	COL_1	COL_2	COL_3	COL_4	COL_5
0	212.0	43.0	555	567	555.0
1	434.0	NaN	603	560	603.0
2	NaN	NaN	102	614	102.0
3	44.0	NaN	77	88	64.0
4	NaN	NaN	809	128	NaN

```
df.dropna(axis=0, thresh=4) # thresh: min aceitável de valores presentes
```

	COL_1	COL_2	COL_3	COL_4	COL_5
0	212.0	43.0	555	567	555.0
1	434.0	NaN	603	560	603.0
3	44.0	NaN	77	88	64.0



Dados ausentes: Como tratar?

- **Tratamento 02:** atribuir valores aos registros ausentes.
- Atribuir uma **constante** (ex: valor 0)

pandas.DataFrame.fillna

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

	COL_1	COL_2	COL_3	COL_4	COL_5
0	212.0	43.0	555	567	555.0
1	434.0	NaN	603	560	603.0
2	NaN	NaN	102	614	102.0
3	44.0	NaN	77	88	64.0
4	NaN	NaN	809	128	NaN

```
df.fillna(value=0)
```

	COL_1	COL_2	COL_3	COL_4	COL_5
0	212.0	43.0	555	567	555.0
1	434.0	0.0	603	560	603.0
2	0.0	0.0	102	614	102.0
3	44.0	0.0	77	88	64.0
4	0.0	0.0	809	128	0.0

Dados ausentes: Como tratar?

- **Tratamento 02:** atribuir valores aos registros ausentes.
- Atribuir a **média/mediana** da coluna

pandas.DataFrame.fillna

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

```
df['COL_1'] = df['COL_1'].fillna(value = df['COL_1'].mean())
df
```

	COL_1	COL_2	COL_3	COL_4	COL_5
0	212.0	43.0	555	567	555.0
1	434.0	NaN	603	560	603.0
2	230.0	NaN	102	614	102.0
3	44.0	NaN	77	88	64.0
4	230.0	NaN	809	128	NaN



Dataset do hospital:

	Idade	Sexo	Peso	Manchas	Temp.	Int.	Diagnóstico
0	NaN	M	79	NaN	38.0	NaN	Doente
1	18.0	F	67	Inexistentes	39.5	4.0	Doente
2	49.0	M	92	Espalhadas	38.0	2.0	Saudável
3	18.0	M	43	Inexistentes	38.5	8.0	Doente
4	21.0	F	52	Uniformes	37.6	1.0	Saudável
5	22.0	F	72	Inexistentes	38.0	3.0	Doente
6	NaN	F	87	Espalhadas	39.0	6.0	Doente
7	34.0	M	67	Uniformes	38.4	2.0	Saudável

Dados ausentes: Como tratar?

- **Tratamento:** atribuir a **média**

	Idade	Sexo	Peso	Manchas	Temp.	Int.	Diagnóstico
0	27.0	M	79	NaN	38.0	NaN	Doente
1	18.0	F	67	Inexistentes	39.5	4.0	Doente
2	49.0	M	92	Espalhadas	38.0	2.0	Saudável
3	18.0	M	43	Inexistentes	38.5	8.0	Doente
4	21.0	F	52	Uniformes	37.6	1.0	Saudável
5	22.0	F	72	Inexistentes	38.0	3.0	Doente
6	27.0	F	87	Espalhadas	39.0	6.0	Doente
7	34.0	M	67	Uniformes	38.4	2.0	Saudável

- **Tratamento:** atribuir a **mediana**

	Idade	Sexo	Peso	Manchas	Temp.	Int.	Diagnóstico
0	21.5	M	79	NaN	38.0	NaN	Doente
1	18.0	F	67	Inexistentes	39.5	4.0	Doente
2	49.0	M	92	Espalhadas	38.0	2.0	Saudável
3	18.0	M	43	Inexistentes	38.5	8.0	Doente
4	21.0	F	52	Uniformes	37.6	1.0	Saudável
5	22.0	F	72	Inexistentes	38.0	3.0	Doente
6	21.5	F	87	Espalhadas	39.0	6.0	Doente
7	34.0	M	67	Uniformes	38.4	2.0	Saudável

Dados ausentes: Como tratar?

- Tratamento: atribuir a **média**
- Tratamento: atribuir a **mediana**

Se for uma
variável
categórica?

	Idade	Sexo	Peso	Manchas	Temp.	Int.	Diagnóstico
0	27.0	M	79	NaN	38.0	NaN	Doente
1	18.0	F	67	Inexistentes	39.5	4.0	Doente
2	49.0	M	92	Espalhadas	38.0	2.0	Saudável
3	18.0	M	43	Inexistentes	38.5	8.0	Doente
4	21.0	F	52	Uniformes	37.6	1.0	Saudável
5	22.0	F	72	Inexistentes	38.0	3.0	Doente
6	27.0	F	87	Espalhadas	39.0	6.0	Doente
7	34.0	M	67	Uniformes	38.4	2.0	Saudável

Dados ausentes: Como tratar?

- **Tratamento 02:** atribuir valores aos registros ausentes.
- Atribuir a **MODA** da coluna

```
df_hospital_nan['Manchas'] = df_hospital_nan['Manchas'].fillna(value = df_hospital_nan['Manchas'].mode())
df_hospital_nan
```

	Idade	Sexo	Peso	Manchas	Temp.	Int.	Diagnóstico
0	21.5	M	79	Inexistentes	38.0	NaN	Doente
1	18.0	F	67	Inexistentes	39.5	4.0	Doente
2	49.0	M	92	Espalhadas	38.0	2.0	Saudável
3	18.0	M	43	Inexistentes	38.5	8.0	Doente
4	21.0	F	52	Uniformes	37.6	1.0	Saudável
5	22.0	F	72	Inexistentes	38.0	3.0	Doente
6	21.5	F	87	Espalhadas	39.0	6.0	Doente
7	34.0	M	67	Uniformes	38.4	2.0	Saudável



Tratamento de dados redundantes (duplicados)

Pré-processamento de dados

Dados redundantes (duplicados): Como identificar?

- Uso da função `.duplicated()`

`pandas.DataFrame.duplicated`

`DataFrame.duplicated(subset=None, keep='first')`

Return boolean Series denoting duplicate rows.

	brand	style	rating
0	Yum Yum	cup	4.0
1	Yum Yum	cup	4.0
2	Indomie	cup	3.5
3	Indomie	pack	15.0
4	Indomie	pack	5.0



`df1.duplicated()`

```
0    False
1     True
2    False
3    False
4    False
dtype: bool
```

Dados redundantes (duplicados): Como identificar?

- Uso da função `.duplicated()`

	brand	style	rating
0	Yum Yum	cup	4.0
1	Yum Yum	cup	4.0
2	Indomie	cup	3.5
3	Indomie	pack	15.0
4	Indomie	pack	5.0



```
df1.duplicated(keep=False)
```

```
0    True
1    True
2   False
3   False
4   False
dtype: bool
```

```
df1.duplicated(keep='last')
```

```
0    True
1   False
2   False
3   False
4   False
dtype: bool
```



Dados redundantes (duplicados): Como tratar?

- Uso da função `.drop_duplicates()`

`pandas.DataFrame.drop_duplicates`

```
DataFrame.drop_duplicates(subset=None, *, keep='first', inplace=False,
                           ignore_index=False) # \[source\]
```

	brand	style	rating
0	Yum Yum	cup	4.0
1	Yum Yum	cup	4.0
2	Indomie	cup	3.5
3	Indomie	pack	15.0
4	Indomie	pack	5.0



df1.drop_duplicates()			
	brand	style	rating
0	Yum Yum	cup	4.0
2	Indomie	cup	3.5
3	Indomie	pack	15.0
4	Indomie	pack	5.0

Tratamento de dados com ruídos

=

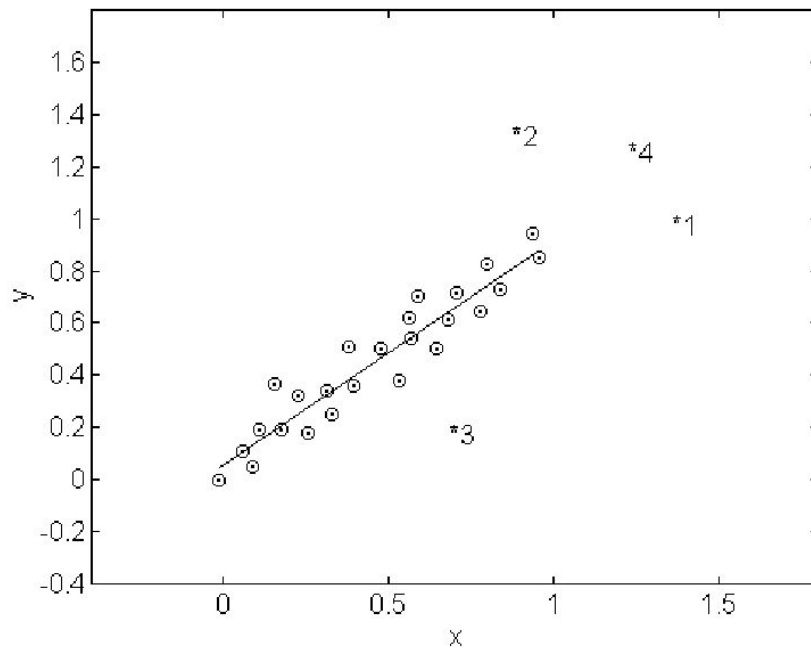
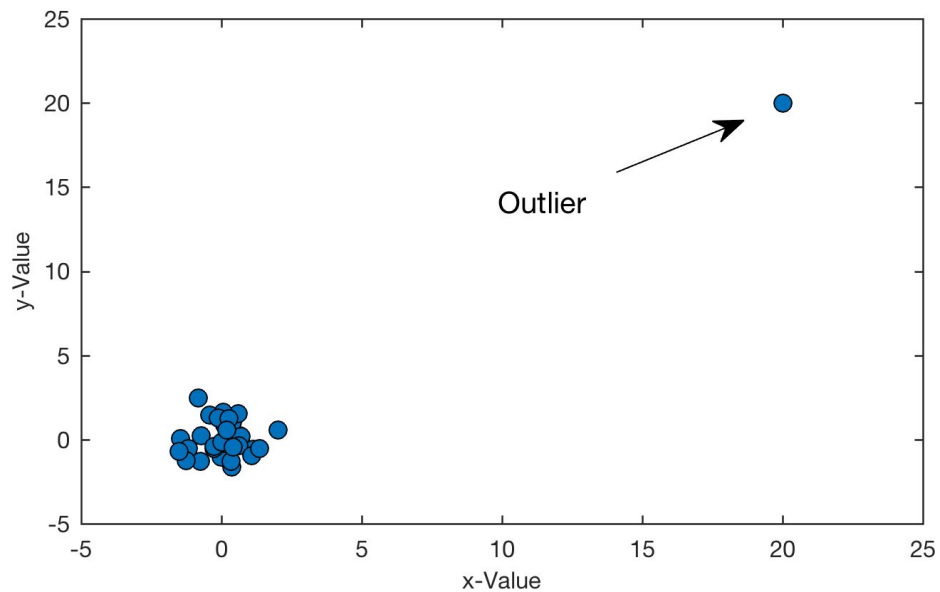
Tratamento de *Outliers*

Pré-processamento de dados

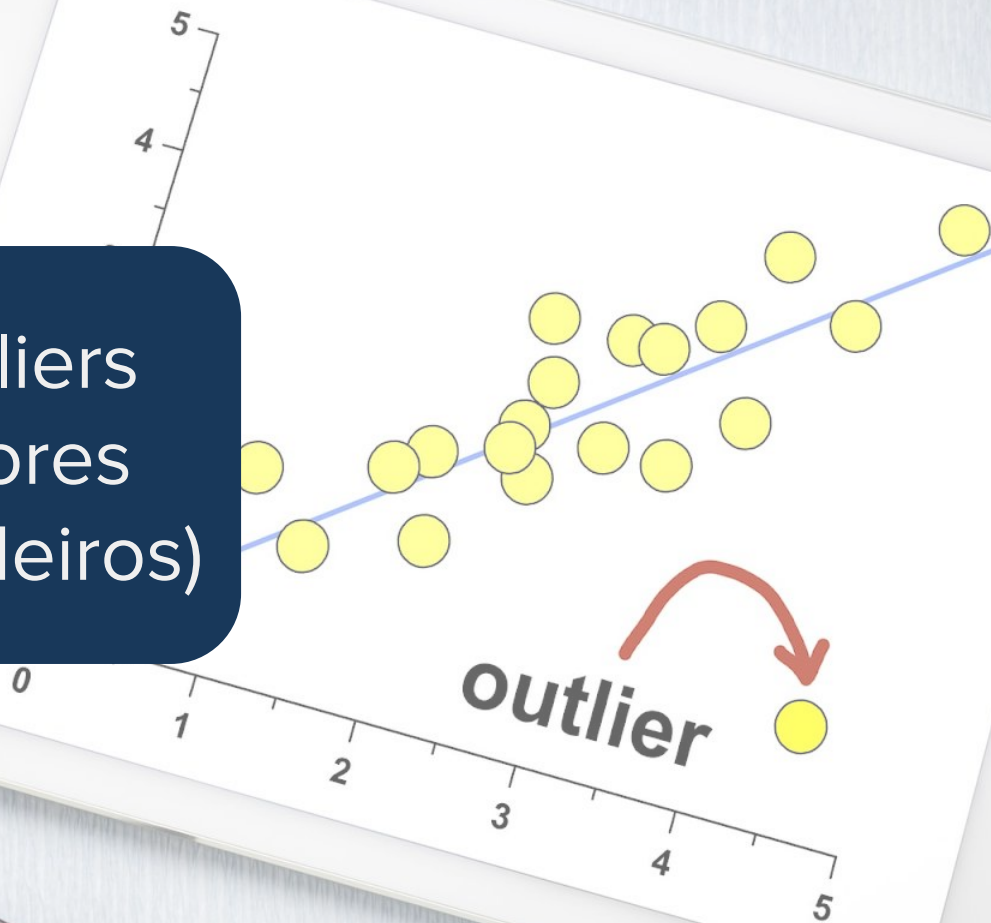
Dados com ruídos ou outlier

- **Ruído** pode ser definido com uma variância ou erro aleatório no valor gerado ou medido para um atributo.
- Os valores ‘fora da curva’ também são conhecidos como **outliers**.
- **Outliers**: dados drasticamente diferentes dos demais; um valor que foge da normalidade ou distribuição dos dados.
- Um dado com ruído pode ser:
 - falha no processo de coleta dos dados (ex.: erro de digitação)
 - falha no processo de transformação (ex.: diferentes unidades)

Outliers

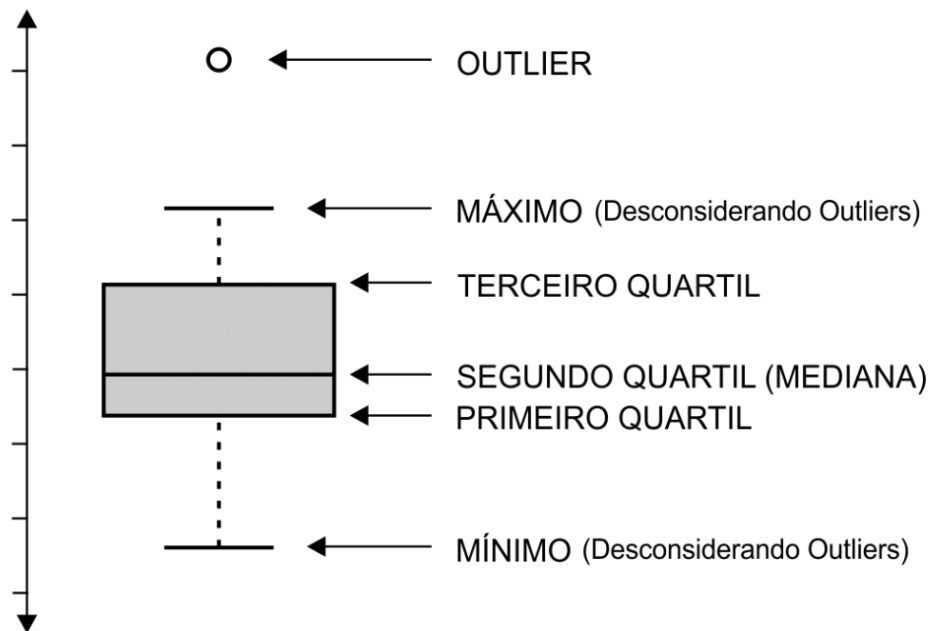


ATENÇÃO: outliers
podem ser valores
legítimos (verdadeiros)



Outliers: Como identificar?

- Técnicas de visualização ou estatística descritiva.
- Uso de medidas de localização quantis/separatrizes: **quartis**
 - **amplitude interquartil**



Dataset de exemplo:

```
df_pesos_alturas.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1000 entries, 1 to 1000
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  -
 0   PESO    1000 non-null   int64
 1   ALTURA 1000 non-null   int64
dtypes: int64(2)
memory usage: 23.4 KB
```

```
df_pesos_alturas.head(10)
```

	PESO	ALTURA
ID_PESSOA		
1	68	168
2	67	173
3	70	174
4	68	172
5	73	168
6	69	170
7	64	171
8	64	175
9	77	169
10	70	168



Outliers: Como identificar?

```
df_pesos_alturas.sort_values(by=['PESO'],
                             ascending=False).head(5)
```

PESO ALTURA

ID_PESSOA

759	701908	171
731	7100	169
758	80	169
52	78	169
56	78	170

```
df_pesos_alturas.sort_values(by=['PESO'],
                             ascending=True).head(5)
```

PESO ALTURA

ID_PESSOA

592	0	169
751	0	170
652	0	171
63	60	170
660	62	170

Outliers: Como identificar?

```
df_pesos_alturas.sort_values(by=['ALTURA'],
                             ascending=False).head(5)
```

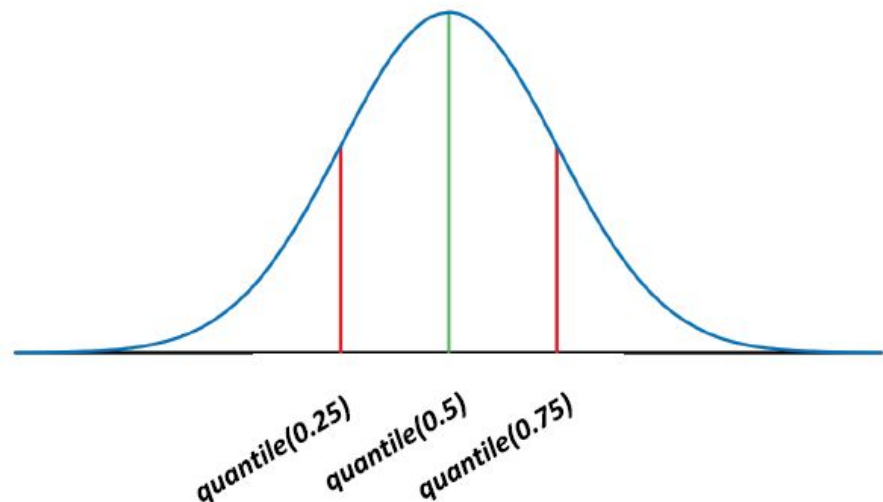
	PESO	ALTURA
ID_PESSOA		
415	69	17809
437	68	16300
115	69	178
402	68	177
367	71	176

```
df_pesos_alturas.sort_values(by=['ALTURA'],
                             ascending=True).head(5)
```

	PESO	ALTURA
ID_PESSOA		
740	73	1
724	67	15
654	67	17
683	65	163
710	68	164

Outliers: Como identificar?

- Uso da função **.quantile()** para filtrar os limites de outliers
- **quantile()** retorna o valor do percentil informando no parâmetro



pandas.DataFrame.quantile

```
DataFrame.quantile(q=0.5, axis=0, numeric_only=False,
                    interpolation='linear', method='single')
```

Return values at the given quantile over requested axis.

Parameters:

q : float or array-like, default 0.5 (50% quantile)

Value between $0 \leq q \leq 1$, the quantile(s) to compute.

axis : {0 or 'index', 1 or 'columns'}, default 0

Equals 0 or 'index' for row-wise, 1 or 'columns' for column-wise.

numeric_only : bool, default False

Include only float, int or boolean data.

Outliers: Como tratar?

- Tratamento 01: remover **os valores extremos superiores e inferiores**.
- Etapas:
 - a. Filtrar essas extremidades.
 - b. Remover os dados acima da extremidade superior e abaixo da extremidade inferior.
- O **valor de 0,3%** pode resolver no exemplo do dataset Pesos_Alturas.

	PESO	ALTURA
ID_PESSOA		
759	701908	171
731	7100	169
758	80	169
52	78	169
56	78	170

	PESO	ALTURA
ID_PESSOA		
592	0	169
751	0	170
652	0	171
63	60	170
660	62	170

	PESO	ALTURA
ID_PESSOA		
415	69	17809
437	68	16300
115	69	178
402	68	177
367	71	176

	PESO	ALTURA
ID_PESSOA		
740	73	1
724	67	15
654	67	17
683	65	163
710	68	164

Outliers: Como tratar?

- Tratamento 01: Remover **os valores extremos superiores e inferiores.**

```
df_pesos_alturas = df_pesos_alturas[
    (df_pesos_alturas['PESO'] > df_pesos_alturas['PESO'].quantile(0.003)) &
    (df_pesos_alturas['PESO'] < df_pesos_alturas['PESO'].quantile(0.997))]
```

df_pesos_alturas.info()

```
<class 'pandas.core.frame.DataFrame'>
Index: 1000 entries, 1 to 1000
Data columns (total 2 columns):
#   Column  Non-Null Count  Dtype
---  -
0    PESO    1000 non-null    int64
1    ALTURA  1000 non-null    int64
dtypes: int64(2)
memory usage: 23.4 KB
```



df_pesos_alturas.info()

```
<class 'pandas.core.frame.DataFrame'>
Index: 994 entries, 1 to 1000
Data columns (total 2 columns):
#   Column  Non-Null Count  Dtype
---  -
0    PESO    994 non-null    int64
1    ALTURA  994 non-null    int64
dtypes: int64(2)
memory usage: 23.3 KB
```

df_pesos_alturas.sort_values(by=['PESO'],
ascending=False).head(5)

	PESO	ALTURA
ID_PESSOA		
291	78	170
208	78	171
905	78	170
56	78	170
509	78	167

Outliers: Como tratar?

- Tratamento 02: substituir **os valores extremos** por **valor médio** sem extremos.
- Uso da função **.where()**

numpy.where

`numpy.where(condition, [x, y,]/)`

Return elements chosen from x or y depending on *condition*.

```
df['PESO'] = np.where(
    df['PESO'] >= df['PESO'].quantile(0.997),
    df[(df['PESO'] > df['PESO'].quantile(0.003)) & (df['PESO'] < df['PESO'].quantile(0.997))]['PESO'].mean(),
    df['PESO'])
```

Pré-processamento de dados

- **Limpeza de dados:**
 - Tratamento de valores ausentes (missing values)
 - Tratamento de valores duplicados
 - Tratamento de outliers
- **Transformação de dados:**
 - Conversão de tipos
 - Normalização
 - Discretização
 - Binarização
 - Codificação
 - Transposição (pivotagem)
- **Seleção de features:**
 - Redução de features



Dúvidas?

Tome cuidado com o pré-processamento!

O pré-processamento não pode alterar a natureza dos dados.

Domine bem os dados (a área de negócio).



O que vimos hoje?

Aula: Pré-processamento de Dados

IMD1151 - Ciência de Dados

Prof. Heitor Florencio

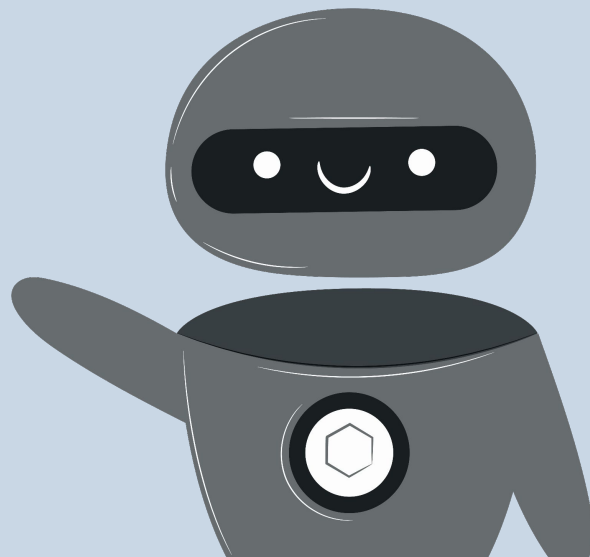
- Revisão: exploração de dados
- Pré-processamento de dados
- Limpeza de dados:
 - Tratamento de valores ausentes
 - Tratamento de valores duplicados
 - Tratamento de outliers



Dúvidas?

Prof. Heitor Florencio
Sala 103 - nPITI/IMD
heitorm@imd.ufrn.br

Prof. Daniel Sabino
Sala A226 - CIVT/IMD
daniel@imd.ufrn.br



Referências

- AMARAL, Fernando. **Introdução à ciência de dados: mineração de dados e big data**. Alta Books Editora, 2016.
- SILVEIRA, Juliano Gomes. **Pré-processamento de Dados**. 2022. Notas de aula.
- SAJJADNIA, Zeinab; KHAYAMI, Raof; MOOSAVI, Mohammad Reza. **Preprocessing breast cancer data to improve the data quality, diagnosis procedure, and medical care services**. Cancer Informatics, v. 19, p. 1176935120917955, 2020.
- CARVALHO, ACPLF et al. **Inteligência Artificial - uma abordagem de aprendizado de máquina**. Rio de Janeiro: LTC, v. 2, p. 45, 2011.
- BROWNLEE, Jason. **Data preparation for machine learning: data cleaning, feature selection, and data transforms in Python**. Machine Learning Mastery, 2020.