MBA em Ciência de Dados

Técnicas Avançadas de Captura e Tratamento de Dados

Módulo III - Aquisição e Transformação de Dados

Discretização

Material Produzido por Moacir Antonelli Ponti

CeMEAI - ICMC/USP São Carlos

Conteúdo:

1. Intervalo ou histograma: *binning*
2. Agrupamento

Referência complementar

DIEZ, David M.; BARR, Christopher D.; CETINKAYA-RUNDEL, Mine. **OpenIntro statistics**. 3.ed. OpenIntro, 2015. Capítulo 1.

Discretização

Valores contínuos podem representar um desafio na análise de dados

- alguns métodos não permitem o uso de valores contínuos, seja como atributo de entrada ou saída.
- é possível que aprender segundo uma hipótese de valor contínuo seja inviável devido à quantidade de dados disponível

Discretizar é criar um novo atributo discreto (com valores finitos e bem definidos) a partir de um atributo contínuo.

- podemos também re-aplicar discretização em dados já discretos, reduzindo a quantidade de valores possíveis

---

Vamos usar uma base com dados do PIB e população (dados reais) e outros indicadores (simulados a partir de dados reais) formulada para exemplificar esse conceito, contendo os seguintes atributos:

- gid - identificador geográfico do município
- UF - unidade federativa
- nome - nome do município
- Censo - ano do censo relativo aos dados
- PIB - total do PIB
- pop - populacao em 2009
- classe - classe do município (de 1 a 5)
- desemprego - índice de desemprego na cidade no ano do Censo
- pop_sanea - porcentagem da população servida por saneamento básico
- expec_vida - expectativa de vida ao nascer no ano de 2017
- pobreza - porcentagem de pessoas em extrema pobreza
- IDH - índice em 2010
- urbaniz - escala de urbanização do município: rural, baixo, médio, alto, muito alto
- dens_pop_urbana - índice de densidade populacional urbana: baixa, média, alta, muito alta

OBS: desemprego, IDH, pobreza e pop_sanea por município foram simulados com base nos dados reais dos estados

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

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

Out[1]:

	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	

5565 rows × 14 columns

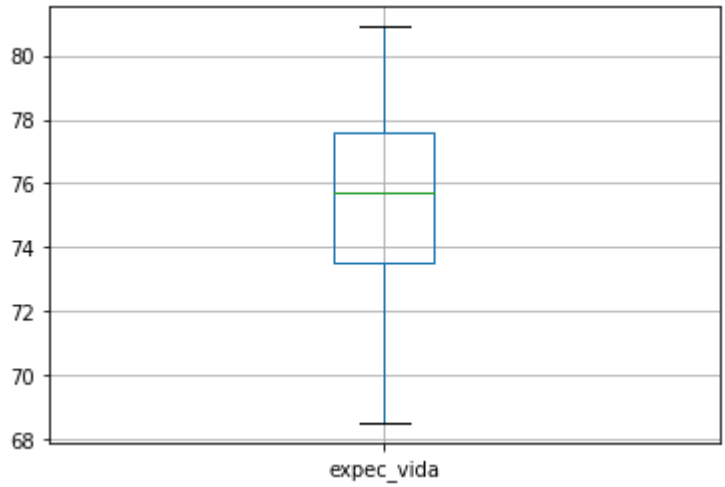
Intervalo ou Histograma

Método que usa intervalos (*bins*) e atribui elementos em cada intervalo um novo valor.

Vamos analisar o atributo expec_vida e discretizá-lo utilizando intervalos

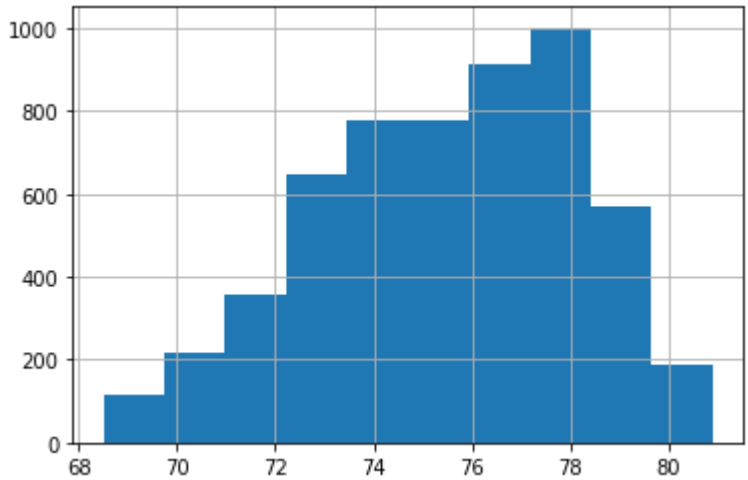
In [2]:
```
atts = 'expec_vida'
data.boxplot(atts)
print(data[atts].unique())
```

```
[73.6 74.2 75.3 73.4 75.4 72.6 72.5 74.6 73.3 74.8 74.9 74.
1 73.7 75.2
 73.1 72.  72.8 71.8 70.6 70.7 73.5 72.7 71.7 70.5 71.4 71.
3 71.1 72.4
 71.9 73.  70.9 73.2 71.6 72.1 71.  72.2 70.8 71.5 72.9 71.
2 73.8 75.
 74.3 74.4 74.  72.3 75.1 73.9 74.7 74.5 75.6 75.5 79.3 78.
77.8 77.2
 78.6 79.  78.4 77.1 79.4 78.3 77.5 77.9 78.5 77.3 77.  79.
8 79.1 77.6
 79.6 79.5 77.7 79.7 77.4 78.8 79.2 78.7 80.  78.1 78.2 78.
9 75.7 75.8
 68.7 68.9 68.8 70.2 69.1 69.  70.3 69.8 70.4 69.5 68.6 69.
7 69.6 70.1
 69.9 69.4 69.2 70.  69.3 68.5 75.9 76.  76.4 76.6 76.2 76.
9 76.1 76.8
 76.5 76.7 76.3 80.5 80.1 79.9 80.4 80.9 80.8 80.6 80.3 80.
2 80.7]
```



In [3]:
```
data[atts].hist()
```

Out[3]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd12f5e50>
```



In [4]:
```
# copiar original
data_orig = data.copy()

# definir intervalos
interv_idade = np.arange(68,82,3)
print(interv_idade)
```

```
[68 71 74 77 80]
```

In [5]:
```
# realizar discretizacao e armazenar
expec_vida_disc =  pd.cut(data['expec_vida'], bins=interv_i
dade)
# inserir nova coluna
data.insert(10, 'expec_vida_disc', expec_vida_disc)

# exibir o tipo da coluna
print(data['expec_vida_disc'].dtype.name)
```

```
category
```

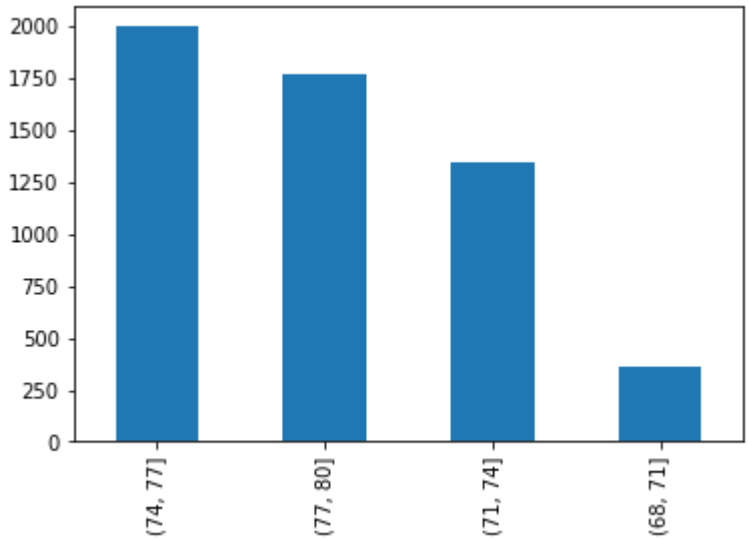
In [7]:
```
# tentando exibir histograma da nova variavel
#data['expec_vida_disc'].hist()
```

In [8]:
```
# para variáveis categoricas podemos usar value_counts
data['expec_vida_disc'].value_counts()
```

Out[8]:
```
(74, 77]    1999
(77, 80]    1770
(71, 74]    1348
(68, 71]     360
Name: expec_vida_disc, dtype: int64
```

In [9]:
```
data['expec_vida_disc'].value_counts().plot(kind='bar')
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd0b81490>
```



Outra opção é nomear os intervalos:

In [11]:
```
interv_idade = np.arange(68,82,3)
labels = ['muito baixo', 'baixo', 'médio', 'alto']
print(interv_idade)
print(labels)
```

```
[68 71 74 77 80]
['muito baixo', 'baixo', 'médio', 'alto']
```

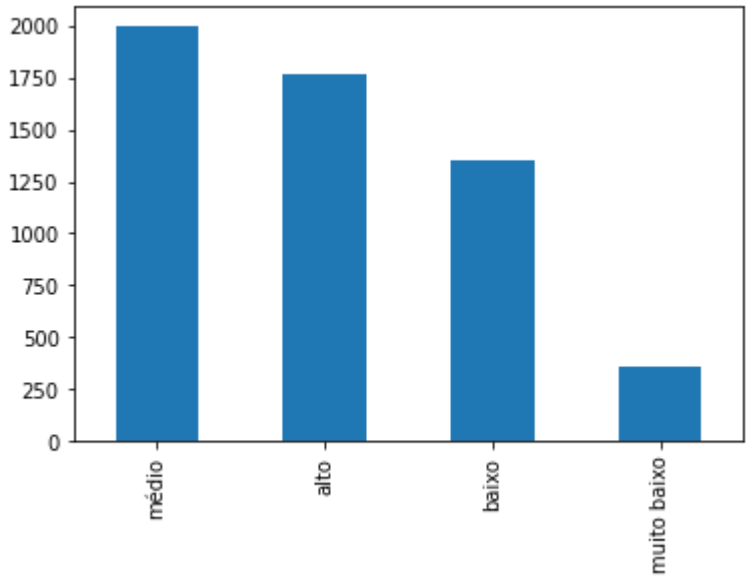
In [12]:
```
data = data_orig.copy()

# realizar discretizacao e armazenar
expec_vida_disc =  pd.cut(data['expec_vida'], bins=interv_i
dade, labels=labels)
# inserir nova coluna
data.insert(10, 'expec_vida_disc', expec_vida_disc)

# exibir o tipo da coluna
print(data['expec_vida_disc'].dtype.name)
data['expec_vida_disc'].value_counts()
```

```
category
```

Out[12]:
```
médio          1999
alto           1770
baixo          1348
muito baixo     360
Name: expec_vida_disc, dtype: int64
```

In [13]:
```
data['expec_vida_disc'].value_counts().plot(kind='bar')
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd0aef580>
```



In [14]:
```
data.groupby('UF').expec_vida_disc.value_counts()
```

Out[14]:

UF	expec_vida_disc	
ACRE	médio	12
	baixo	10
ALAGOAS	baixo	87
	muito baixo	15
AMAPÁ	baixo	9
	médio	7
AMAZONAS	baixo	56
	muito baixo	6
BAHIA	baixo	246
	médio	171
CEARÁ	médio	97
	baixo	87
DISTRITO FEDERAL	alto	1
ESPIRITO SANTO	alto	76
	médio	2
GOIÁS	médio	150
	baixo	96
MARANHÃO	muito baixo	191
	baixo	26
MATO GROSSO	médio	93
	baixo	48
MATO GROSSO DO SUL	médio	70
	alto	8
MINAS GERAIS	alto	539
	médio	314
PARANÁ	alto	238
	médio	161
PARAÍBA	baixo	145
	médio	78
PARÁ	baixo	135
	muito baixo	8
PERNAMBUCO	médio	96
	baixo	89
PIAUÍ	baixo	112
	muito baixo	112
RIO DE JANEIRO	médio	63
	alto	29
RIO GRANDE DO NORTE	médio	145
	alto	22
RIO GRANDE DO SUL	médio	450
	alto	46
RONDÔNIA	baixo	26
	muito baixo	26
RORAIMA	baixo	13
	muito baixo	2
SANTA CATARINA	alto	205
SERGIPE	baixo	66
	médio	9
SÃO PAULO	alto	606
	médio	39
TOCANTINS	baixo	97
	médio	42

Name: expec_vida_disc, dtype: int64

Pandas ainda permite discretização baseada em **quantis**

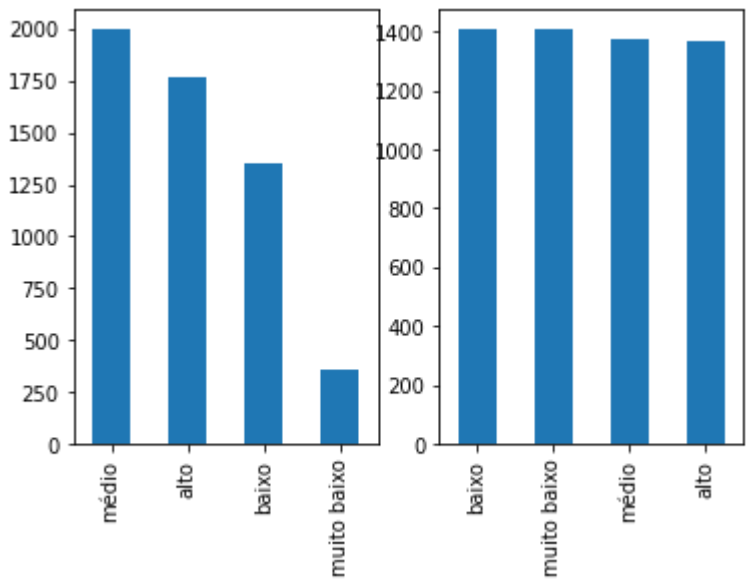
Note que agora usamos o mesmo rótulo, mas a interpretação será diferente, pois ele irá selecionar de forma balanceada cortando pelos *quantis* e não pelos intervalos fixos

In [15]:
```
labels = ['muito baixo', 'baixo', 'médio', 'alto']

# realizar discretizacao e armazenar
expec_vida_qdisc =  pd.qcut(data['expec_vida'], 4, labels=l
abels)
# inserir nova coluna
data.insert(11, 'expec_vida_4quant', expec_vida_qdisc)

# exibir o tipo da coluna
print(data['expec_vida_4quant'].value_counts())
```

```
baixo          1409
muito baixo    1408
médio          1378
alto           1370
Name: expec_vida_4quant, dtype: int64
```

In [16]:
```
plt.subplot(121)
data['expec_vida_disc'].value_counts().plot(kind='bar')
plt.subplot(122)
data['expec_vida_4quant'].value_counts().plot(kind='bar')
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1dd0af7af0>
```



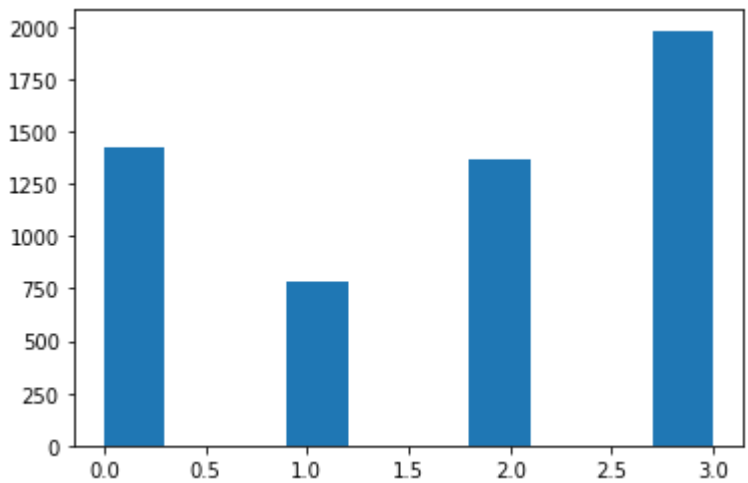
Agrupamento

Método que agrupa valores considerando uma ou mais variáveis e considera valores discretos como o rótulo dos grupos

Vamos analisar o atributo expec_vida e discretizá-lo

In [18]:
```
from sklearn.mixture import GaussianMixture

X = np.array(data['expec_vida']).reshape(-1,1)

gmm = GaussianMixture(n_components=4, random_state=10).fit
(X)

probs = np.round(gmm.predict_proba(X), 3)

clusters = gmm.predict(X)

print(probs[:10,:])
print(clusters[:10])

h = plt.hist(clusters)
```

```
[[0.    0.094 0.    0.906]
 [0.006 0.039 0.    0.955]
 [0.256 0.008 0.004 0.732]
 [0.    0.128 0.    0.872]
 [0.324 0.007 0.005 0.664]
 [0.256 0.008 0.004 0.732]
 [0.    0.41  0.    0.59 ]
 [0.256 0.008 0.004 0.732]
 [0.    0.128 0.    0.872]
 [0.    0.463 0.    0.537]]
[3 3 3 3 3 3 3 3 3 3]
```



In [19]:
```
# atribuindo a nova variável ao dataframe
data['expec_vida_clu'] = clusters

# verificando como ficou o agrupamento em termos de mínimos
e máximos
print(data.groupby('expec_vida_clu').expec_vida.min())
print(data.groupby('expec_vida_clu').expec_vida.max())
```

```
expec_vida_clu
0    75.7
1    68.5
2    77.7
3    72.5
Name: expec_vida, dtype: float64
expec_vida_clu
0    77.6
1    72.4
2    80.9
3    75.6
Name: expec_vida, dtype: float64
```

Resumo:

- Discretização pode ser uma ferramenta importante para gerar novos atributos que sumarizam as informações e permitem análises inviáveis com os atributos originais
- Intervalo:
  - fixo
  - quantis
- Agrupamento:
  - abordagem *data-driven*