Cap05. Limpeza dos Dados

Ferramentas genéricas como o Pandas e ferramentas especializadas como o **pyjanitor** podem ser usadas para ajudar na limpeza dos dados.

Ajustar nomes das colunas

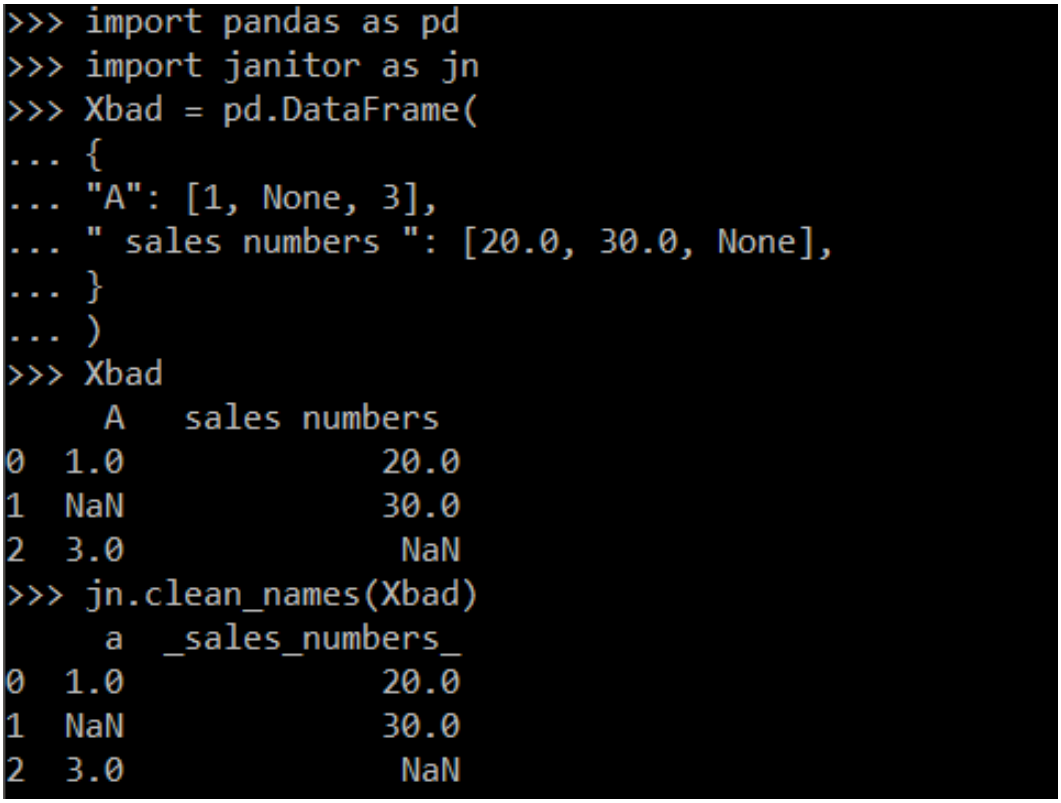
Ao usar Pandas, ter nomes de colunas apropriadas para Python torna possível o **acesso aos atributos**.

A função **jn.clean_names() do pyjanitor** devolverá um DataFrame com as colunas em letras minúsculas e os espaços substituídos por underscores:

```
import janitor as jn

Xbad = pd.DataFrame(
   {
       "A": [1, None, 3],
       " sales numbers ": [20.0, 30.0, None],
   }
)

jn.clean_names(Xbad)
```



```
>>> import pandas as pd
>>> import janitor as jn
>>> Xbad = pd.DataFrame(
... {
... "A": [1, None, 3],
... " sales numbers ": [20.0, 30.0, None],
... }
... )
>>> Xbad
     A   sales numbers
0  1.0             20.0
1  NaN             30.0
2  3.0              NaN
>>> jn.clean_names(Xbad)
     a  _sales_numbers_
0  1.0             20.0
1  NaN             30.0
2  3.0              NaN
```

Dica

Recomendo atualizar as colunas usando atribuição com índices, o método **.assign()** ou atribuições com **.loc[]** ou **.iloc[]**.

Também não recomendo usar atribuição para atualizar colunas em Pandas. Por causa do risco de sobrescrever métodos existentes com o mesmo nome de uma coluna, não há garantias de que uma atribuição vá funcionar.

A biblioteca pyjanitor é conveniente, mas **não nos permite remover espaços em branco em torno das colunas**.

Podemos usar o Pandas para ter um controle mais minucioso ao renomear as colunas:

```
def clean_col(name):
   return (
     name.strip().lower().replace(" ", "_")
   )
```

```
Xbad.rename(columns=clean_col)
```

```
>>> Xbad.rename(columns=clean_col)
    a sales_numbers
0 1.0 20.0
1 NaN 30.0
2 3.0 NaN
```

Substituindo valores ausentes

A função coalesce do pyjanitor recebe um DataFrame e uma lista de colunas para serem consideradas.

Essa é uma funcionalidade semelhante àquela que vemos no Excel e em bancos de dados SQL. Ela devolve o primeiro valor não nulo para cada linha:

```
jn.coalesce(
  Xbad,
  columns=["A", " sales numbers "],
  new_column_name="val",
)
  val
```

```
   val
0 1.0
1 30.0
2 3.0
```

Se quisermos substituir os valores ausentes por um valor específico, podemos usar o método **`.fillna()`** do DataFrame:

```
Xbad.fillna(10)
```

```
>>> Xbad.fillna(10)
      A   sales numbers
0   1.0            20.0
1  10.0            30.0
2   3.0            10.0
>>>
```

ou a função **jn.fill_empty()** do pyjanitor:

```
jn.fill_empty(
   Xbad,
   columns=["A", " sales numbers "],
   value=10,
```

```
>>> jn.fill_empty(
... Xbad,
... columns=["A", " sales numbers "],
... value=10,
... )
      A   sales numbers
0   1.0            20.0
1  10.0            30.0
2   3.0            10.0
```

Imputações

Com frequência, usaremos imputações de dados mais específicas do Pandas, do scikit-learn ou do fancyimpute para fazer substituição de valores nulos por coluna.

Para uma verificação de sanidade antes de criar os modelos, você pode usar o Pandas para garantir que todos os valores ausentes foram tratados.d

Verificar se há alguma célular ausente no DF

O código a seguir devolve um único booleano para informar se há alguma célula com valor ausente em um DataFrame:

```
df.isna().any().any()
# True
```