MBA em Ciência de Dados

Técnicas Avançadas de Captura e Tratamento de Dados

Módulo III - Aquisição e Transformação de Dados

Codificação de Variáveis Categóricas

Material Produzido por Moacir Antonelli Ponti

CeMEAI - ICMC/USP São Carlos

Variáveis categóricas

Variáveis categóricas, geralmente expressas por meio de texto, não são utilizáveis diretamente por parte dos modelos estatísticos e computacionais.

Obter codificações numéricas dessas variáveis se torna importante para permitir análise por um número mais amplo de ferramentas

In [1]:
```
# carregando as bibliotecas necessárias
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# carregando dados
data_orig = pd.read_csv("./dados/municipios_mba.csv")
data_orig.head()
```

Out[1]:

	gid	UF	nome	Censo	PIB	pop	classe	desemprego	po
0	752	ACRE	Acrelândia	2010.0	151120.015625	12241	2	5.2	33
1	747	ACRE	Assis Brasil	2010.0	48347.300781	5662	1	5.0	32
2	748	ACRE	Brasiléia	2010.0	194979.828125	20238	1	3.0	31
3	754	ACRE	Bujari	2010.0	88708.031250	6772	2	4.8	33
4	751	ACRE	Capixaba	2010.0	89052.679688	9287	1	4.4	33

Vamos estudar a variável `urbaniz` , juntamente com `IDH` e `pop_sanea` apenas para visualizar

In [3]:
```
data = data_orig.copy()

# definindo variáveis
attrs = ['urbaniz', 'IDH', 'pop_sanea']

for var in attrs:
    print(var, '-', data[var].dtype)

# eliminando valores nulos
data = data.dropna(subset=attrs)
data
```

```
urbaniz - object
IDH - int64
pop_sanea - float64
```

Out[3]:

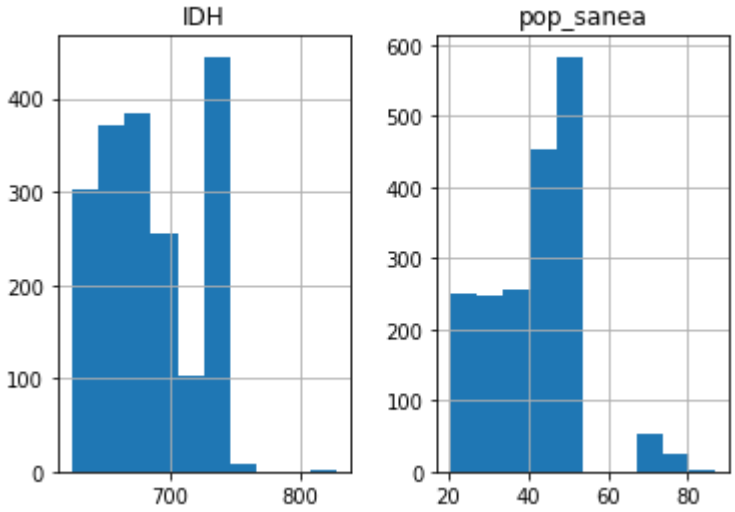
	gid	UF	nome	Censo	PIB	pop	classe	des
0	752	ACRE	Acrelândia	2010.0	151120.015625	12241	2	
1	747	ACRE	Assis Brasil	2010.0	48347.300781	5662	1	
2	748	ACRE	Brasiléia	2010.0	194979.828125	20238	1	
3	754	ACRE	Bujari	2010.0	88708.031250	6772	2	
4	751	ACRE	Capixaba	2010.0	89052.679688	9287	1	
...	...	...	...	...	...	...	...	
5560	1011	TOCANTINS	Tocantinópolis	2010.0	124657.000000	21826	1	
5561	5545	TOCANTINS	Tupirama	2010.0	34883.894531	1474	3	
5562	5546	TOCANTINS	Tupiratins	2010.0	30757.437500	2143	2	
5563	5141	TOCANTINS	Wanderlândia	2010.0	66966.773438	9493	1	
5564	1107	TOCANTINS	Xambioá	2010.0	117627.132812	11099	2	

1870 rows × 14 columns

vamos verificar a distribuição dos dados que temos:

In [4]:
```
data[attrs].hist()
```

Out[4]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x
7f18fbe927f0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x
7f18f9e61d00>]],
      dtype=object)
```



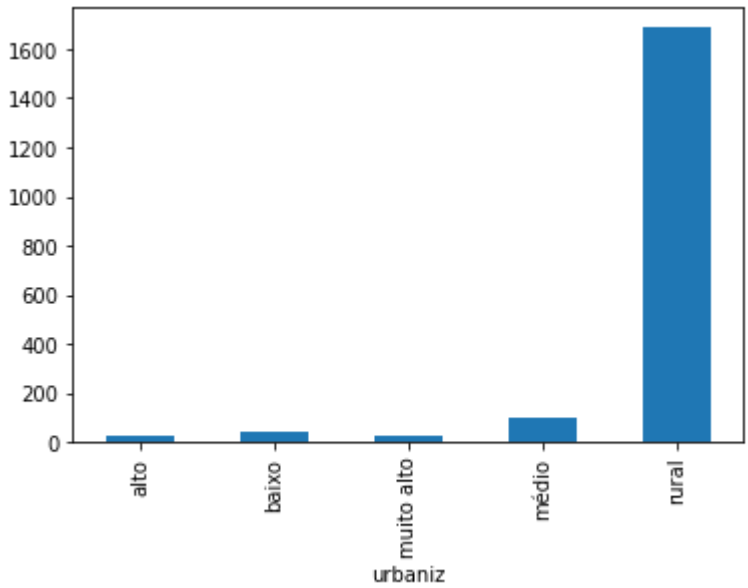
A variável do tipo "objeto" é categórica para fins semânticos.

Por isso é preciso agregar `urbaniz` para plotar o histograma.

OBS: em pandas há um tipo "category" que funciona diferente, veremos abaixo.

In [5]:
```
data.groupby('urbaniz').size().plot(kind='bar')
data['urbaniz'].unique()
```

Out[5]:
```
array(['rural', 'baixo', 'médio', 'muito alto', 'alto'], dt
ype=object)
```



Há várias possibilidades para mapear valores categóricos em numéricos.

Categóricos nominais

Casos em que não há relação de ordenação entre os elementos, podemos usar código ordenado de forma alfabética.

Nulos são representados por -1

1. usando código associado ao tipo category (pandas)

In [6]:
```
data['urban_cod'] = data['urbaniz'].astype("category").cat.
codes
data
```

Out[6]:

	gid	UF	nome	Censo	PIB	pop	classe	des
0	752	ACRE	Acrelândia	2010.0	151120.015625	12241	2	
1	747	ACRE	Assis Brasil	2010.0	48347.300781	5662	1	
2	748	ACRE	Brasiléia	2010.0	194979.828125	20238	1	
3	754	ACRE	Bujari	2010.0	88708.031250	6772	2	
4	751	ACRE	Capixaba	2010.0	89052.679688	9287	1	
...	...	...	...	...	...	...	...	
5560	1011	TOCANTINS	Tocantinópolis	2010.0	124657.000000	21826	1	
5561	5545	TOCANTINS	Tupirama	2010.0	34883.894531	1474	3	
5562	5546	TOCANTINS	Tupiratins	2010.0	30757.437500	2143	2	
5563	5141	TOCANTINS	Wanderlândia	2010.0	66966.773438	9493	1	
5564	1107	TOCANTINS	Xambioá	2010.0	117627.132812	11099	2	

1870 rows × 15 columns

Uma desvantagem é que usamos um critério numérico arbitrário (ordem alfabética), mas pode ser que não faça sentido dizer que a diferença entre os valores faça sentido.

Exemplo, a diferença entre: "rural" e "média" é 1

O que isso significa?

2. usando map() por meio de dicionário

Outra opção, mais genérica, é utilizar um mapa.

Abaixo vamos usar de 0 até 4, sendo 0 = rural e 4 = muito alto

In [7]:
```
cat_urban = data['urbaniz'].unique()
num_urban = np.arange(data['urbaniz'].unique().shape[0])
print(cat_urban)
print(num_urban)

map_urban = dict(zip(cat_urban, num_urban))
print("\nDicionário:")
print(map_urban)
```

```
['rural' 'baixo' 'médio' 'muito alto' 'alto']
[0 1 2 3 4]

Dicionário:
{'rural': 0, 'baixo': 1, 'médio': 2, 'muito alto': 3, 'alto
': 4}
```

A partir do dicionário podemos criar um mapa entre os valores existentes e novos valores, agora numéricos.

Essa opção é uma das mais rápidas.

In [11]:
```
data['urban_map'] = data['urbaniz'].map(map_urban)
data
```

Out[11]:

	gid	UF	nome	Censo	PIB	pop	classe	des
0	752	ACRE	Acrelândia	2010.0	151120.015625	12241	2	
1	747	ACRE	Assis Brasil	2010.0	48347.300781	5662	1	
2	748	ACRE	Brasiléia	2010.0	194979.828125	20238	1	
3	754	ACRE	Bujari	2010.0	88708.031250	6772	2	
4	751	ACRE	Capixaba	2010.0	89052.679688	9287	1	
...	...	...	...	...	...	...	...	
5560	1011	TOCANTINS	Tocantinópolis	2010.0	124657.000000	21826	1	
5561	5545	TOCANTINS	Tupirama	2010.0	34883.894531	1474	3	
5562	5546	TOCANTINS	Tupiratins	2010.0	30757.437500	2143	2	
5563	5141	TOCANTINS	Wanderlândia	2010.0	66966.773438	9493	1	
5564	1107	TOCANTINS	Xambioá	2010.0	117627.132812	11099	2	

1870 rows × 17 columns

Note que, mesmo ordenado, as diferenças podem não fazer tanto sentido.

Nesse caso a diferença entre "rural" e "média" é 2

Categóricos ordinais

Para cenários em que as categorias podem ser ordenadas, como notas, faixas salariais, escalas, podemos usar ordenação categórica:

In [12]:
```
cat_ord_urban = ['rural', 'baixo', 'médio', 'alto', 'muito
alto']
print(cat_ord_urban)

urban_ord_type = pd.api.types.CategoricalDtype(categories=c
at_ord_urban, ordered=True)
data['urban_ord'] = data['urbaniz'].astype(urban_ord_type)
```

```
['rural', 'baixo', 'médio', 'alto', 'muito alto']
```

In [13]:
```
urban_atts = ['urbaniz', 'urban_cod', 'urban_map', 'urban_o
rd']
for var in urban_atts:
    print(var, '-', data[var].dtype)

data['urban_ord']
```

```
urbaniz - object
urban_cod - int8
urban_map - int64
urban_ord - category
```

Out[13]:
```
0       rural
1       rural
2       baixo
3       rural
4       rural
        ...
5560    rural
5561    rural
5562    rural
5563    rural
5564    rural
Name: urban_ord, Length: 1870, dtype: category
Categories (5, object): [rural < baixo < médio < alto < mui
to alto]
```

In [14]:
```
data.loc[:13, urban_atts]
```

Out[14]:

	urbaniz	urban_cod	urban_map	urban_ord
0	rural	4	0	rural
1	rural	4	0	rural
2	baixo	1	1	baixo
3	rural	4	0	rural
4	rural	4	0	rural
5	médio	3	2	médio
6	rural	4	0	rural
7	baixo	1	1	baixo
8	rural	4	0	rural
9	baixo	1	1	baixo
10	rural	4	0	rural
11	rural	4	0	rural
12	baixo	1	1	baixo
13	rural	4	0	rural

Aqui, temos a ordenação codificada nos metadados, mas não há operadores!

In [24]:
```
# subtracao "medio" - "rural"
#data.loc[5, 'urban_ord']  - data.loc[0, 'urban_ord']

# comparacao "medio" > "rural"
#if (data.loc[5, 'urban_ord'] > data.loc[0, 'urban_ord']):
#    print("médio > rural")
```

Codificação *one-hot* ou *dummy variables*

Transforma cada possível categoria em uma variável binária indicando presença ou não de um valor.

Em pandas é possível usar a função `get_dummies()`

In [25]:
```
dummy_vars = pd.get_dummies(data['urbaniz'])

data = pd.concat([data, dummy_vars], axis=1, sort=False)

atts_dummy = ['urbaniz'] + list(cat_urban)
print(atts_dummy)

data.loc[:13 , atts_dummy]
```

```
['urbaniz', 'rural', 'baixo', 'médio', 'muito alto', 'alto
']
```

Out[25]:

	urbaniz	rural	baixo	médio	muito alto	alto
0	rural	1	0	0	0	0
1	rural	1	0	0	0	0
2	baixo	0	1	0	0	0
3	rural	1	0	0	0	0
4	rural	1	0	0	0	0
5	médio	0	0	1	0	0
6	rural	1	0	0	0	0
7	baixo	0	1	0	0	0
8	rural	1	0	0	0	0
9	baixo	0	1	0	0	0
10	rural	1	0	0	0	0
11	rural	1	0	0	0	0
12	baixo	0	1	0	0	0
13	rural	1	0	0	0	0

Resumo:

- Pode ser preciso codificar variáveis categóricas em diferentes formatos
- Considerar a natureza dos dados e sua aplicação para definir o tipo destino:
  - categórico ordinal
  - codificação nominal
  - codificação ordinal
  - one-hot / dummy variables