Agrupamento: k-médias

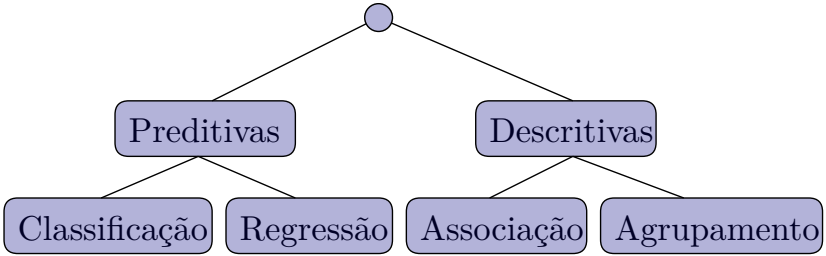
Alexandre Checoli Choueiri

15/04/2023

Conteúdo

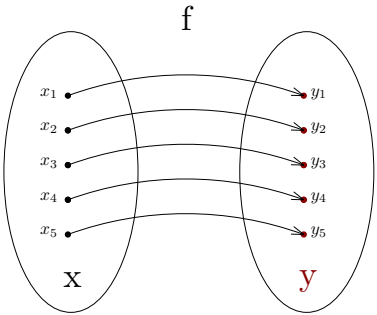
Introdução

As tarefas de mineração de dados podem ser separadas em 2 grandes grupos: tarefas **preditivas** e **descritivas**. Por sua vez, as tarefas descritivas são agrupadas em tarefas de **Associação** e de **Agrupamento**. As tarefas descritivas estão ligadas ao conceito de *aprendizagem não supervisionada*.



Introdução

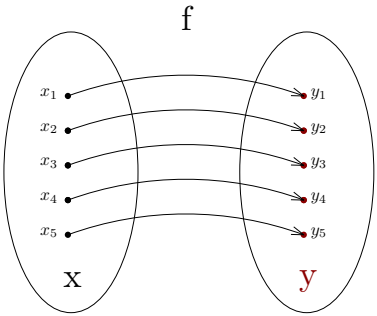
Tarefas da mineração



Relembrando: o termo **supervisionado** se relaciona com a função que queremos encontrar nas tarefas preditivas, temos o valor de y, que é o valor "certo" ou target, daí o termo supervisão.

Introdução

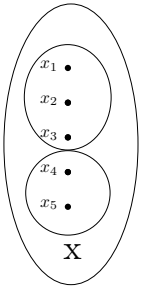
Tarefas da mineração



Queremos fazer uma estimativa, porém possuímos o valore correto para fornecer a supervisão para o algoritmo.

Introdução

Tarefas da mineração



Já no aprendizado **não supervisionado**, não existe um conjunto y (*target*), e o objetivo é extrair informações/padrões de todos os atributos (colunas).

Definição

Agrupamento/clustering: Dado um conjunto de pontos n-dimensionais, o agrupamento visa criar subconjuntos dos pontos, de forma que os elementos dentro de um subconjunto mantenham alguma semelhança entre si, e em relação a elementos de outros conjuntos alguma diferença.

Definição

Agrupamento/clustering: Dado um conjunto de pontos n-dimensionais, o agrupamento visa criar subconjuntos dos pontos, de forma que os elementos dentro de um subconjunto mantenham alguma semelhança entre si, e em relação a elementos de outros conjuntos alguma diferença.

Essa definição é muito genérica, dando espaço para diversos tipos de agrupamento.

Definição

Agrupamento/clustering: Dado um **conjunto de pontos n-dimensionais**, o agrupamento visa criar subconjuntos dos pontos, de forma que os elementos dentro de um subconjunto mantenham alguma semelhança entre si, e em relação a elementos de outros conjuntos alguma diferença.

OBS: O que é um ponto n-dimensional?

Definição

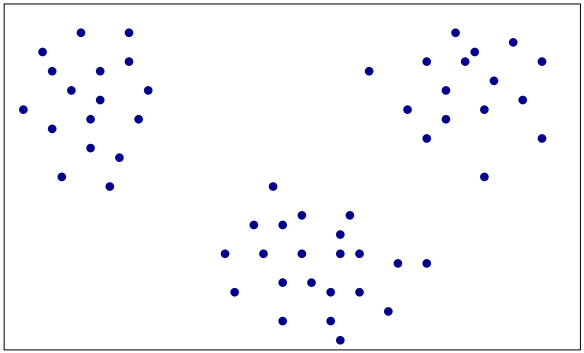
Agrupamento/clustering: Dado um **conjunto de pontos n-dimensionais**, o agrupamento visa criar subconjuntos dos pontos, de forma que os elementos dentro de um subconjunto mantenham alguma semelhança entre si, e em relação a elementos de outros conjuntos alguma diferença.

OBS: O que é um ponto n-dimensional? Estamos acostumados com pontos em duas dimensões (p = [2,4]), porém podemos ter pontos no espaço n-dimensional (p = [2,3,...,n]).

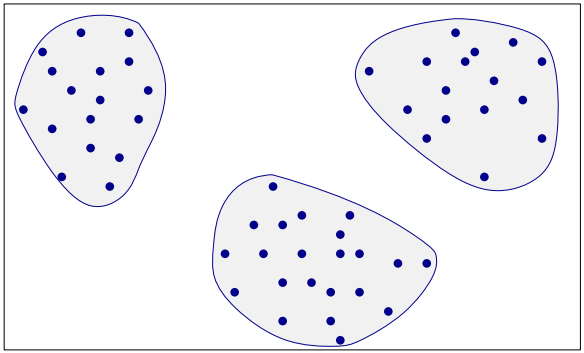
O cérebro humano faz um ótimo trabalho de agrupamento, **para aqueles elementos que são visuais**.

Introdução

Tipos de agrupamento



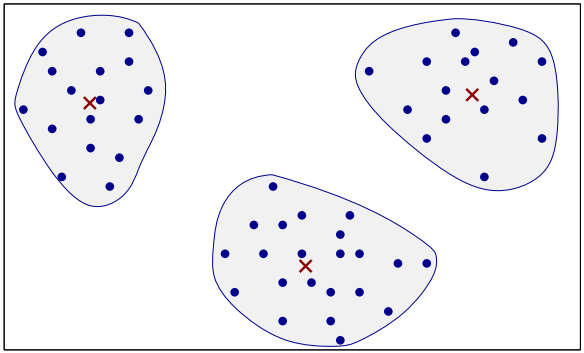
O cérebro humano faz um ótimo trabalho de agrupamento, **para aqueles elementos que são visuais**.Considerando o conjunto de pontos em 2 dimensões acima, você consegue definir grupos de pontos semelhantes?



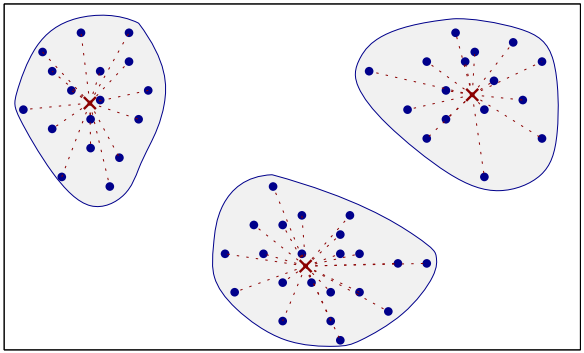
É óbvio que conseguimos...um trabalho um pouco mais difícil é tentarmos identificar **qual a lógica** que o nosso cérebro usou para identificar tais grupos.

Introdução

Tipos de agrupamento



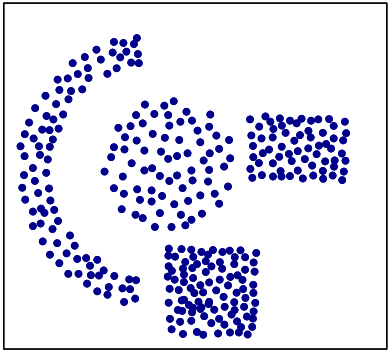
Podemos pensar em um centro para cada grupo, de forma que os elementos desse grupo estão mais próximos do seu centro do que de qualquer outro centro.



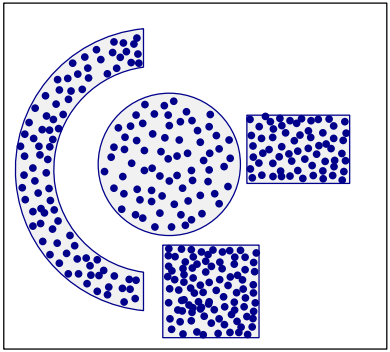
Esse tipo de agrupamento é o mais comum, e é conhecido como **agrupamento por protótipos**, pois o centro de cada grupo é uma representação de todos os pontos (uma média), ou seja, é o protótipo do grupo.

Introdução

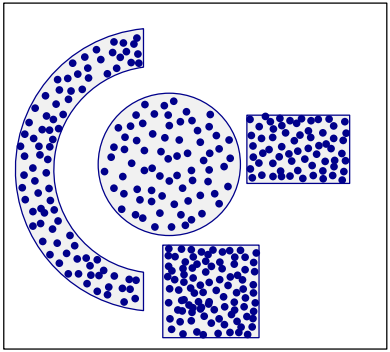
Tipos de agrupamento



Considere os pontos acima. Podemos usar a mesma lógica anterior para criar os grupos neste exemplo?



NÃO. Neste caso podemos pensar em uma lógica de proximidade com vizinhos. Os pontos em um grupo estão mais próximos de outro ponto no grupo do que em qualquer outro ponto de outros grupos.



Esse agrupamento é conhecido como **agrupamento por densidade**.

Dessa forma vemos que o conceito de grupo depende muito do que se deseja e das características dos dados. Para cada definição de grupo existem diferentes algoritmos que podem ser usados. Para o caso mais comum (agrupamento por protótipos), podemos usar o algoritmo **k-means**.

O algoritmo *k-means*

O algoritmo *k-means*

Idéia geral

O algoritmo **k-mean**s funciona de forma iterativa: inicialmente k protótipos são gerados aleatóriamente, e os pontos mais próximos de cada protótipos são a ele atribuídos. A cada iteração, a posição de cada protótipo é atualizada como a média de todos os pontos a ela atribuídos. O algoritmo para quando não houver mais atualização na posição dos protótipo, ou essa for irrisória.

O algoritmo *k-means*

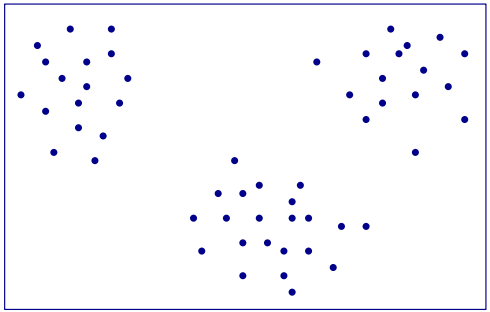
Idéia geral

O algoritmo **k-mean**s funciona de forma iterativa: inicialmente k protótipos são gerados aleatóriamente, e os pontos mais próximos de cada protótipos são a ele atribuídos. A cada iteração, a posição de cada protótipo é atualizada como a média de todos os pontos a ela atribuídos. O algoritmo para quando não houver mais atualização na posição dos protótipo, ou essa for irrisória.

OBSERVAÇÃO: O número de protótipos (ou seja, de grupos k) é definido pelo usuário no início do algoritmo.

O algoritmo *k-means*

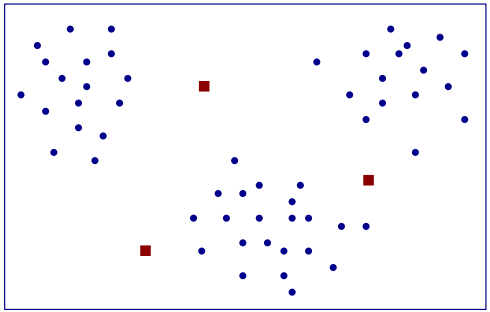
Exemplo



Considere o conjunto de pontos acima, e que iniciamos o algoritmo com k $= 3$ (por sorte!).

O algoritmo *k-means*

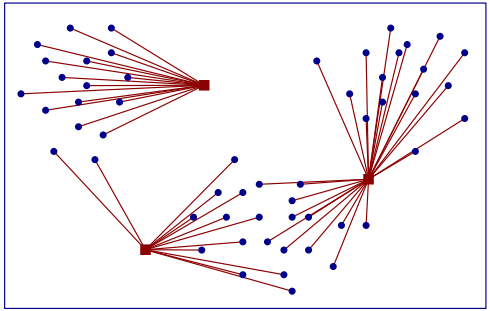
Exemplo



Inicialmente criamos 3 coordenadas (x,y) aleatórias para cada centroide k.

O algoritmo *k-means*

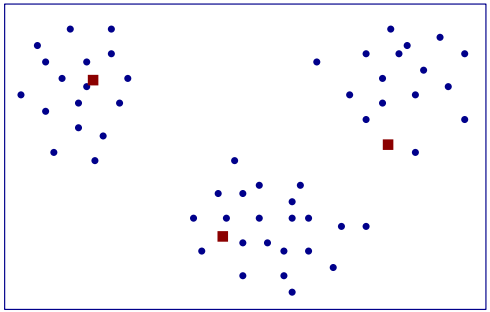
Exemplo



Em seguida calculamos a distância entre todos os pontos e todos os centroides. Cada ponto é atribuído ao centroide mais próximo de si.

O algoritmo *k-means*

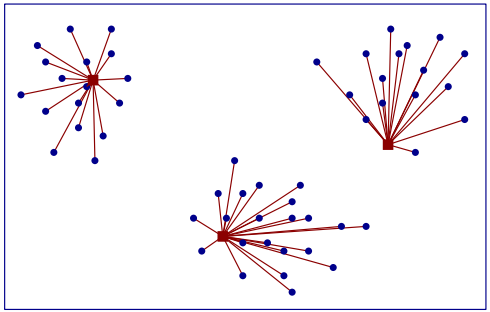
Exemplo



Em seguida as coordenadas dos centroides são atualizadas como a média de todos os pontos a ele atribuídos. É como se os pontos "puxassem"o centroide para seu centro de massa.

O algoritmo *k-means*

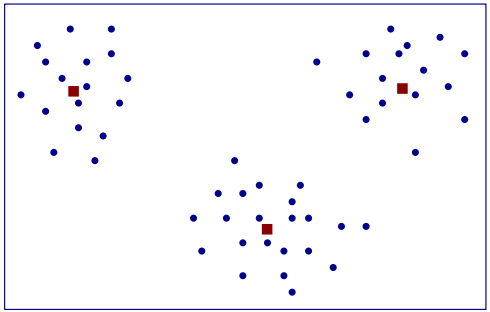
Exemplo



O processo é então repetido...todas as distâncias entre pontos e centroides são calculadas, e os pontos novamente atribuídos ao centroide mais próximo.

O algoritmo *k-means*

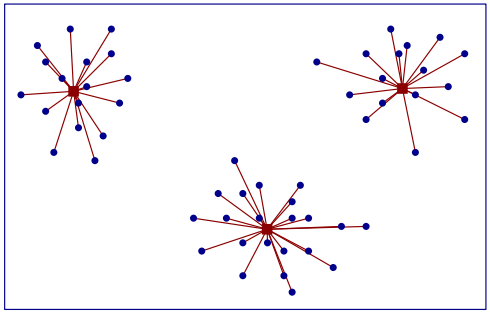
Exemplo



E novamente os centroides são atualizados com as médias dos pontos a eles atribuídos.

O algoritmo *k-means*

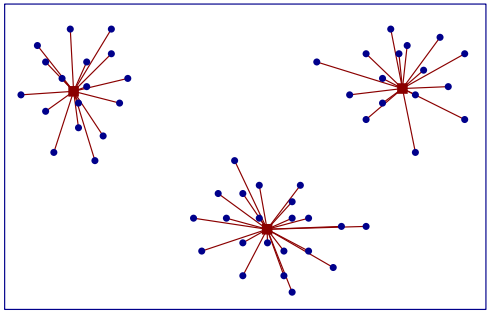
Exemplo



Uma última atribuição de pontos a centroides é feita. Note que se mais uma iteração fosse realizada nenhum centroide mudaria de lugar. Esse pode ser um critério de parada para o algoritmo.

O algoritmo *k-means*

Pseudocódigo



Uma última atribuição de pontos a centroides é feita. Note que se mais uma iteração fosse realizada nenhum centroide mudaria de lugar. Esse pode ser um critério de parada para o algoritmo.

O algoritmo *k-means*

Pseudocódigo

Abaixo é mostrado um pseudocódigo para o *k-means*

Algorithm 1 *k-means*

Selecione k pontos aleatóriamente
while Critério de parada não atingido **do**
    Atribua cada ponto dos dados ao centroide mais próximo
    Recalcule as coordenadas dos k centroides como a média dos pontos a eles atribuídos
end while
return Atribuição de pontos a cada centroide

A distância

A distância

A distância

No algoritmo *k-means* a distância mais utilizada é a distância euclidiana, ou norma L2. Sejam X1 e X2 dois pontos no espaço n dimensional, a distância entre eles é dada por:

$$d(X1, X2) = \sqrt{\sum_{i=1}^{n} (X1_i - X2_i)^2}$$

A distância

A distância

No algoritmo *k-means* a distância mais utilizada é a distância euclidiana, ou norma L2. Sejam X1 e X2 dois pontos no espaço n dimensional, a distância entre eles é dada por:

$$d(X1, X2) = \sqrt{\sum_{i=1}^{n} (X1_i - X2_i)^2}$$

Exemplo 1:
Sejam os pontos X1 = [1,3] e X2 = [2,4].

A distância

A distância

No algoritmo *k-means* a distância mais utilizada é a distância euclidiana, ou norma L2. Sejam X1 e X2 dois pontos no espaço n dimensional, a distância entre eles é dada por:

$$d(X1, X2) = \sqrt{\sum_{i=1}^{n} (X1_i - X2_i)^2}$$

Exemplo 1:
Sejam os pontos X1 = [1,3] e X2 = [2,4].
Temos que n = 2 e $d(X1, X2) = \sqrt{(1-2)^2 + (3-4)^2}$.

A distância

A distância

No algoritmo *k-means* a distância mais utilizada é a distância euclidiana, ou norma L2. Sejam X1 e X2 dois pontos no espaço n dimensional, a distância entre eles é dada por:

$$d(X1, X2) = \sqrt{\sum_{i=1}^{n} (X1_i - X2_i)^2}$$

Exemplo 1:
Sejam os pontos X1 = [1,3] e X2 = [2,4].
Temos que n = 2 e $d(X1, X2) = \sqrt{(1-2)^2 + (3-4)^2}$.
Exemplo 2:
Sejam os pontos X1 = [1,3,5] e X2 = [2,9,2].

A distância

A distância

No algoritmo *k-means* a distância mais utilizada é a distância euclidiana, ou norma L2. Sejam X1 e X2 dois pontos no espaço n dimensional, a distância entre eles é dada por:

$$d(X1, X2) = \sqrt{\sum_{i=1}^{n} (X1_i - X2_i)^2}$$

Exemplo 1:
Sejam os pontos X1 = [1,3] e X2 = [2,4].
Temos que n = 2 e $d(X1, X2) = \sqrt{(1-2)^2 + (3-4)^2}$.

Exemplo 2:
Sejam os pontos X1 = [1,3,5] e X2 = [2,9,2].
Temos que n = 3 e $d(X1, X2) = \sqrt{((1-2)^2 + (3-4)^2)}$

O objetivo

O objetivo

O *k-means* pode ser entendido como um problema de **otimização**, em que o objetivo é atribuir pontos a centróides, de tal forma que soma de todas as distâncias dos pontos a seus centróides seja minimizada.

O objetivo

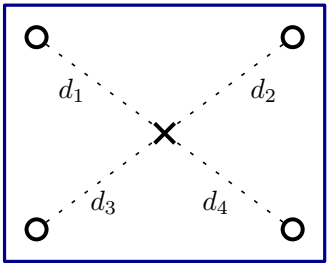
O *k-means* pode ser entendido como um problema de **otimização**, em que o objetivo é atribuir pontos a centróides, de tal forma que soma de todas as distâncias dos pontos a seus centróides seja minimizada.
Uma forma comum (e equivalente à distância) é chamada de SSE (Summed Squared Error), básicamente sendo a soma das distâncias elevadas ao quadrado, dado por:

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} d(C_i, x)^2$$

em que:

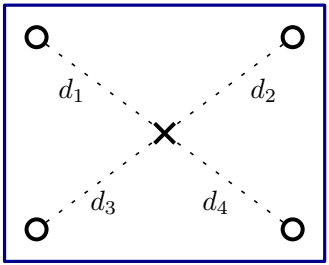
$$\begin{cases} k : \text{número de grupos} \\ C_i : \text{centróide do grupo } i \\ x \in C_i \text{ pontos x atribuídos ao centróide } C_i \end{cases}$$

O objetivo



A vantagem de termos um **objetivo** (minimizar o SSE) para o problema, é que podemos valorar e **comparar diferentes soluções** para um mesmo conjunto de dados. Considere o exemplo acima com 4 pontos e 1 único grupo.

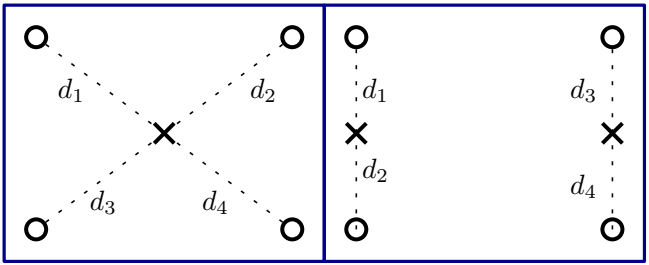
O objetivo



O SSE é dado por:

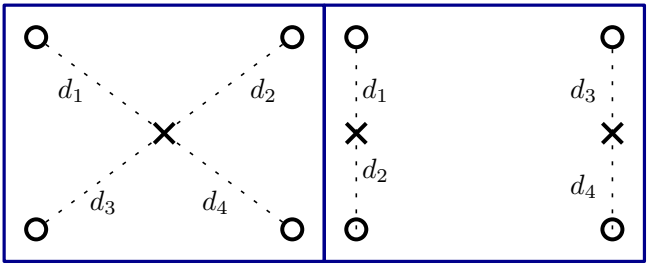
$$SSE = d_1^2 + d_2^2 + d_3^2 + d_4^2$$

O objetivo



Considere agora o segundo agrupamento, com 2 grupos. Considerando o nosso **objetivo**, qual dos dois agrupamentos é melhor?

O objetivo



Considere agora o segundo agrupamento, com 2 grupos. Considerando o nosso **objetivo**, qual dos dois agrupamentos é melhor? Obviamente o segundo agrupamento possui um SSE menor do que o primeiro, portanto ele é **melhor**.

O número de grupos k

O número de grupos k

Mas o que diferenciou o agrupamento 1 do agrupamento 2? O número de *clusters*! Vamos analisar **o que ocorre com o SSE nos limites** possíveis para o número de grupos. Considerando um conjunto de dados com n pontos.

O número de grupos k

Mas o que diferenciou o agrupamento 1 do agrupamento 2? O número de *clusters*! Vamos analisar **o que ocorre com o SSE nos limites** possíveis para o número de grupos. Considerando um conjunto de dados com n pontos.

Qual é o número mínimo de grupos que esse banco pode ter, e qual o SSE para esse número?

O número de grupos k

Mas o que diferenciou o agrupamento 1 do agrupamento 2? O número de *clusters*! Vamos analisar **o que ocorre com o SSE nos limites** possíveis para o número de grupos. Considerando um conjunto de dados com n pontos.

$$k = 1 \rightarrow \text{SSE máximo}$$

Qual é o número mínimo de grupos que esse banco pode ter, e qual o SSE para esse número?

O mínimo de grupos para qualquer banco de dados é 1, e nesse caso o SSE será máximo.

O número de grupos k

Mas o que diferenciou o agrupamento 1 do agrupamento 2? O número de *clusters*! Vamos analisar **o que ocorre com o SSE nos limites** possíveis para o número de grupos. Considerando um conjunto de dados com n pontos.

$$k = 1 \rightarrow \text{SSE máximo}$$
$$k = \text{n} \rightarrow \text{SSE mínimo}$$

Qual é o número mínimo de grupos que esse banco pode ter, e qual o SSE para esse número?

O mínimo de grupos para qualquer banco de dados é 1, e nesse caso o SSE será máximo.

O máximo de grupos para qualquer banco de dados é n, e nesse caso o SSE será mínimo.

O número de grupos k

Percebemos então que o objetivo do nosso problema é um pouco mais sutil.

Cuidado

Para minimizar o SSE, basta **escolhermos $\mathbf{k = n}$**. Porém isso não agrega nenhuma informação, sendo que cada ponto seria seu próprio cluster! Precisamos então **encontrar o número de k** de tal forma que o erro seja minimizado, porém que o agrupamento dos dados nos forneça alguma informações.

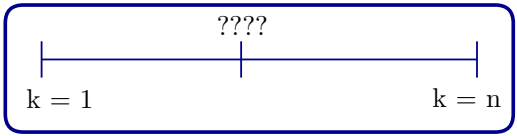
O número de grupos k

Percebemos então que o objetivo do nosso problema é um pouco mais sutil.

Cuidado

Para minimizar o SSE, basta **escolhermos k = n**. Porém isso não agrega nenhuma informação, sendo que cada ponto seria seu próprio cluster! Precisamos então **encontrar o número de k** de tal forma que o erro seja minimizado, porém que o agrupamento dos dados nos forneça alguma informações.

Precisamos então decidir qual o número ideal de grupos (k), entre *1* e *n*.



O número de grupos k

Uma forma muito comum e fácil de determinarmos o número ideal de k, é pelo chamado **gráfico de cotovelo** (elbow-plot).

A premissa do método é a seguinte: sabemos que $k = 1$ gera um SSE muito alto, se executarmos com $k = 2$, esse SSE deve ser reduzido, e assim sucessivamente até $k = n$. No entanto, em algum momento **a diferença de SSE de um número k para um número $k + 1$ vai ser muito pequena**, indicando que **aumentar o k não aumenta significativamente o ganho de informação**.

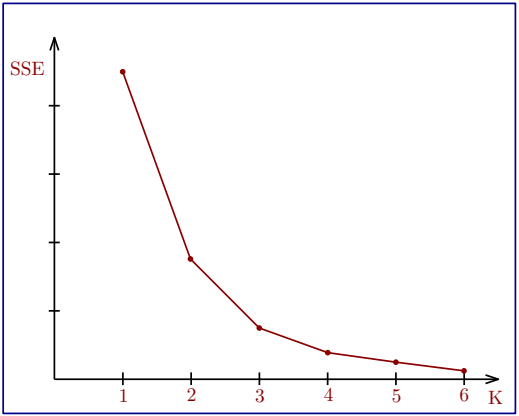
O número de grupos k

Uma forma muito comum e fácil de determinarmos o número ideal de k, é pelo chamado **gráfico de cotovelo** (elbow-plot).

A premissa do método é a seguinte: sabemos que $k = 1$ gera um SSE muito alto, se executarmos com $k = 2$, esse SSE deve ser reduzido, e assim sucessivamente até $k = n$. No entanto, em algum momento **a diferença de SSE de um número k para um número k + 1 vai ser muito pequena**, indicando que **aumentar o k não aumenta significativamente o ganho de informação**.

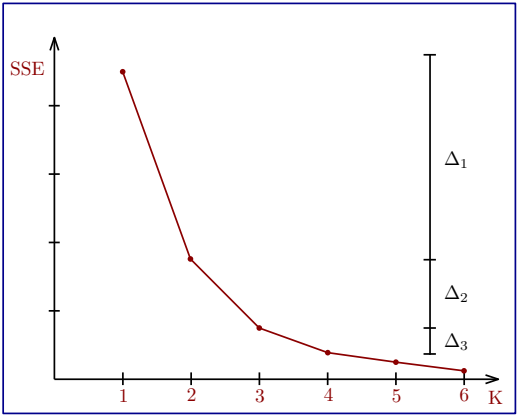
A maneira mais fácil de se verificar esse ponto de corte, é rodar o algoritmo com diversos valores de k (iniciando em 1), coletar os SSEs para cada agrupamento e plotar em um gráfico.

O número de grupos k



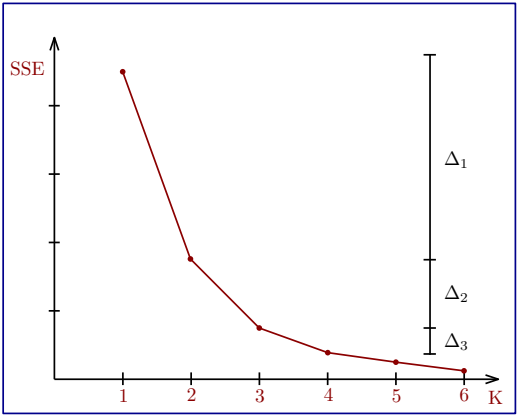
Considere o gráfico acima, em que o gráfico foi construído para valores de k $= 1$ até 6.

O número de grupos k



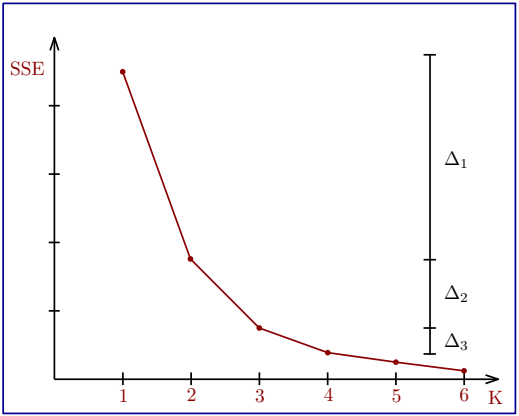
Se analisarmos as diferenças nos SSEs para cada par de ks consecutivos, temos valores de Δ.

O número de grupos k



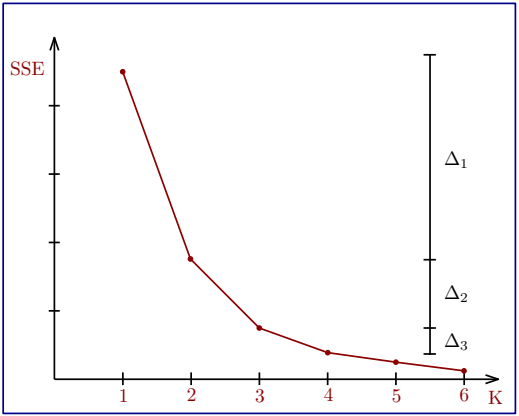
Com $\Delta_1 > \Delta_2 > \Delta_3$, como esperado.

O número de grupos k



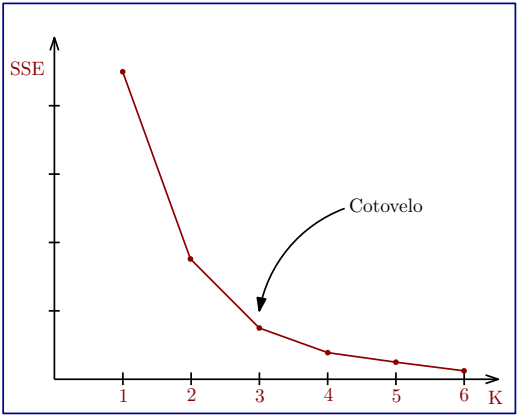
No entanto, parece que a partir de Δ_3 as diferenças são pequenas, e **muito próximas umas das outras**.

O número de grupos k



O que nos permite inferir que o aumento no número de grupos não está contribuindo para uma redução significativa do SSE a partir de k = 3.

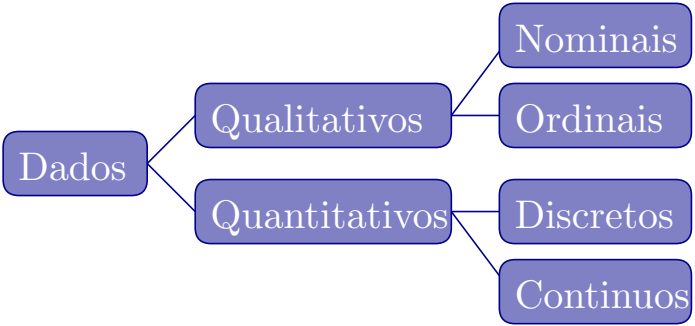
O número de grupos k



Assim, podemos determinar visualmente que o **número ideal de grupos** para esse conjunto é 3 (forma um "cotovelo" no gráfico).

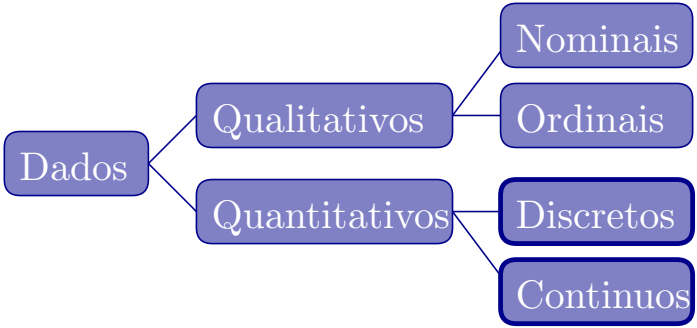
Transformações nos dados

Tipos de dados



De forma simplificada, podemos separar os tipos de dados em 2 grupos: **Qualitativos** e **Quantitativos**.

Tipos de dados



Dados **Quantitativos** são aqueles numéricos. **Discretos** são números inteiros (ex. 1,4,5) e **Contínuo**s são os reais (ex. 1.45, 2.8).

Tipos de dados

Já vimos que o algoritmo *k-means* usa a distância euclidiana para determinar se um ponto está próximo de outro.

$$d(X1, X2) = \sqrt{\sum_{i=1}^{n} (X1_i - X2_i)^2}$$

Para dados quantitativos esse calculo é direto. Considere o banco de dados a seguir, com os dados da idade e do salário de consumidores de um shopping center.

Idade	**Salário**
24	2000
20	1000
60	18000
28	8500
33	6000

Tipos de dados

Já vimos que o algoritmo *k-means* usa a distância euclidiana para determinar se um ponto está próximo de outro.

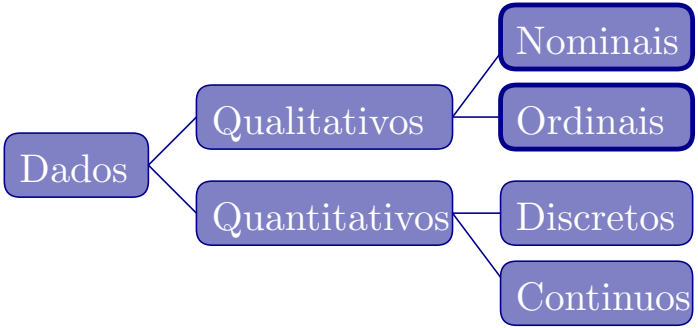
$$d(X1, X2) = \sqrt{\sum_{i=1}^{n}(X1_i - X2_i)^2}$$

Para dados quantitativos esse calculo é direto. Considere o banco de dados a seguir, com os dados da idade e do salário de consumidores de um shopping center.

Idade	**Salário**
24	2000
20	1000
60	18000
28	8500
33	6000

Para calcularmos a distancia entre os dois primeiro clientes, temos que: X1 = [24,2000] e X2 = [20,1000]. Basta aplicar a fórmula.

Tipos de dados



Já os dados **Qualitativos** não são (necessariamente) numéricos. Dados **Nominais** são aqueles em que não existe uma ordem ou hierarquia. Podemos pensar em dados nominais como **Classes** (eg1. azul, amarelo, verde) (eg2. honda, toyota, bmw).

Tipos de dados

Considere o banco de dados abaixo, em que temos a informações do tipo de carro (dado nominal) que cada cliente dirige:

Idade	**Salário**	Carro
24	2000	Honda
20	1000	Toyota
60	18000	Toyota
28	8500	Chevrolet
33	6000	Honda

Tipos de dados

Considere o banco de dados abaixo, em que temos a informações do tipo de carro (dado nominal) que cada cliente dirige:

Idade	**Salário**	**Carro**
24	2000	Honda
20	1000	Toyota
60	18000	Toyota
28	8500	Chevrolet
33	6000	Honda

Para calcularmos a distancia entre os dois primeiro clientes, temos que: X1 = [24,2000,"Honda"] e X2 = [20,1000,"Toyota"]. Como a fórmula é aplicada aos dados nominais?

Não é possível da forma como está! Precisamos transformar os dados. Nesse caso podemos usar a técnica da **binarização**.

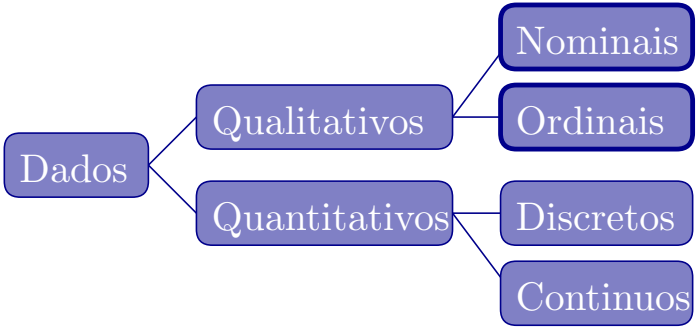
Tipos de dados

A **binarização** de um atributo consiste em contar o número de classes do atributo e criar uma nova coluna para cada elemento desse grupo. Em seguida, para cada registro marca-se 1 na coluna referente à classe do registro e zero em todas as outras.

Idade	Salário	Carro
24	2000	Honda
20	1000	Toyota
60	18000	Toyota
28	8500	Chevrolet
33	6000	Honda

Idade	Salário	Hon	Toy	Che
24	2000	1	0	0
20	1000	0	1	0
60	18000	0	1	0
28	8500	0	0	1
33	6000	1	0	0

Tipos de dados



Finalmente, nos dados **Ordinais** existe uma ordem gradativa nas classes. (eg. frio, morno, quente) (eg2. pouco, médio, muito).

Tipos de dados

Considere o banco de dados abaixo, em que temos agora a informação da avaliação que o cliente fez a respeito do serviço do shopping:

Idade	Salário	Carro	Serviço
24	2000	Honda	Bom
20	1000	Toyota	Ruim
60	18000	Toyota	Médio
28	8500	Chevrolet	Ruim
33	6000	Honda	Médio

Novamente não conseguimos usar a escala Ruim, Medio, Bom no cálculo das distâncias.

Para dados ordinais é mais simples, podemos simplesmente atribuir valores inteiros crescentes de acordo com a evolução da escala, por exemplo:

- Ruim:1
- Médio:2
- Bom:3

Tipos de dados

Com as transformações realizadas, podemos usar o banco de dados no algoritmo *k-means*.

Idade	**Salário**	**Carro**	**Serviço**
24	2000	Honda	Bom
20	1000	Toyota	Ruim
60	18000	Toyota	Médio
45	15000	Chevrolet	Ruim
33	6000	Honda	Médio

Idade	**Salário**	**Hon**	**Toy**	**Che**	**Av.**
24	2000	1	0	0	3
20	1000	0	1	0	1
60	18000	0	1	0	2
45	15000	0	0	1	1
33	6000	1	0	0	2

O problema das escalas

O problema das escalas

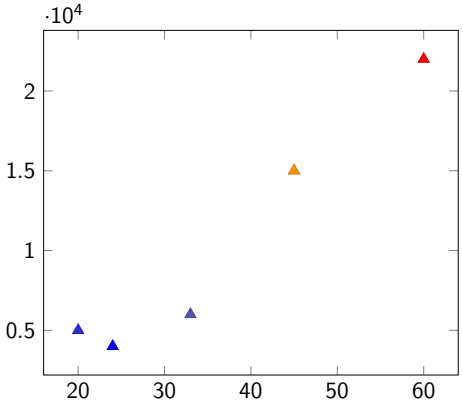
Um outro problema que pode ocorrer no k-mean se relaciona às escalas das variáveis

Considere o banco de dados abaixo:

Idade	**Salário**
24	4000
20	5000
60	22000
45	15000
33	6000

O problema das escalas

Plotando os dados em um *scatter*-plot temos o gráfico abaixo.



O problema das escalas

Podemos calcular a distância entre dois pontos da seguinte forma.

Idade	Salário
24	4000
20	5000
60	22000
45	15000
33	6000

$$d = \sqrt{(22000 - 15000)^2 + (60 - 45)^2} = 7000.016$$

O problema das escalas

Considere agora uma grande alteração nas idades. **De 45 para 120 anos!**

Idade	**Salário**
24	4000
20	5000
60	22000
120	15000
33	6000

$d = \sqrt{(22000 - 15000)^2 + (60 - 120)^2} = 7000.257$.

Comparando os resultados temos uma diferença nas distancias de 0.24

O problema das escalas

Realizando agora uma pequena alteração no salário (de 22000 para 21000)

Idade	**Salário**
24	4000
20	5000
60	21000
45	15000
33	6000

$d = \sqrt{(21000 - 15000)^2 + (60 - 45)^2} = 6000.0187$.

Gerando uma diferença de mais 999.99 unidades!

O problema das escalas

Note que isso gera um problema de **dominância** no algoritmo: pontos muito diferentes em relação a algum atributo podem não gerar efeito nenhum na classificação, na medida em que pequenas alterações em outros podem gerar grandes variações. Uma forma de corrigir esse problema é pela padronização dos dados.

A idéia da padronização é deixar todos os atributos em uma mesma escala. Existem inúmeras formas de padronização (cada uma adequada a uma situação), de forma que devemos avaliar os nossos dados e o que queremos atingir com a padronização.

O problema das escalas

ATENÇÃO À NOMENCLATURA: Em estatística o termo **normalização** é usado quando queremos transformar os dados, deixando-os com média 0 e desvio padrão 1. Para isso usamos a seguinte fórmula:

$$x' = (x - \bar{x})/s_x \tag{1}$$

Em que:

$$\begin{cases} x' : \text{novo valor} \\ x : \text{valor atual} \\ s_x : \text{ desvio padrão da amostra} \end{cases}$$

Porém, em algumas referências de mineração de dados os termos foram usados como sinônimos (existe uma grande quantidade de explicações na internet, algumas contraditórias). Podemos pensar da seguinte forma então: **padronizar se refere a transformar uma variável em outra, de forma que a normalização é um tipo de padronização**.

O problema das escalas

Voltando ao nosso problema. Que tipo de padronização podemos usar para deixar todos os atributos na mesma escala?

Idade	**Salário**
24	4000
20	5000
60	22000
45	15000
33	6000

Se dividirmos cada elemento de cada coluna pelo máximo valor da mesma coluna, temos o seguinte.

O problema das escalas

Um outro problema que pode ocorrer no k-mean se relaciona às escalas das variáveis

Idade	**Salário**
24/60	4000/22000
20/60	5000/22000
60/60	22000/22000
45/60	15000/22000
33/60	6000/22000

Idade	**Salário**
0.40	0.18
0.33	0.22
1.00	1.00
0.75	0.68
0.55	0.27

O problema das escalas

Realizando essas transformações, os exemplos que calculamos acima geram as seguintes variações:

1. Alterando a idade (45 para 120): 0.188
2. Alterando o salário (22000 para 21000): 0.024

Ou seja, a alteração na idade afetou mais a distância do que a alteração no salário, o que era o esperado de se ocorrer.

O problema das escalas

Como foi dito, existem diversas formas de padronização. A divisão pelo maior valor não é adequada quando existem valores negativos, nesse caso podemos dividir todos os elementos de uma coluna pela amplitude dos dados (maior valor - menor valor).

maximize z subject to

$$z \leq x - y + M\epsilon \tag{2}$$
$$z \leq y - x + M(1 - \epsilon) \tag{3}$$

where $\epsilon \in \{0, 1\}$ and M is a very large number: large enough that when $\epsilon = 1$, the first constraint doesn't restrict z, and when $\epsilon = 0$, the second constraint doesn't restrict z. In this case, we have specific bounds on x, y and so taking M to be something like $2(b - a + d - c)$ could work

Visualização e redução de dimensionalidade

Visualização e redução de dimensionalidade

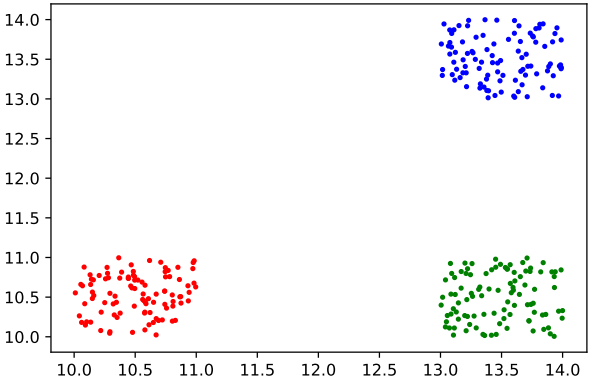
Quando terminamos de realizar um agrupamento, gostamos de **visualizar espacialmente** como ficaram os grupos, para nos certificarmos que está tudo fazendo sentido.

Por exemplo, considere um conjunto de dados com o seguinte formato:

Idade	**Salário**
24	4000
20	5000
...	...
60	21000
45	15000
33	6000

Visualização e redução de dimensionalidade

Podemos visualizar os grupos diretamente em um plano cartesiano (usando um *scatter plot* e alterando as cores dos elementos de cada grupo.)



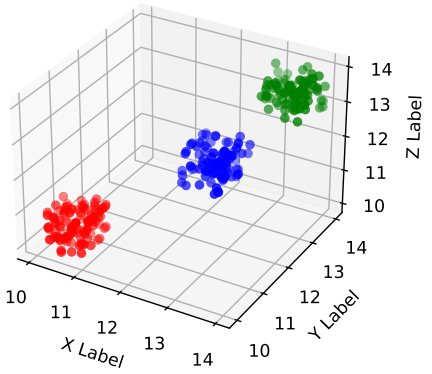
Visualização e redução de dimensionalidade

Esse método pode ser usado para conjuntos de dados com 3 dimensões, considere o banco de dados a seguir:

Idade	Salário	Nota
24	4000	3
20	5000	2
...	...	
60	21000	2
45	15000	6
33	6000	7

Visualização e redução de dimensionalidade

Podemos visualizar os grupos diretamente em um plano cartesiano (usando um *scatter plot* e alterando as cores dos elementos de cada grupo, porém agora com 3 dimensões).



Visualização e redução de dimensionalidade

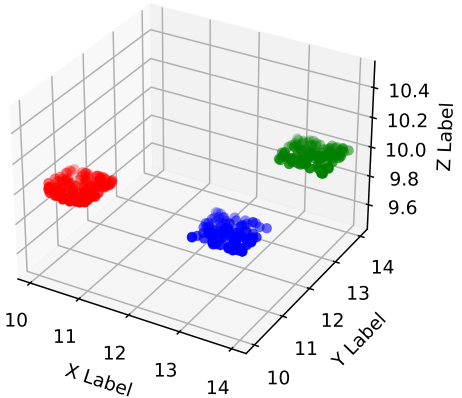
Mas e quando temos dados com mais de 3 dimensões? Como podemos visualizar os grupos formados?

Considere o seguinte conjunto de dados, para entendermos a premissa do método:

Idade	**Salário**	**Nota**
24	4000	3
20	5000	3
...	...	
60	21000	3
45	15000	3
33	6000	3

Plotando os pontos teríamos algo da seguinte forma:

Visualização e redução de dimensionalidade



Visualização e redução de dimensionalidade

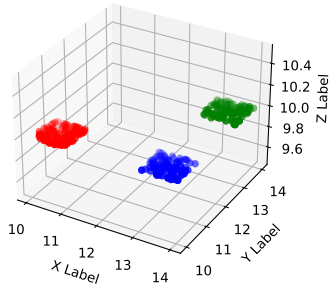
Note, porém, que todos os valores do eixo z são constantes, ou seja, o atributo z não ajuda na explicabilidade dos grupos, e portanto pode ser removido.

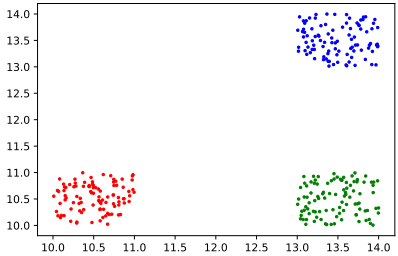
Considere o seguinte conjunto de dados, para entendermos a premissa do método:

Idade	**Salário**	**Nota**
24	4000	3
20	5000	3
...	...	
60	21000	3
45	15000	3
33	6000	3

Visualização e redução de dimensionalidade

Um outro problema que pode ocorrer no k-mean se relaciona às escalas das variáveis





Visualização e redução de dimensionalidade

De forma simular funcionam os métodos para redução da dimensionalidade. Um deles é o **PCA** (*Principal Components Analysis*)

O PCA identifica os atributos que mais contribuem para a variabilidade dos dados e cria uma combinação linear com k novos elementos, que são usados para representar os n originais. Esses novos elementos **não tem nenhuma relação com os anteriores** (representativa), somente no sentido de explicação da variabilidade.

Visualização e redução de dimensionalidade

Por exemplo, suponha que temos um banco com 3 atributos e queremos realizar a redução para 2. Uma possível saída do **PCA** é mostrada abaixo.

Idade	**Salário**	**Nota**
24	4000	3
20	5000	3
...	...	
60	21000	3
45	15000	3
33	6000	3

α	β
0.40	0.18
0.33	0.22
1.00	1.00
0.75	0.68
0.55	0.27

Note que a escala de α e β não tem mais relação nenhuma com Idade, Salário ou Nota. Com esse novo conjunto podemos visualizar por meio de um *scatterplo*t em 2 dimensões.