Cap04. Dados Ausentes

É necessário lidar com dados ausentes. Exploraremos um pouco mais o assunto. A maioria dos algoritmos não funcionará se houver dados ausentes.

Exceções dignas de nota são as recentes bibliotecas inovadoras **XGBoost, CatBoost e LightGBM.**

Assim como ocorre em relação a várias questões no machine learning, não há respostas únicas sobre como tratar dados ausentes. Além do mais, a ausência de dados poderia representar diferentes situações. Suponha que tenhamos recebido dados de censo e que um atributo idade tenha sido informado como ausente. Isso é porque a amostra não quis revelar a sua idade? Não sabia a idade? Quem fez as perguntas se esqueceu de pedi-la? Há algum padrão para as idades ausentes? Ela está correlacionada com outro atributo? Ou é totalmente aleatória?

Maneiras de lidar com dados ausentes:

- remover qualquer linha com dados ausentes;
- remover qualquer coluna com dados ausentes;
- imputar dados aos valores ausentes;
- criar uma coluna para informar que os dados estavam ausentes.

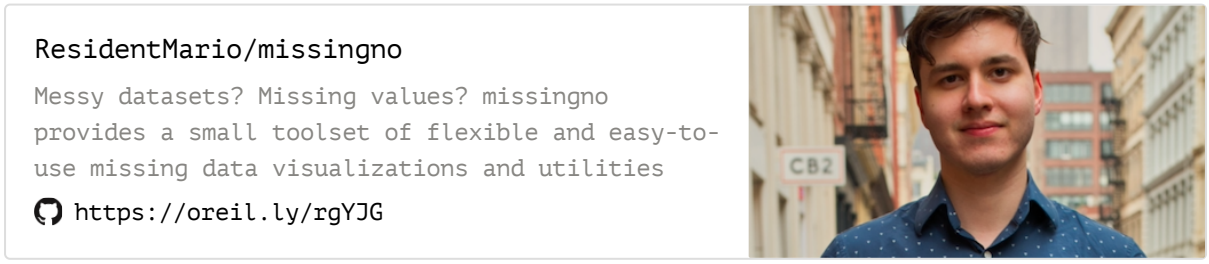
Analisando dados ausentes

Vamos olhar os dados do Titanic. Pelo fato de Python tratar True e False como 1 e 0 respectivamente, podemos usar esse truque no pandas para obter o percentual dos dados ausentes:

```
df.isnull().mean() * 100
"""
pclass 0.000000
survived 0.000000
name 0.000000
sex 0.000000
age 20.091673
```

```
sibsp 0.000000
parch 0.000000
ticket 0.000000
fare 0.076394
cabin 77.463713
embarked 0.152788
boat 62.872422
body 90.756303
home.dest 43.086325
dtype: float64
"""
```

Padrões em dados ausentes com missigno



Para visualizar padrões nos dados ausentes, utilize a biblioteca missingno, a qual é conveniente para visualizar áreas contíguas de dados ausentes, que sinalizariam que os dados ausentes não são aleatórios (veja a Figura 4.1).

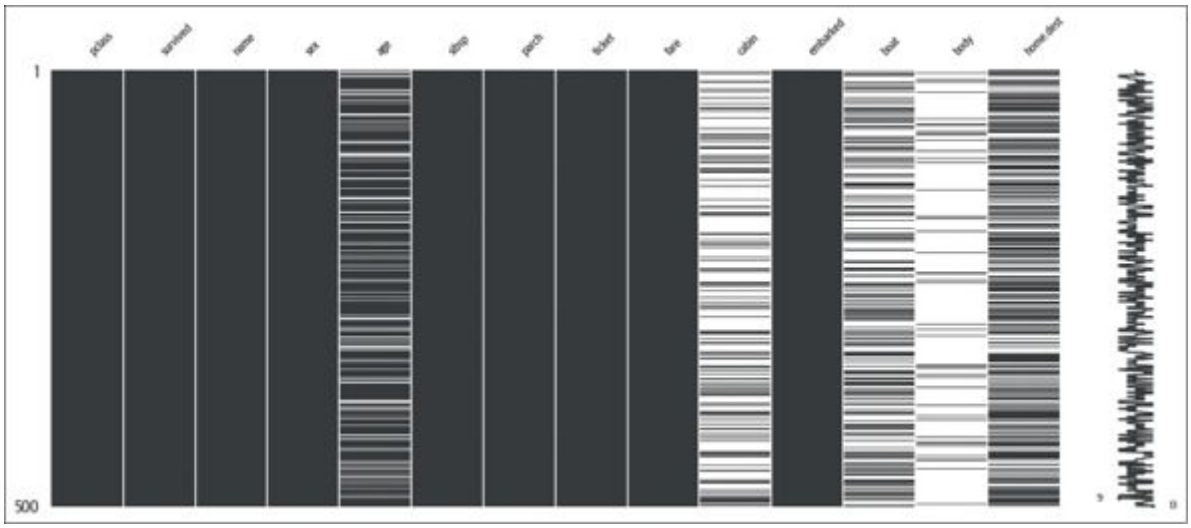


Figura 4.1 – Locais em que há dados ausentes. Nenhum padrão se destaca claramente.

A função matrix inclui uma área em destaque do lado direito. Padrões nesse local também indicariam que os dados ausentes não são aleatórios. Talvez seja necessário limitar o número de amostras para que seja possível ver os padrões:

```
import missingno as msno

ax = msno.matrix(orig_df.sample(500))

ax.get_figure().savefig("images/mlpr_0401.png")
```

Podemos criar um gráfico de barras com os contadores de dados ausentes usando o pandas (veja a Figura 4.2).

```
fig, ax = plt.subplots(figsize=(6, 4))

(1 - df.isnull().mean()).abs().plot.bar(ax=ax)

fig.savefig("images/mlpr_0402.png", dpi=300)
```

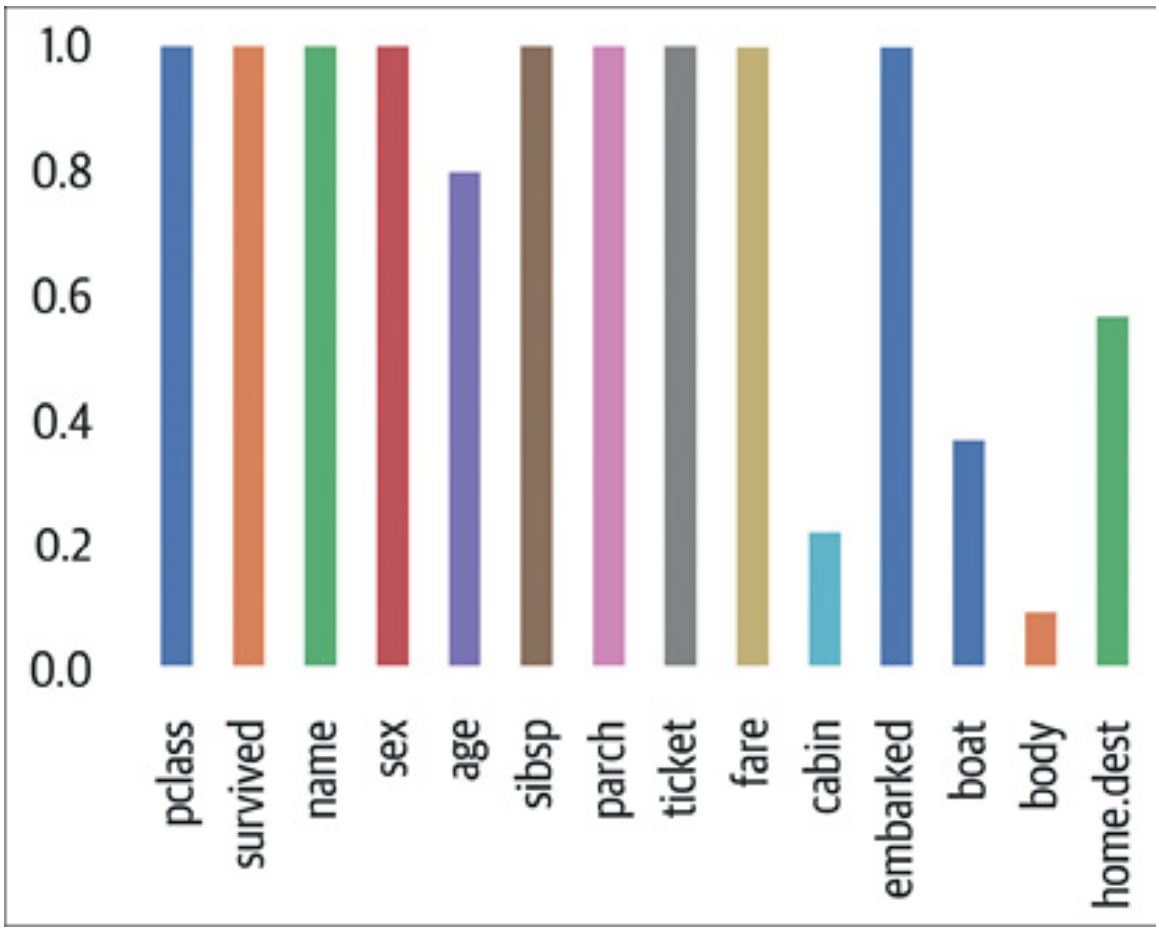


Figura 4.2 - Percentual de dados não ausentes com o pandas. Os dados de boat (barco salva-vidas) e body (número de identificação do corpo) causam vazamento de informações, portanto devemos ignorá-los. É interessante o fato de haver idades ausentes.

Podemos também usar a biblioteca missingno para gerar o mesmo gráfico:

```
ax = msno.bar(orig_df.sample(500))

ax.get_figure().savefig("images/mlpr_0403.png")
```

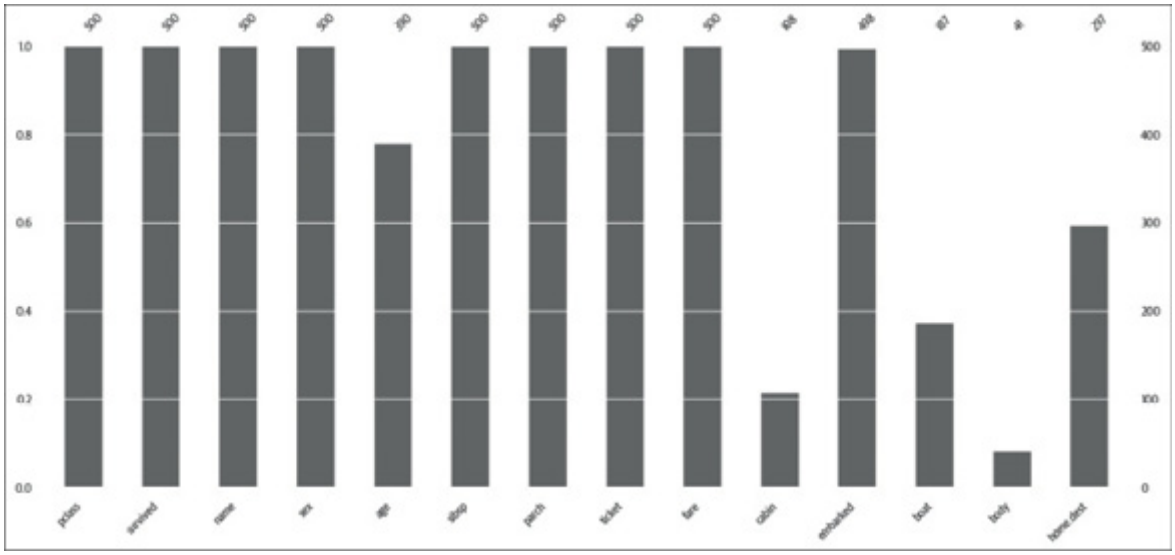


Figura 4.3 - Percentuais de dados não ausentes gerados com o missingno.

Um heat map (mapa de calor) pode ser criado, mostrando se há correlações no caso de haver dados ausentes (veja a Figura 4.4). Em nosso caso, não parece haver correlações entre os atributos que apresentam dados ausentes:

```
>>> ax = msno.heatmap(df, figsize=(6, 6))

>>> ax.get_figure().savefig("/tmp/mlpr_0404.png")
```

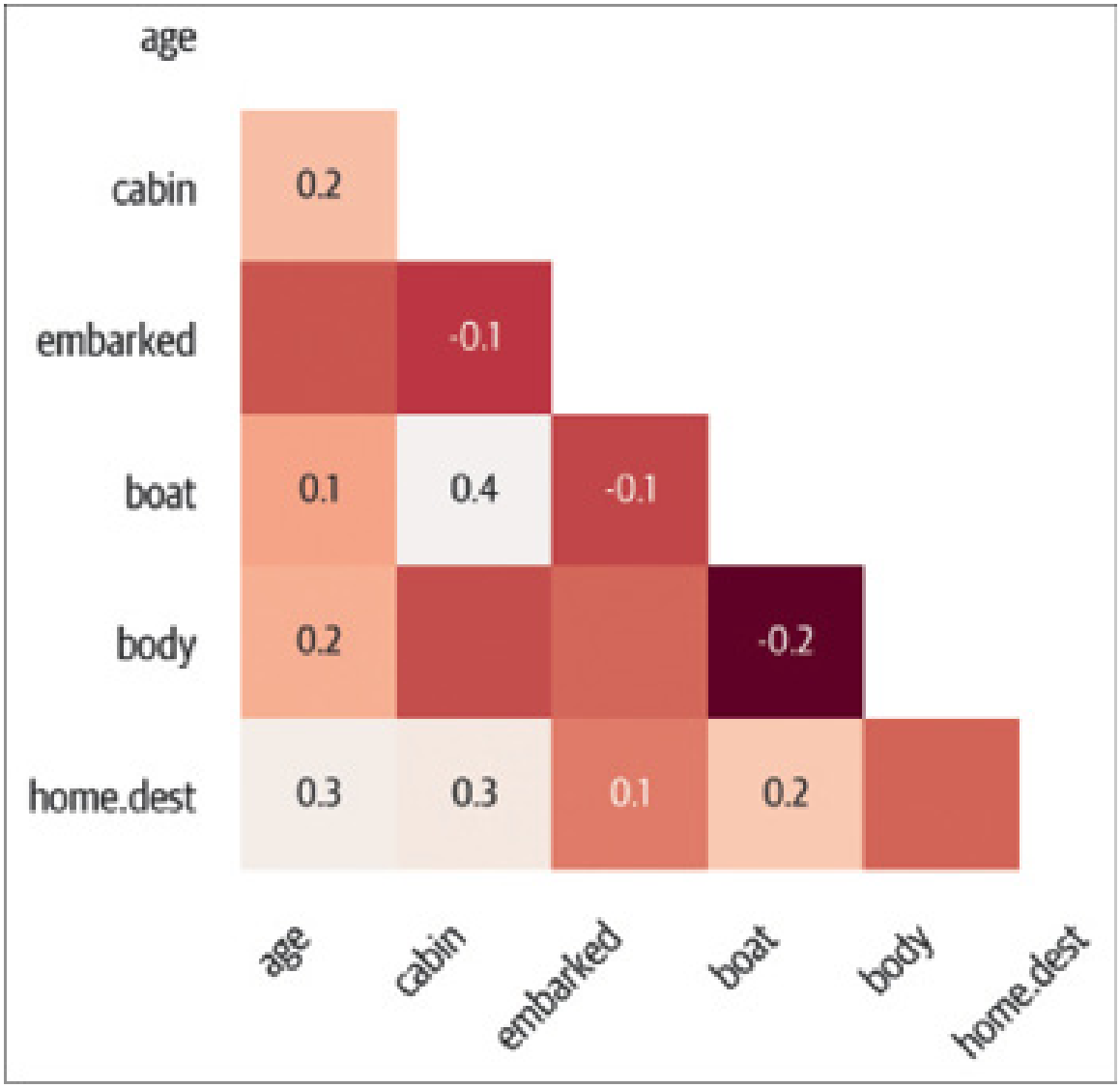


Figura 4.4 - Correlações entre dados ausentes geradas com o missingno.

Podemos criar um dendrograma que mostra os clusterings (agrupamentos) de dados ausentes (veja a Figura 4.5). As folhas que estão no mesmo nível fazem a predição da presença umas das outras (cheias ou preenchidas). Os braços verticais são usados para mostrar a diferença entre os clusters. Braços curtos indicam que os ramos são semelhantes:

```
ax = msno.dendrogram(df)

ax.get_figure().savefig("images/mlpr_0405.png")
```

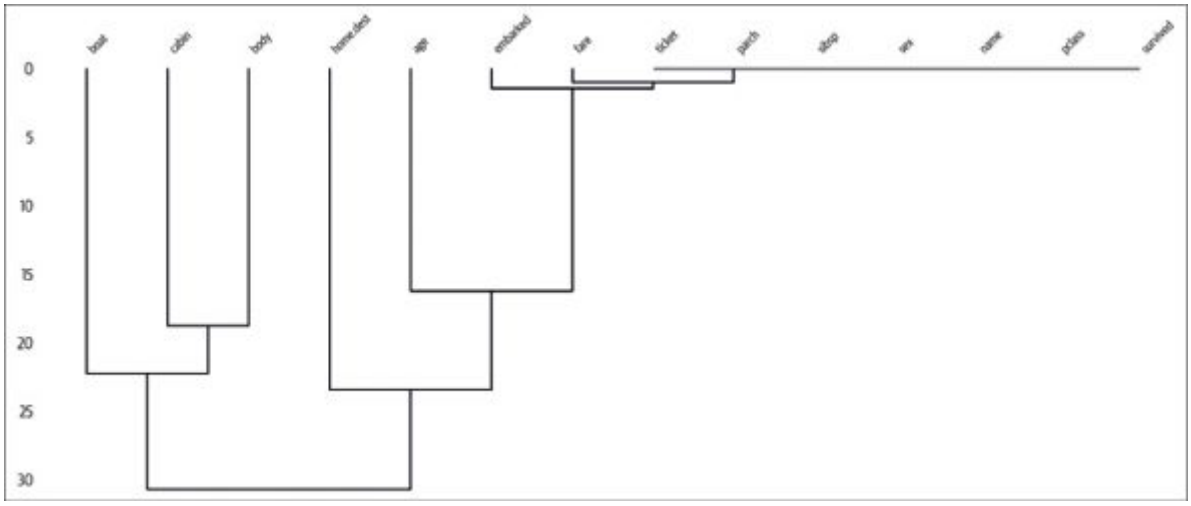


Figura 4.5 - Dendrograma de dados ausentes gerado com o missingno. Podemos ver as colunas que não têm dados ausentes na parte superior à direita.

Descartando dados ausentes

A biblioteca pandas é capaz de descartar todas as linhas contendo dados ausentes usando o método .dropna:

```
df1 = df.dropna()
```

Para descartar colunas, podemos observar quais colunas contêm dados ausentes e utilizar o método .drop. É possível passar uma lista de nomes de colunas ou um único nome de coluna:

```
df1 = df.drop(columns='cabin')
```

Como alternativa, podemos usar o método .dropna e definir axis=1 (descartar no eixo das colunas):

```
df1 = df.dropna(axis=1)
```

Tome cuidado ao descartar dados. Em geral, encaro essa opção como um último recurso.

Imputando dados

Depois que tivermos uma ferramenta para predição de dados, podemos usá-la para prever valores no caso de dados ausentes.

A tarefa geral para definir valores a dados ausentes chama-se imputação (imputation).

Se você estiver imputando dados, será necessário construir um pipeline e usar a mesma lógica de imputação na criação do modelo e no momento da predição. A classe SimpleImputer do scikit-learn trabalhará com a média, a mediana e com os valores mais frequentes dos atributos.

O comportamento default é calcular a média:

```
from sklearn.impute import SimpleImputer

num_cols = df.select_dtypes(include="number").columns

im = SimpleImputer() # média

imputed = im.fit_transform(df[num_cols])
```

Especifique strategy='median' ou strategy='most_frequent' para substituir o valor usado na substituição pela mediana ou pelo valor mais comum, respectivamente. Se quiser preencher o dado com um valor constante, por exemplo, -1, use strategy='constant' junto com fill_value=-1.

Dica

Você pode usar o método .fillna do Pandas para imputar dados aos valores ausentes também. Certifique-se, porém, de que não causará vazamento de dados. Se você estiver preenchendo valores usando a média, **não se esqueça de usar o mesmo valor de média na criação do modelo e no momento da predição**.

As estratégias para usar o valor mais frequente ou uma constante podem ser aplicadas com dados numéricos ou do tipo string. A média e a mediana exigem dados numéricos.

Biblioteca fancyimpute

A biblioteca fancyimpute implementa vários algoritmos e sua interface está em consonância com o scikit-learn.

Infelizmente, a maioria dos algoritmos são transdutivos, o que significa que você não poderá chamar o método .transform por si só após a adequação do algoritmo. IterativeImputer é indutivo (foi passado do fancyimpute para o scikit-learn) e aceita uma transformação após a adequação ao modelo.

Acrescentando colunas informativas

A ausência de dados por si só pode fornecer alguns sinais a um modelo. A biblioteca Pandas é capaz de acrescentar uma nova coluna para informar que um valor estava ausente:

```
def add_indicator(col):
  def wrapper(df):
    return df[col].isna().astype(int)

return wrapper

df1 = df.assing(
  cabin_missing = add_indicator("cabin"))
```