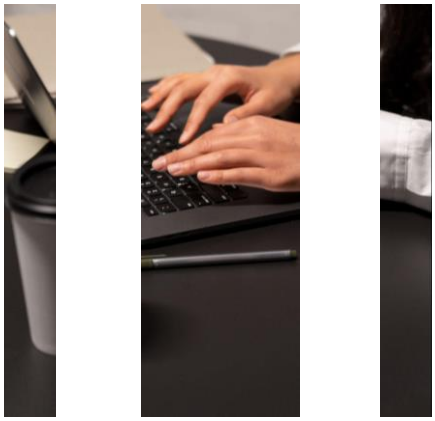
Extração, Tratamento e Carregamento de Dados

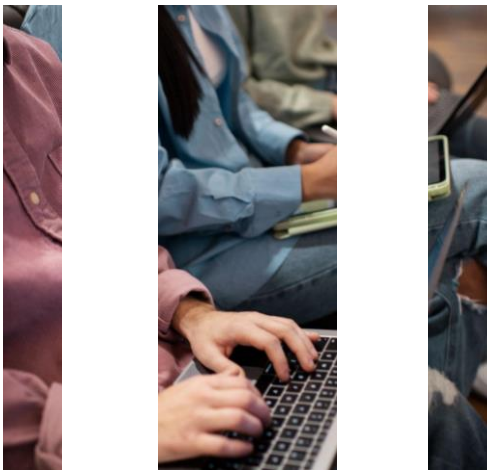


1

05

Módulo 5

Extração, Tratamento e Carregamento de Dados

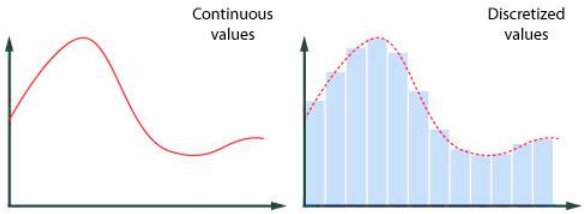


Balduíno Mateus

2

Discretização

- A Discretização reduz o número de valores de um dado atributo contínuo pela divisão da amplitude do atributo em intervalos;



3

Discretização

- Interesse: redução do número de valores. Muito importante em árvores de decisão;

4

Discretização

Alguns algoritmos de Data Mining aceitam apenas valores categóricos

- Procuram discretizar valores contínuos em intervalos.

Melhor discretização depende de:

- Algoritmo que utilizará os valores discretizados;
- Demais atributos
- ...

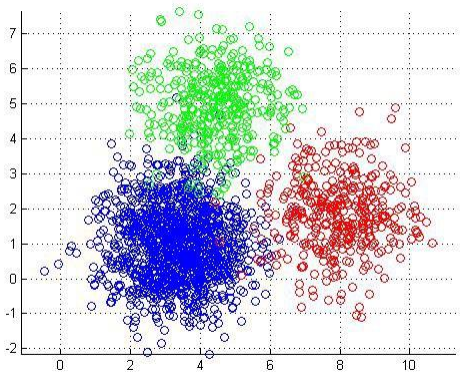
5

Discretização

- Existem vários algoritmos na literatura
- Algoritmos podem ser divididos como:
  - Não supervisionados
    - utilizam somente os valores do atributo a ser discretizado.
  - Supervisionados
    - direcionados para classificação usam informação das classes das respetivas instâncias.

6

Discretização



Inspeção Visual

- Observa gráfico com valores dos atributos e determina visualmente os intervalos de acordo com a distribuição natural dos dados

Clustering

- Utiliza algum algoritmo de agrupamento de dados para descobrir automaticamente a distribuição dos dados

7

Discretização

Dados contínuos com frequência são discretizados ou, de modo alternativo, separados em compartimentos (bins) para análise.

A função pandas cut() é usada para separar os elementos da matriz em diferentes compartimentos.

8

Discretização no pandas

Considere o caso de alguns dados uniformemente distribuídos, divididos em quartos:

```
data = pd.DataFrame({'Data': np.random.rand(20)})
data.head()
```

	Data
0	0.838006
1	0.583263
2	0.551771
3	0.356790
4	0.451201

```
pd.cut(data['Data'],4)
```

```
0    (0.696, 0.916]
1    (0.475, 0.696]
2    (0.475, 0.696]
...
18   (0.0333, 0.255]
19    (0.475, 0.696]
```

```
pd.value_counts(cats)
```

```
(0.255, 0.475] 7
(0.475, 0.696] 6
(0.696, 0.916] 4
(0.0333, 0.255] 3
Name: Data,
dtype: int64
```

9

Discretização

Uma função intimamente relacionada, qcut, compartimenta os dados com base nos quantis da amostra.

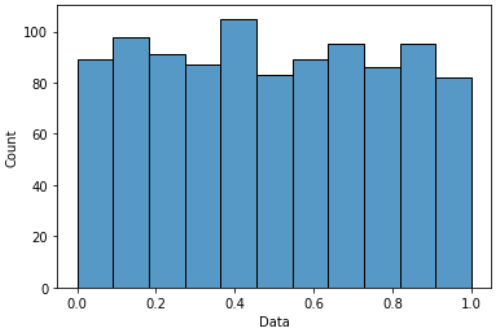
Conforme a distribuição dos dados, usar cut em geral não resultará em cada compartimento com o mesmo número de pontos de dados.

Como qcut utiliza quantis da amostra, por definição, terá como resultado compartimentos rigorosamente do mesmo tamanho:

- data = pd.DataFrame({'Data': np.random.rand(1000)})

10

Discretização




```
cats = pd.qcut(data['Data'],4)
```

```
0       (0.243, 0.496]
1    (-0.0009638, 0.243]
2       (0.243, 0.496]
3       (0.243, 0.496]
4       (0.751, 0.999]
           ...
995     (0.496, 0.751]
996     (0.243, 0.496]
997     (0.496, 0.751]
998     (0.751, 0.999]
999  (-0.0009638, 0.243]
```

```
pd.value_counts(cats)
```

```
(0.00145, 0.241]    250
(0.241, 0.511]      250
(0.511, 0.751]      250
(0.751, 0.999]      250
dtype: int64
```

11

Discretização

De modo semelhante a cut, podemos passar os próprios quantis (números entre 0 e 1)

- cats=pd.qcut(data['Data'],[0,0.1,0.5,0.9,1])
- pd.value_counts(cats)
  - (0.107, 0.496] 400
  - (0.496, 0.893] 400
  - (-0.0009638, 0.107] 100
  - (0.893, 0.999] 100

12

13

Discretização função cut

- Podemos utilizar a função 'CUT' de duas maneiras: especificando diretamente o número de BINS e deixando o pandas fazer o trabalho de calcular BINS de tamanho igual, ou podemos especificar manualmente as bordas de BIN como desejarmos.

Discretização

Importação da base de dados

	Nome	salário	Idade	Tempo_de_trabalho	Avaliação Do Empregador
0	Maria	900	27	5	7
1	António	800	33	3	10
2	Luís	750	38	5	14
3	Ana	1500	21	3	9
4	Marques	1400	45	6	15

14

Discretização

```
bins = [18, 25, 35, 60, 100]
data['Intervalos'] = pd.cut(data['Idade'], bins)
pd.value_counts(data['Intervalos'])
```

	Nome	salário	Idade	Tempo_de_trabalho	Avaliação Do Empregador	Intervalos
0	Maria	900	27	5	7	(25, 35]
1	António	800	33	3	10	(25, 35]
2	Luís	750	38	5	14	(35, 60]
3	Ana	1500	21	3	9	(18, 25]
4	Marques	1400	45	6	15	(35, 60]

15

Discretização

```
data['Intervalos'] = pd.cut(data['Idade'], bins, labels=['Jovem','Adulto', 'Meia Idade', 'Mais velho'] )
pd.value_counts(data['Intervalos'])
data.head()
```

	Nome	salário	Idade	Tempo_de_trabalho	Avaliação Do Empregador	Intervalos
0	Maria	900	27	5	7	Adulto
1	António	800	33	3	10	Adulto
2	Luís	750	38	5	14	Meia Idade
3	Ana	1500	21	3	9	Jovem
4	Marques	1400	45	6	15	Meia Idade

16

Discretização

```
data['Intervalos'] = pd.cut(data['Idade'],4 ,labels = Nom_grupo)
data.head()
```

	Nome	salário	Idade	Tempo_de_trabalho	Avaliação Do Empregador	Intervalos
0	Maria	900	27	5	7	Mais Novo
1	António	800	33	3	10	Mais Novo
2	Luís	750	38	5	14	Meia Idade
3	Ana	1500	21	3	9	Mais Novo
4	Marques	1400	45	6	15	Meia Idade

17

Discretização

```
data['Tempo']= (data['Tempo_de_trabalho'] >= 3).astype(int)
```

	Nome	salário	Idade	Tempo_de_trabalho	Avaliação Do Empregador	Tempo
5	Smith	2000	65	8	8	1
6	Marques	1400	45	6	18	1
7	Ana	800	36	2	13	0
8	Lucélio	1150	45	6	6	1
9	Waldemar	2000	63	20	18	1

18

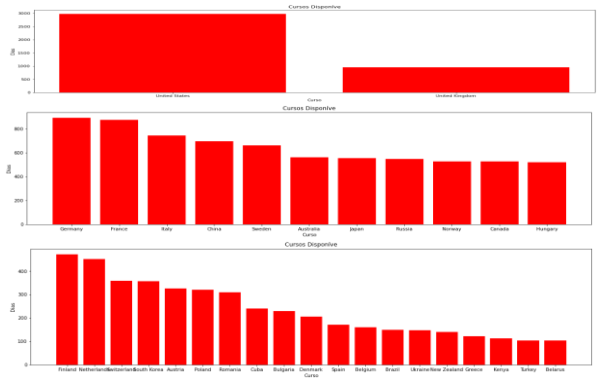
Exercícios 19

- Com o dataframe anterior divide os resultados da avaliação em 4 intervalos [Mau, Suficiente, Bom e Muito Bom], expõe os funcionários que apresentaram bom e muito bom desempenho na avaliação. destes, apresenta quais são os candidatos que apresentam o ano de experiência inferior a 3 anos.
- Consoante a nota de avaliação bom e muito bom eleva 15% dos salário dos funcionários que ganham nemos ou igual de 800 euros e reduza o mesmo equivalente para os funcionários que ganham a cima ou igual de 1000 que tiveram mau e suficiente.
- Apresenta um novo dataframe com salário atualizado, exporta mesmo para formato csv.
- Apresenta o gráfico de barras dos funcionários com os seus respetivos salários.

19

Exercícios 20

- Cria um programa que subdivide os países por top 4, 3, 2 ,1, usando os intervalos [0, 700, 800, 900, 3000] em função da totalidade de medalhas ganhas.
- Apresenta graficamente os países que estão entre o top1 , top2 e top3.



20