INTELIGÊNCIA ARTIFICIAL APLICADA - MACHINE LEARNING

AULA 5

Prof. Antonio Willian Sousa

CONVERSA INICIAL

Anteriormente, vimos diversos classificadores, com destaque para o fato de que todos eram supervisionados. Ou seja, necessitavam de um conjunto de dados em que as indicações das classes a qual cada instância pertencia estavam registradas no próprio conjunto de dados de treino ou de testes. Entretanto, nem sempre essa informação está disponível ou sua obtenção pode ser demorada e custosa.

Quando se tem um problema de aprendizado de máquina e precisamos definir as classes das instâncias, essa avaliação é feita por pessoas, observando cada elemento do conjunto de dados de treino e de testes, um a uma, e marcando as categorias ou classes de cada instância. Esse processo, além de demorado, está sujeito a erros e falhas, sendo por isso necessário que se obtenha a marcação de mais de um avaliador. O que pode conduzir a divergências nas anotações das classes, sendo necessário utilizar métodos que evitem esses tipos de problemas.

Quando a anotação das categorias das instâncias não é possível de imediato, uma alternativa é buscarmos extrair conhecimento dos dados, sem considerar o conhecimento prévio sobre elas. Para situações desse tipo utilizamos métodos de aprendizagem não supervisionada. Esses métodos são separados em dois grupos: agrupamento e associação. Problemas de agrupamento são aqueles em que queremos descobrir a formação de grupos nos dados, por exemplo, agrupar clientes com base nos seus padrões de compra. Já problemas de associação são aqueles nos quais queremos descobrir regras que possam descrever os dados, como nos casos em que um cliente compra um produto e tende a comprar outro.

Nesta aula, falaremos sobre os métodos de aprendizagem não supervisionada, em especial sobre os algoritmos k-means e agrupamento hierárquico, além de fornecer uma base teórica sobre o conceito de distância que é fundamental para técnicas desse tipo. Ao final, falaremos ainda sobre outros métodos de aprendizagem não supervisionada que podem ser utilizados.

TEMA 1 – ALGORITMO K-MEANS

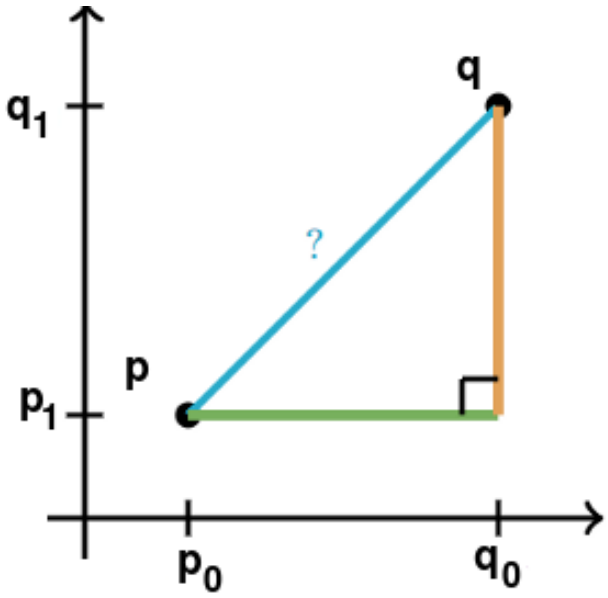
Nesta seção, apresentaremos o algoritmo k-means e a forma como ele é utilizado para problemas de aprendizagem de máquina não supervisionados. Esse algoritmo funciona executando uma busca por agrupamentos dentro do conjunto de dados fornecidos e pode ser aplicado em diversos problemas como segmentação de clientes, detecção de anomalias nos dados, segmentação de imagens, entre outros.

1.1 Agrupamento no k-means

O funcionamento do k-means se dá pela partição do conjunto de dados em k-agrupamentos de maneira que as instâncias de um mesmo grupo apresentam similaridades entre si e diferenças com as instâncias dos demais grupos. A similaridade entre as instâncias é determinada por meio da distância entre elas, de forma semelhante ao processo executado no algoritmo kNN que aprendemos anteriormente.

Há diversas maneiras de medirmos a distância entre instâncias de dados. A distância euclidiana é uma das medidas mais comuns em problemas de aprendizagem de máquina. O Gráfico 1 mostra como é efetuado o cálculo da distância euclidiana para dois pontos (instâncias) em um espaço de duas dimensões.

Gráfico 1 – Distância euclidiana entre dois pontos no espaço de duas dimensões



Fonte: Sousa, 2020.

No Gráfico 1, p = (p_0, p_1), q = (q_0, q_1) e a distância entre os dois pontos é dada por $dist(p,q) = \sqrt{(q_0 - p_0)^2 + (q_1 - p_1)^2}$.

Como já vimos anteriormente, os vetores de características costumam ter muito mais que dois valores, como nessa situação. Assim, para o cálculo entre instâncias (pontos) em um espaço de atributos maior, o valor da distância será calculado, expandindo o cálculo em duas dimensões, por meio da seguinte fórmula:

p = (p_0, p_1, …, p_{n-1})

q = (q_0, q_1, …, q_{n-1})

$$dist(p,q) = \sqrt{\sum_{i=0}^{n-1}(q_i - p_i)^2}$$

Assim, se o nosso vetor de características possuir cinco dimensões, o cálculo será feito da seguinte forma:

$$dist(p,q) = \sqrt{(q_0 - p_0)^2 + (q_1 - p_1)^2 + (q_2 - p_2)^2 + (q_3 - p_3)^2 + (q_4 - y_p)^2}$$

Assim, todos os valores dos atributos que definem as instâncias tomam parte no cálculo da distância entre elas. Esse mesmo raciocínio pode ser utilizado para qualquer tamanho de vetor de características.

Para simplificar o entendimento do método de agrupamento do k-means, suponhamos que seja necessário o conjunto de dados de 1.000 clientes conforme a Tabela 1 a seguir.

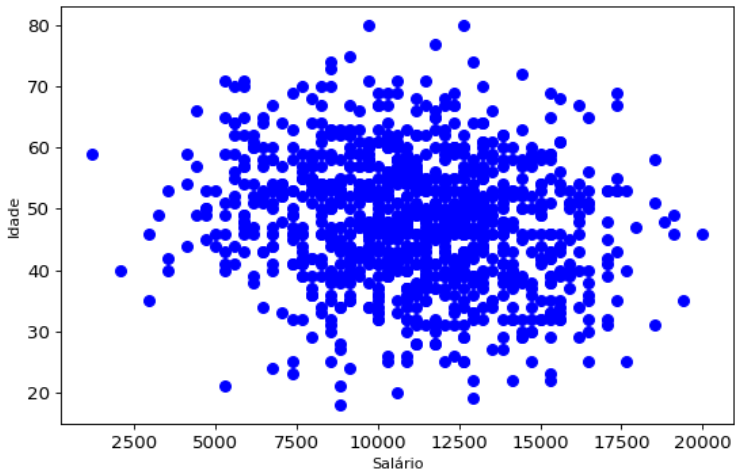
Tabela 1 – Conjunto de dados de 1.000 clientes

	idade	gosta de esportes	gosta de ler	gosta de filmes	codigo municipio	sexo	nacionalidade	salario
0	25.0	1.0	1.0	1.0	7.0	1.0	6.0	4882.47
1	50.0	1.0	1.0	1.0	14.0	1.0	3.0	13604.12
2	36.0	1.0	1.0	1.0	15.0	1.0	5.0	13022.68
3	37.0	1.0	1.0	1.0	7.0	1.0	6.0	6045.36
4	44.0	1.0	1.0	1.0	9.0	1.0	6.0	2556.70
...	...	...	...	...	...	...	...	...
995	44.0	1.0	1.0	1.0	12.0	1.0	1.0	20000.00
996	27.0	1.0	1.0	1.0	2.0	1.0	7.0	6626.80
997	26.0	1.0	1.0	1.0	9.0	1.0	5.0	6626.80
998	53.0	1.0	1.0	1.0	16.0	1.0	4.0	15542.27
999	35.0	1.0	1.0	1.0	7.0	1.0	6.0	7208.25

Fonte: Sousa, 2020.

Esse conjunto de dados possui oito atributos que serão utilizados para definir cada uma das instâncias dos clientes. Por conta do número de atributos, não conseguimos visualizar a distribuição das instâncias por meio de um gráfico com todos os atributos, mas podemos avaliar em relação a pelo menos um par de atributos. No Gráfico 2, por exemplo, podemos ver a distribuição do conjunto de dados de clientes em relação aos atributos *salário* e *idade*.

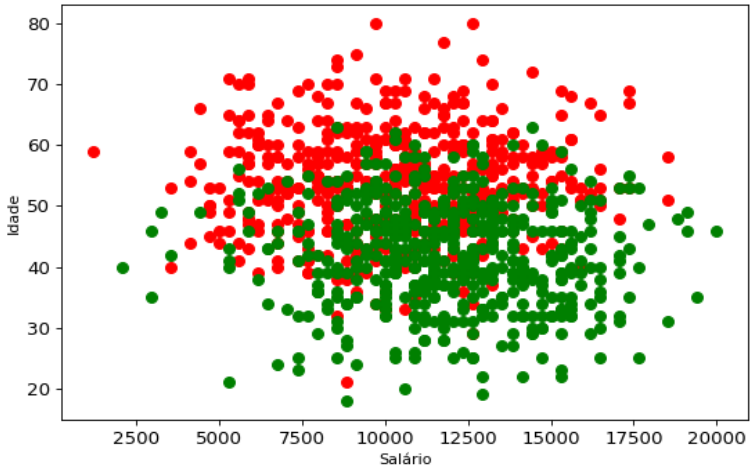
Gráfico 2 – Clientes por salário e idade



Fonte: Sousa, 2020.

Com base no gráfico anterior, podemos perceber que é difícil definir uma separação em grupos do conjunto completo de dados. Contudo, o k-means consegue agrupar as instâncias de clientes em dois grupos distintos, conforme mostrado no Gráfico 3.

Gráfico 3 – Clientes separados em dois grupos distintos



Fonte: Sousa, 2020.

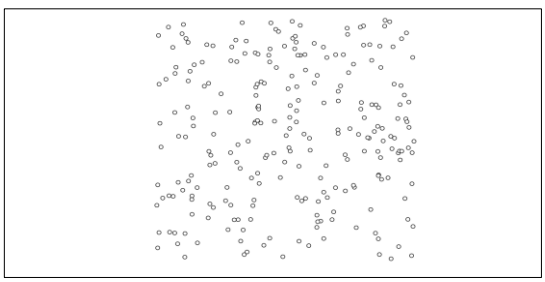
1.2 Algoritmo do k-means

O algoritmo k-means funciona de forma iterativa, recebendo o valor dos **k** agrupamentos que devem ser buscados e executando os seguintes passos.

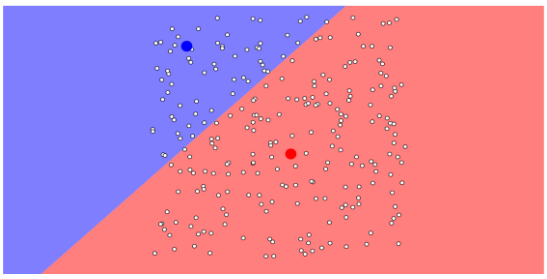
1. Seleciona os centros de cada agrupamentos (centroides).
2. Calcula a distância entre os centroides e todas as instâncias.
3. Coloca cada instância como parte do agrupamento mais próximo.
4. Busca uma nova posição para os centroides, calculando a média de todas as instâncias do agrupamento.
5. Repete os passos 2, 3 e 4 até que os centroides não mudem mais de lugar.

Os elementos da Figura 1 a seguir ilustram o funcionamento desse processo.

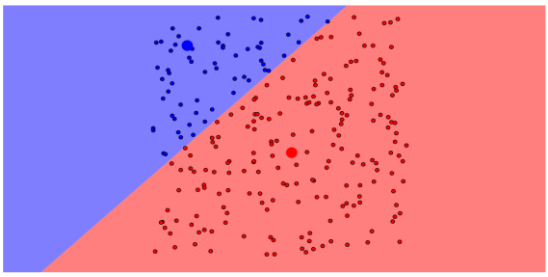
Figura 1 – Funcionamento



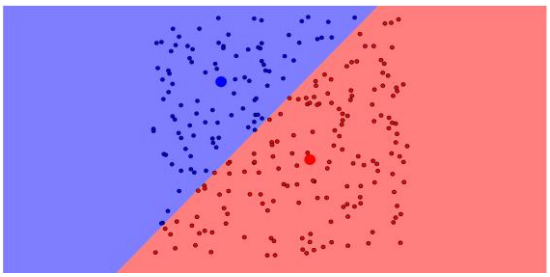
1 – Conjunto de dados sem agrupamentos



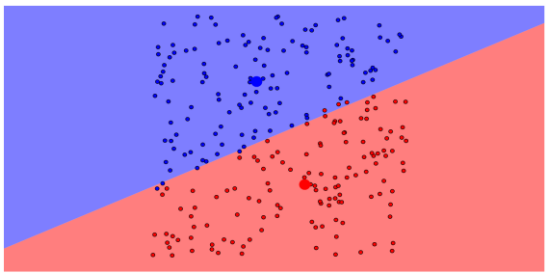
2 – Definidos centroides para k=2



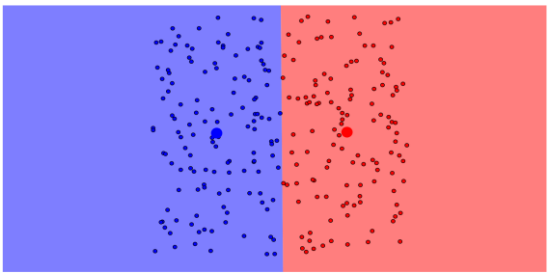
3 – Instâncias atribuídas ao centroide mais próximo



4 – Busca nova posição dos centroides



5 – Executa passos 2,3 e 4 novamente



6 – Até que haja convergência dos centroides

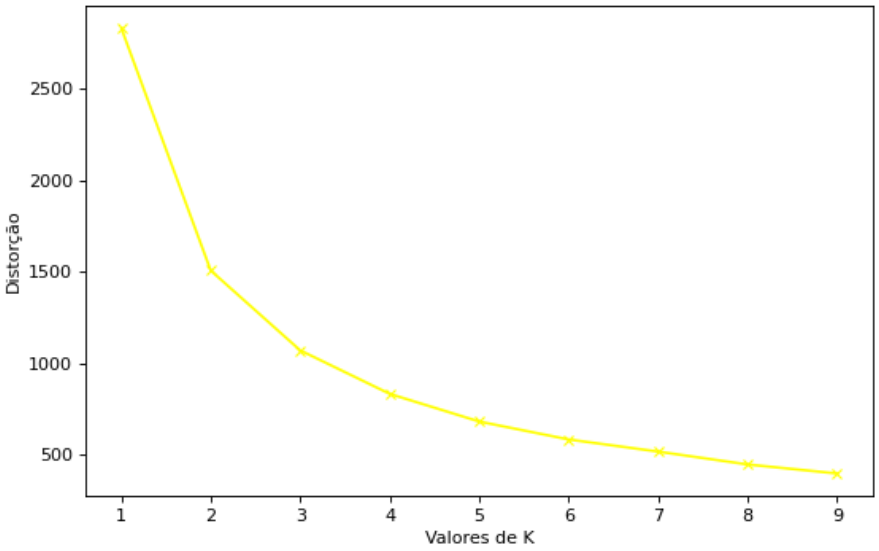
Fonte: Sousa, 2020.

O processo de convergência dos centroides pode ser bastante demorado dependendo da quantidade de instâncias. Também deve ser considerado que o posicionamento inicial dos centroides pode contribuir para uma redução na quantidade de cálculos a serem efetuados. Além disso, também devemos considerar o valor de **k** a ser escolhido, pois todos os cálculos dependerão desse valor que determina quanto centroides teremos.

Uma estratégia comumente utilizada para a posição inicial dos centroides é a escolha aleatória, contudo, essa estratégia pode acabar colocando centroides muito próximos e isso fará com que a convergência seja muito lenta. Uma estratégia que, em repetidas execuções, tende a apresentar melhores resultados é posicionar o primeiro centroide aleatoriamente e os seguintes o mais distante possível uns dos outros, evitando que os centroides sejam posicionados muito próximos e facilitando o processo de convergência.

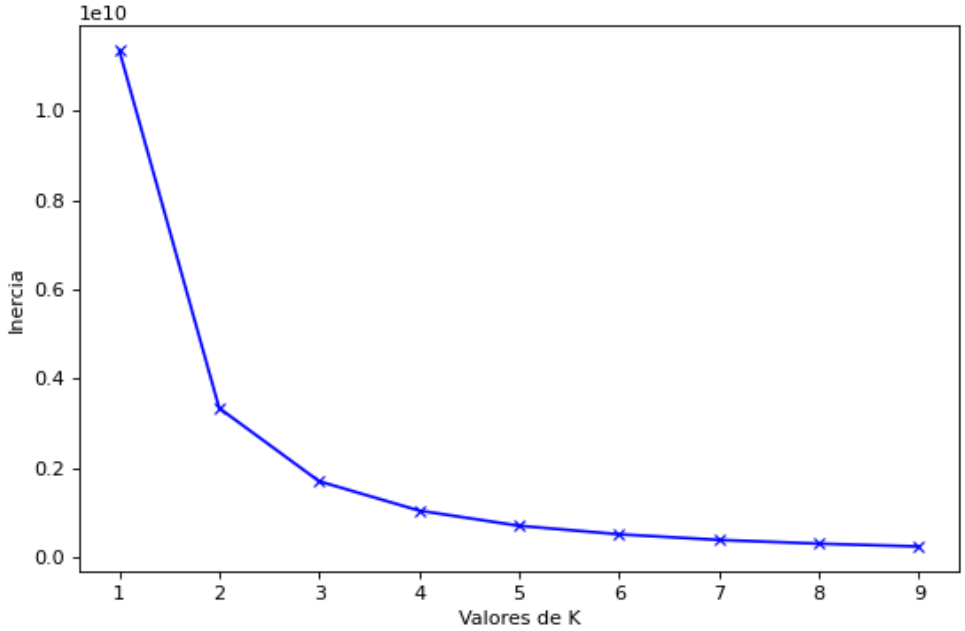
Uma maneira de determinar uma melhor quantidade de agrupamentos é utilizando o método do cotovelo, que consiste em avaliar o ponto em que a distorção e a inércia decrescem linearmente, no qual a distorção é a distância média quadrática dos centroides dos agrupamentos e a inércia é a soma dos quadrados das distâncias das instâncias para os centroides do seu *cluster*. Nos Gráficos 4 e 5 estão disponibilizados os valores de inércia e de distorção do nosso conjunto de dados, cujo valor de k=5 seria uma boa escolha.

Gráfico 4 – Valores de distorção



Fonte: Sousa, 2020.

Gráfico 5 – Valores de inércia



Fonte: Sousa, 2020.

O processo de escolha do número de centroides, bem como o cálculo das posições intermediárias e definitivas destes podem ser feitos utilizando-se diferentes métricas de distância, sendo esta também uma escolha determinante para a *performance* do k-means. Nem sempre é garantido que o ponto de inflexão (cotovelo) seja encontrado, pois os valores de dispersão e inércia dependerão dos dados utilizados. Há também situações em que o melhor número de agrupamentos será um valor que está além do ponto de inflexão, pois

deve ser considerado um valor que traga equilíbrio entre os valores de distorção e inércia.

TEMA 2 – MEDIDAS DE DISTÂNCIA

Em diferentes algoritmos de aprendizagem de máquina, tanto aqueles utilizados para aprendizagem supervisionada como não supervisionada, é fundamental o uso de uma função de distância entre as instâncias do conjunto de dados. Em algoritmos como o kNN e o k-means, o cálculo da distância pode ser determinante na qualidade da predição obtida, bem como na velocidade de treinamento e execução dos modelos.

Na seção anterior, vimos como é calculada a distância euclidiana entre duas instâncias. Nesta seção, apresentaremos outras medidas de distância que comumente são utilizadas em aprendizagem de máquina e as situações em que cada uma pode ser aplicada.

2.1 Distância Manhattan

A distância Manhattan deve ser utilizada em casos nos quais se necessita calcular a distância entre dois pontos definida por um caminho que deve obedecer ao formato de uma grade. Essa distância também é referenciada como medida de distância do táxi ou distância de quarteirões da cidade. O Gráfico 6 mostra a distância Manhattan entre dois pontos em um espaço de duas dimensões.

A distância Manhattan é calculada com base na diferença absoluta das coordenadas dos pontos. Assim, diferentemente da distância euclidiana, diversos caminhos podem gerar os mesmos valores da distância Manhattan. O valor da distância pode ser obtido para qualquer quantidade de atributos, por intermédio da fórmula:

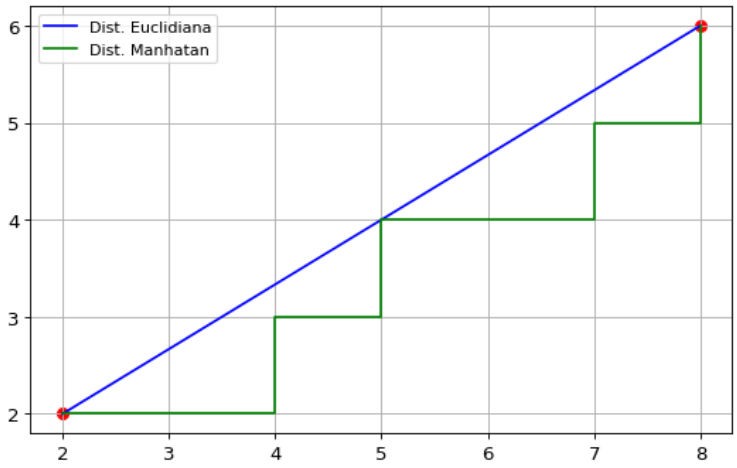
$$dist(p,q) = \sum_{i=0}^{n-1} |q_i - p_i|$$

Assim, se o vetor de características apresentar cinco dimensões, o cálculo será feito da seguinte forma:

$$dist(p,q) = |q_0 - p_0| + |q_1 - p_1| + |q_2 - p_2| + |q_3 - p_3| + |q_4 - y_p|$$

A distância Manhattan é usualmente utilizada em substituição à distância euclidiana em situações nas quais a dimensionalidade dos dados seja elevada.

Gráfico 6 – Distância Manhattan e euclidiana entre dois pontos



Fonte: Sousa, 2020.

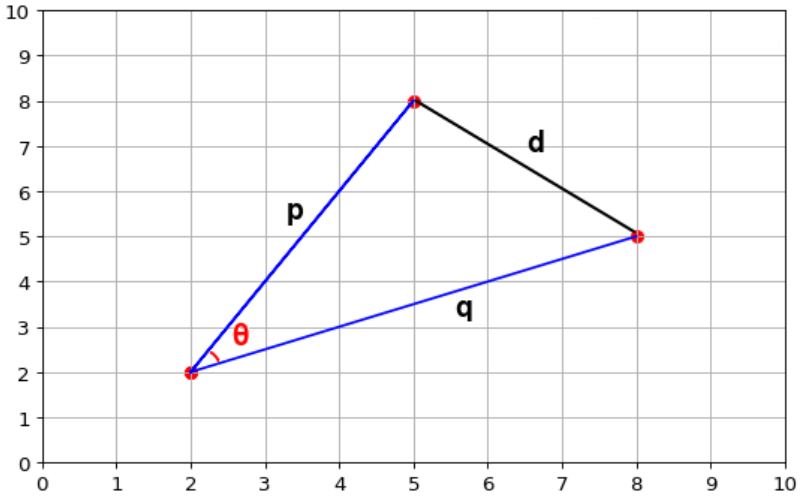
2.2 Distância Cosseno

A distância Cosseno deve ser utilizada em situações nas quais a magnitude dos vetores de características não importe, mas a sua orientação, sim. Essa distância é obtida por meio da similaridade Cosseno entre os dois vetores de características de duas instâncias, que é determinada pela seguinte fórmula.

$$similaridade = cos\theta = \frac{p.q}{||p||.||q||} = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} p_i}\sqrt{\sum_{i=1}^{n} q_i}}$$

O Gráfico 7 exibe a distância entre os dois pontos e o ângulo determinado pelos próprios vetores.

Gráfico 7 – Cosseno entre dois vetores **p** e **q**



Fonte: Sousa, 2020.

Pelo Gráfico 7, podemos perceber que, se considerarmos um intervalo de 0 a 90° do ângulo entre os dois vetores, quanto maior o ângulo, menor será o valor do cosseno e maior será a distância entre os pontos e vice-versa. Assim, quanto mais próximos os vetores estiverem de um alinhamento nos seus sentidos, mais similares eles serão considerados e maior será a sua distância cosseno. Dessa definição, podemos perceber que a distância Cosseno e a similaridade Cosseno são inversamente proporcionais.

Dessa maneira, a distância cosseno será determinada pela fórmula $DistânciaCosseno(p,q) = 1 - SimilaridadeCosseno(p,q)$.

Assim, além das situações em que o ângulo entre os vetores for menor que 45°, teremos outros seis casos específicos conforme a Tabela 2 a seguir.

Tabela 2 – Outros casos específicos

Ângulo θ	Valor Cos θ	Distância Cosseno
45°	0,525	0,475
90°	0	1
0°	1	0
180°	–1	2
270°	0	1
360°	1	0

Fonte: Sousa, 2020.

A partir dos valores do ângulo entre os vetores das instâncias, podemos notar que, quando eles estão em sentidos opostos (180°), a similaridade será mínima e a distância máxima. O contrário ocorre quando os vetores estão alinhados (0° e 360°) com similaridade máxima.

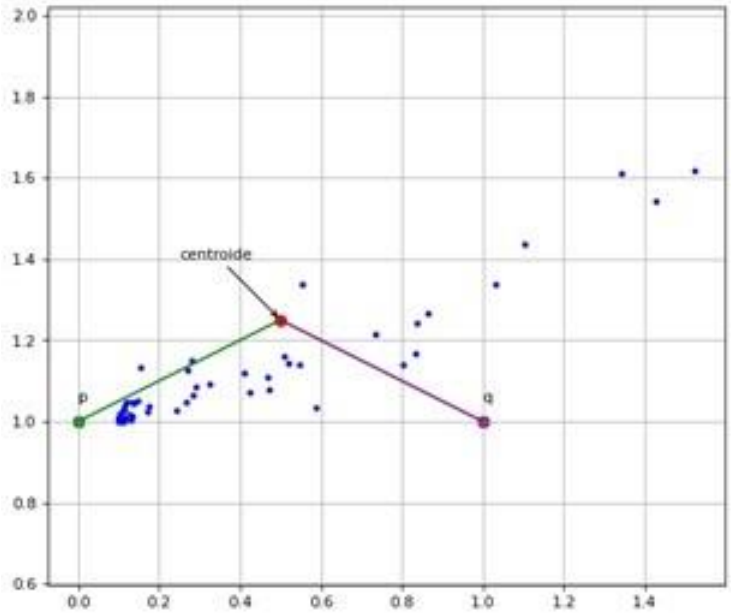
A distância Cosseno é comumente utilizada em sistemas de recomendação, por meio da comparação dos hábitos ou das preferências de clientes, bem como na comparação de documentos, em que estes podem ter vetores de características de diferentes magnitudes, mas ainda assim é possível determinar o quão similares as suas instâncias são.

2.3 Distância de Mahalanobis

A distância de Mahalanobis é uma medida que serve para avaliar tanto a distância de uma instância em relação ao conjunto de dados quanto a dissimilaridade entre dois vetores de instâncias com a mesma distribuição. Uma de suas vantagens em relação a outras medidas, como a distância euclidiana, reside no fato de ser invariante à escala. Ou seja, diferentemente da distância euclidiana, não é necessário executar a normalização dos valores das características.

O uso da distância de Mahalanobis pode ser vantajoso na determinação de agrupamentos, por meio do cálculo da distância das instâncias aos centroides, como mostrado no Gráfico 8.

Gráfico 8 – Distância de dois pontos para um centroide



Fonte: Sousa, 2020.

Podemos perceber que as duas instâncias indicadas no Gráfico 8 apresentam a mesma distância euclidiana do centroide do agrupamento, mas o ponto em verde deve apresentar maior semelhança com o agrupamento que o segundo ponto. Como no cálculo da distância euclidiana a correlação entre características não é considerada, essas duas serão consideradas similares.

A distância de Mahalanobis, para uma única instância, pode ser calculada a partir da seguinte fórmula.

$$D(x) = \sqrt{(x-\mu)^T S^{-1}(x-\mu)}$$

Em que:

- x é a instância, que pode ser representada pelo seu vetor de características $x = (x_1, x_2, \ldots, x_n)$;
- S é a matriz de covariância;
- μ é um conjunto de observações sobre os dados com média $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$.

Como mencionamos anteriormente, a distância de Mahalanobis é muito utilizada em problemas de agrupamento, bem como em problemas de classificação e análise de padrões. Como a distância avalia uma instância em relação à distribuição das outras instâncias do conjunto de dados, ela pode ser uma forma eficiente de buscar *outliers* e encontrar desvios nas predições.

2.4 Distância de Hamming

Há diversas métricas de distância que podem ser utilizadas em aprendizagem de máquina e sua escolha está relacionada ao problema em questão. Até agora vimos métricas que comparam pontos no espaço de características. Para situações em que precisamos comparar *strings*, há uma métrica chamada de *distância de Hamming* para esse tipo de medição, a qual considera o número de *bit* posicionais nos quais as *strings* diferem.

Para calcular a distância de Hamming entre duas *strings* **a** e **b**, executa-se uma operação XOR entre elas (a ⊕ b) e contabiliza-se o total de 1's na *string* resultante. Assim, a distância de Hamming pode ser utilizada tanto para avaliar similaridade de *strings* como de vetores de atributos *one-hot enconder*.

O código a seguir calcula a diferença entre duas *strings*.

```
from scipy.spatial import distance
s1 = "texto1"
s2 = "texto2"
distancia = distance.hamming(list(s1), list(s2))*len(s1)
```

No caso de um conjunto de características codificadas como *one-hot encoder*, conforme mostrado na Tabela 3, o cálculo da distância entre duas instâncias desse *dataset* pode ser feito da mesma forma mostrada anteriormente.

```
from scipy.spatial import distance
    distance.hamming([0,0,1,0,0,0,1], [0,0,0,0,0,0,1])
```

Tabela 3 – Data com características *one-hot encoder*

	tipo_transmissao_1	tipo_transmissao_2	tipo_transmissao_3	tipo_transmissao_4	tracao_1	tracao_2	tracao_3
1	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1
3	0	0	1	0	0	0	1
4	0	0	1	0	0	0	1
5	0	0	1	0	0	0	1
...	...	...	...	...	...	...	...
11746	0	0	0	0	0	0	1
11747	0	0	0	0	0	0	0
11748	0	0	0	0	0	0	1
11749	0	0	0	0	0	0	0
11750	0	0	0	0	0	0	1

Fonte: Sousa, 2020.

TEMA 3 – USANDO K-MEANS

Agora que já sabemos como o algoritmo do k-means funciona e estabelecemos o conceito de distância entre instâncias no espaço de atributos, bem como apresentamos diferentes métricas de distância que podem ser utilizadas, veremos como utilizar o k-means em problemas reais.

3.1 Agrupando os dados

Ao utilizar o k-means para executar agrupamento dos dados, necessitamos ter em mente quanto grupos estamos buscando. Pelo menos a princípio, o que os grupos encontrados pelo k-means significam não está muito claro, até que exploremos as instâncias posicionadas nos agrupamentos. Para ilustrar o funcionamento do k-means, consideremos o *dataset* de clientes mostrado a seguir.

Tabela 4 – *Dataset* de clientes

	idade	peso	altura	ultima compra	codigo municipio	sexo	total comprado	salario
0	52.0	37.0	124.0	194.0	6.0	0.0	455.0	10391.11
1	57.0	57.0	117.0	97.0	7.0	1.0	1516.0	13315.56
2	34.0	89.0	188.0	549.0	12.0	1.0	4849.0	2453.33
3	55.0	48.0	120.0	194.0	4.0	1.0	304.0	14568.89
4	52.0	52.0	102.0	162.0	6.0	1.0	531.0	16240.00
...	...	...	...	...	...	...	...	...
995	28.0	66.0	172.0	291.0	13.0	1.0	3637.0	1617.78
996	34.0	57.0	171.0	468.0	12.0	1.0	4167.0	3915.56
997	32.0	25.0	110.0	678.0	17.0	1.0	6061.0	5168.89
998	56.0	55.0	124.0	49.0	4.0	1.0	379.0	16240.00
999	23.0	31.0	122.0	404.0	12.0	1.0	7652.0	8928.89

Fonte: Sousa, 2020.

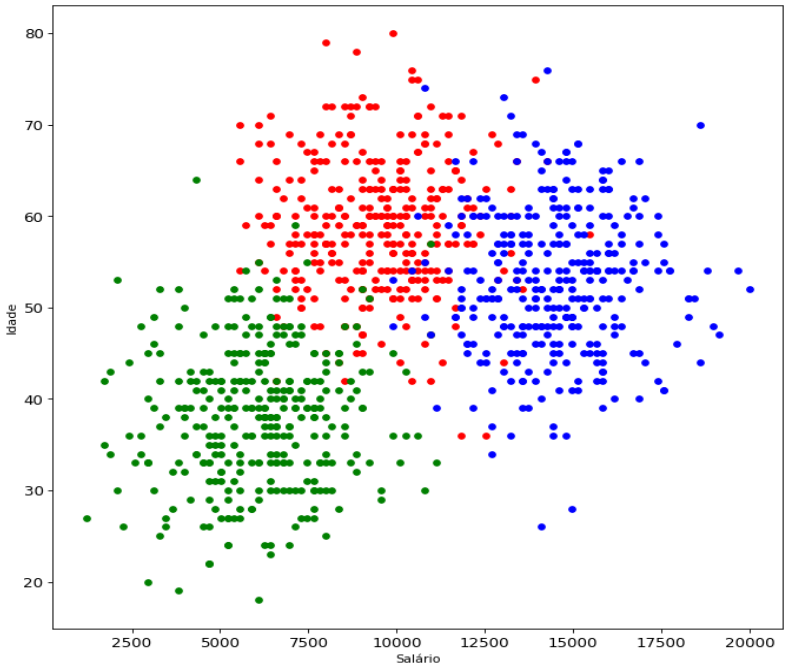
O código a seguir executa ao k-means sobre esse conjunto de dados, buscando determinar a existência de três agrupamentos nos dados.

```
from sklearn.cluster import KMeans
import pandas as pd
import numpy as np
# Lê o conjunto de dados
X = pd.read_csv("dados_clientes_compras.csv").to_numpy()
```

```
# Executa k-means e realiza predição
modelo_kmeans = Kmeans(n_clusters=3).fit(X)
modelo_kmeans.fit(X)
y_pred = modelo_kmeans.predict(X)
```

Após a predição dos agrupamentos, podemos visualizar os grupos por meio da relação entre dois atributos como salário e idade dos clientes. No Gráfico 9 foi exibido o posicionamento das instâncias em cada um dos grupos detectados.

Gráfico 9 – Agrupamentos detectados pelo k-means



Fonte: Sousa, 2020.

Pela disposição das instâncias nessa representação, podemos perceber que algumas instâncias de um grupo aparecem bem afastadas do centro do seu grupo e, aparentemente, se sobrepondo a outras de um grupo diferente. Isso acontece pelo fato de a nossa visualização utilizar apenas dois atributos, enquanto o vetor de características das instâncias é definido por um total de oito atributos. Infelizmente, não é possível visualizarmos de forma gráfica um espaço de oito dimensões, assim, o que podemos fazer é utilizar um método de redução que consiga representar em um espaço de duas dimensões os vetores de oito dimensões. Para isso, podem ser utilizados métodos de projeção. Entre eles, o mais comumente utilizado é um método chamado de *Principal Component Analysis* (PCA).

3.2 Principal Component Analysis

O PCA é uma técnica de redução de dimensionalidade dos dados, tornando conjuntos de dados de alta dimensionalidade mais fáceis de serem visualizados. Como o método reduz a quantidade de atributos a serem considerados, ele também reduz o custo computacional, permitindo que os modelos aprendam de forma mais rápida.

A execução do PCA exige normalização dos dados para evitar que características com valores absolutos e altos dominem o processo. O código a seguir o executa normalizando os dados e reduzindo os vetores de características de oito dimensões para novos vetores com apenas duas dimensões.

```
 from sklearn.decomposition import PCA
import pandas as pd
import numpy as np

# Lê o conjunto de dados
df_clientes = pd.read_csv("dados_clientes_compras.csv").to_numpy()

 # Cria o modelo PCA e reduz a dimensão dos dados para duas
 pca = PCA(n_components=2)
 pca.fit(df_clientes)

 # Transfor the scaled data to the new PCA space
 clientes_2pca = pca.transform(df_clientes)

 # Converte para um dataframe
 df_clientes_2pca = pd.DataFrame(clientes_2pca, index=df_usuarios.index,
     columns=['PC1','PC2'])
 df_clientes_2pca['cluster'] = clusters
```

O resultado será conforme a Tabela 5 a seguir.

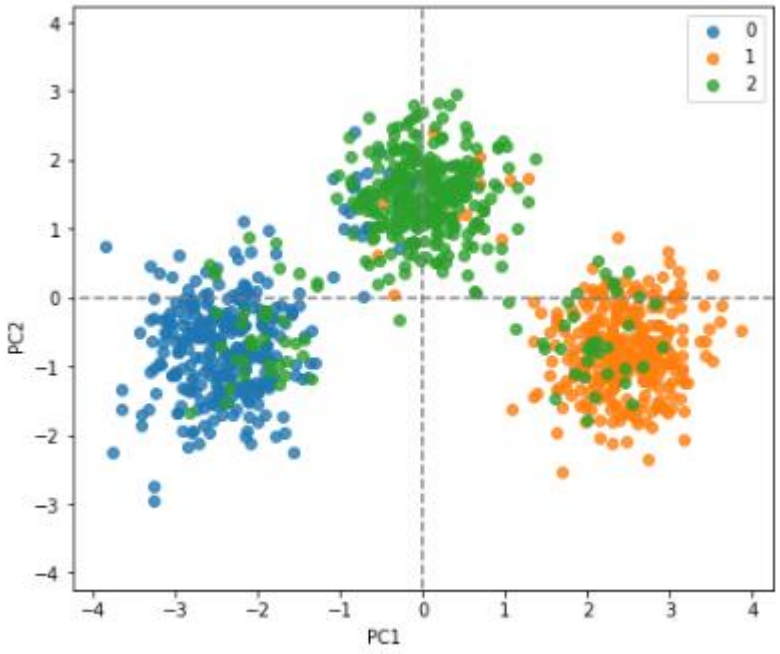
Tabela 5 – Resultado

	PC1	PC2	cluster
0	0.254961	1.594636	2
1	1.693504	-1.607076	1
2	-2.123146	-1.517009	0
3	0.536663	0.336852	2
4	-2.101821	-0.610420	2
...	...	...	...
995	-3.001214	-0.628468	0
996	-2.803128	-0.514234	0
997	-2.474521	-0.602548	0
998	2.413928	-0.148478	1
999	1.541450	-1.120015	1

Fonte: Sousa, 2020.

Agora, com apenas duas dimensões, a visualização dos agrupamentos fica facilitada. No Gráfico 10, podemos ver os agrupamentos de forma mais clara e considerando todos os elementos do vetor de características.

Gráfico 10 – Agrupamentos plotados para dois componentes principais



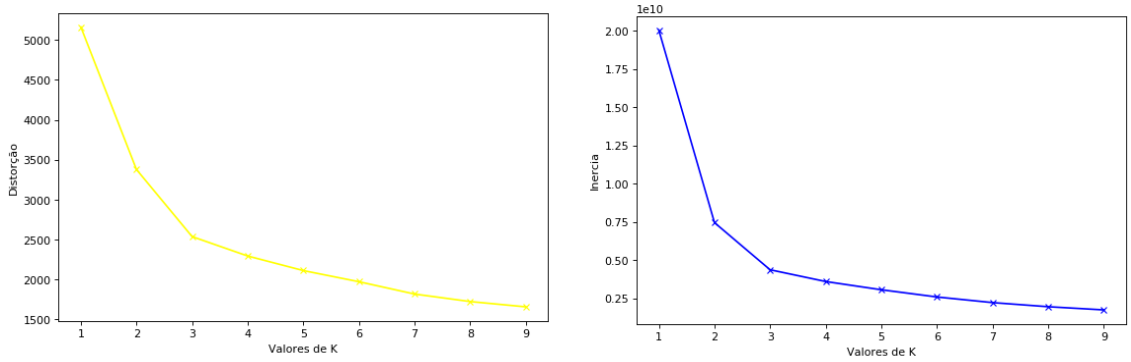
Fonte: Sousa, 2020.

3.3 Entendo os agrupamentos

Antes de explorarmos os agrupamentos, devemos verificar se o número de agrupamentos escolhido é adequado. Para nos certificarmos do número de

agrupamentos, podemos avaliar por meio do método do cotovelo, conforme mostrado no Gráficos 11.

Gráfico 11 – Valores de distorção e inércia



Fonte: Sousa, 2020.

Por meio dos gráficos de distorção e inércia, percebemos que o ponto de inflexão está no ponto k=3. Assim, a escolha de três agrupamentos, provavelmente, foi acertada, de maneira que, após treinarmos o modelo do k-means, é possível executar a redução da dimensão do vetor de características para visualizar melhor a distribuição dos agrupamentos. Podermos utilizar o modelo obtido para predizer em qual agrupamento alocar novas instâncias, precisamos saber o significado de cada um desses agrupamentos. Ou seja, quais são as características ou o comportamento que definem as instâncias que compõem cada agrupamento.

Inicialmente, devemos avaliar o qual bom foi o resultado do processo de definição dos agrupamentos. Uma forma de fazer isso é utilizando o coeficiente de Silhouette, que mede de forma simplificada o quão perto uma instância está daquelas que pertencem ao seu agrupamento e quão distante ela está dos demais agrupamentos. Assim, para obter o valor do coeficiente é necessário calcular para cada instância a distância média dela para todas as outras do agrupamento (d_c), bem como a distância média das instâncias do agrupamento mais próximo (d_e), de maneira que o coeficiente para cada instância **i** (S_i) pode ser definido pela fórmula:

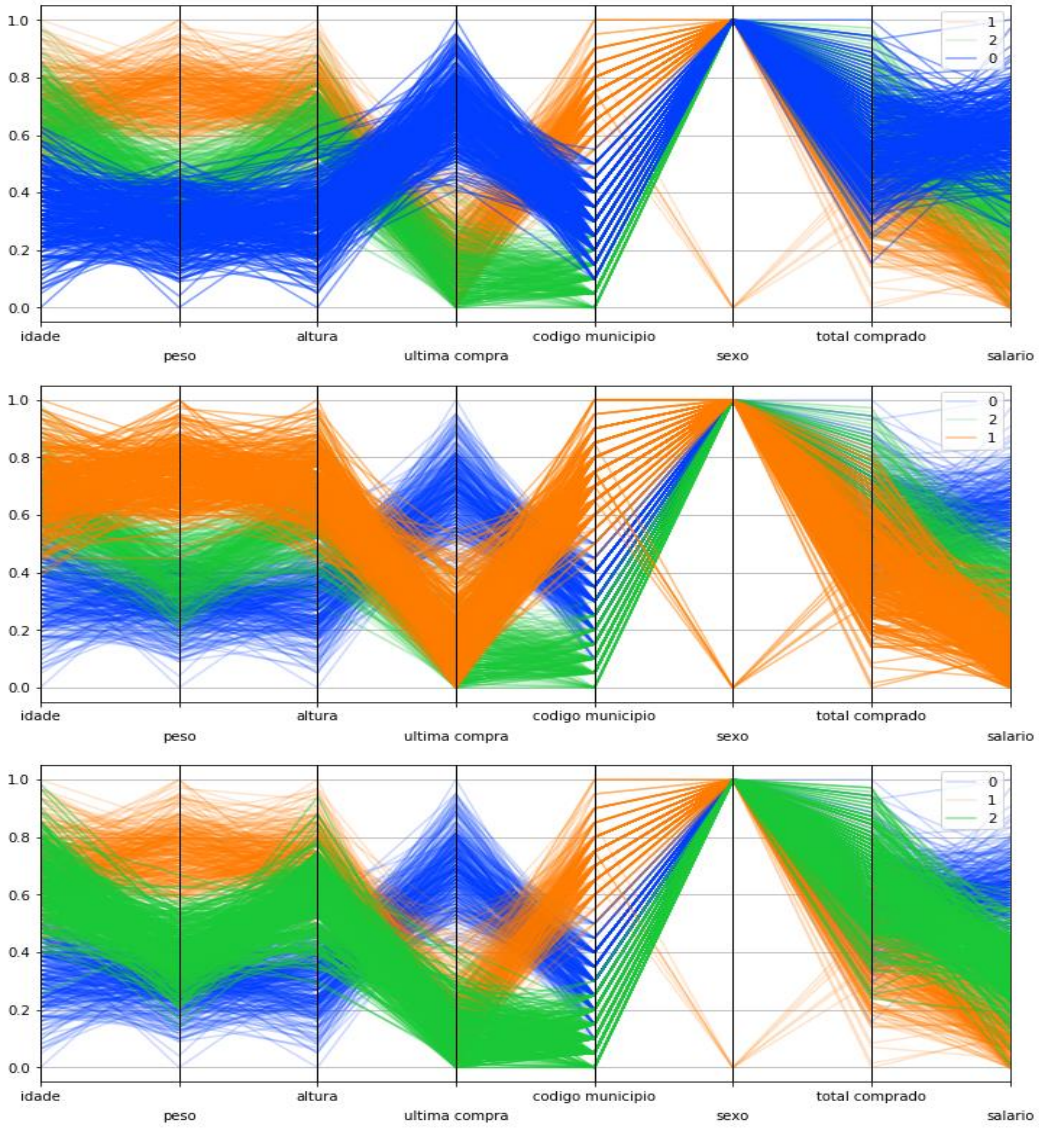
$$S(i) = \frac{d_c(i) - d_e(i)}{max\left(d_c(i), d_e(i)\right)}$$

O valor do coeficiente que mais se aproxima de 1 indica que existe uma separação entre os agrupamentos. Já quando o valor se aproxima de 0, indica que os pontos, mesmo pertencendo a diferentes agrupamentos, estão muito

próximos, havendo sobreposição de posições. Nem sempre serão obtidos valores do coeficiente próximos de 1, devendo sempre ser buscado o maior valor possível, pois muitas vezes a distribuição espacial dos dados permite apenas determinado intervalo de valores. Nesses casos, devemos avaliar se a retirada ou a inclusão de alguma característica pode melhorar os valores. Para isso, é preciso executar um processo de exploração e reavaliação dos dados.

Além do valor do coeficiente de Silhouette, também podemos avaliar os agrupamentos obtidos por intermédio de uma representação gráfica conhecida como plotagem de coordenadas paralelas. Essa representação consiste em plotar em um gráfico todos os valores das características de cada instância ao longo de um único eixo, dando destaque para um agrupamento por vez. Assim, é possível avaliar quais valores se destacam mais em cada agrupamento, dando uma ideia mais clara da formação destes. Na Figura 2, podemos ver a plotagem dos três agrupamentos que foram obtidos anteriormente para o *dataset* de clientes e suas compras.

Figura 2 – Coordenadas paralelas para três agrupamentos de clientes



Fonte: Sousa, 2020.

Com base na plotagem das coordenadas paralelas, podemos perceber que há uma separação entre os agrupamentos de acordo com o salário e uma separação clara em apenas dois grupos dos últimos valores comprados, bem como um viés de gênero indicando que boa parte dos clientes são apenas de um gênero.

TEMA 4 – AGRUPAMENTO HIERÁRQUICO

Agrupamento hierárquico é o nome dado a um conjunto de algoritmos que funciona criando agrupamentos aninhados, por meio de fusão ou divisão. Sua principal vantagem está no fato de não ser necessário definir o número de

agrupamentos na hierarquia entre os níveis, facilitando as etapas de exploração e avaliação das instâncias dos agrupamentos.

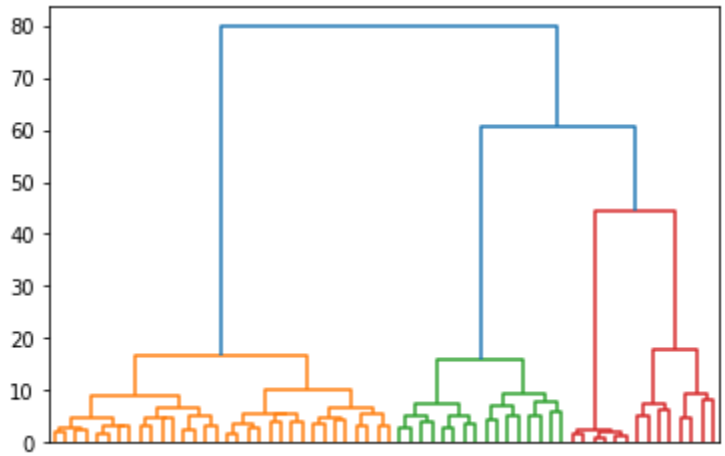
4.1 Algoritmo de agrupamento hierárquico

O agrupamento hierárquico é executado seguindo os seguintes passos.

1. Inicia com o número de agrupamentos (n) igual ao número total de instâncias, com cada agrupamento contendo apenas uma instância.
2. Agrupa os dois agrupamentos que são mais próximos um do outro, reduzindo para n-1 agrupamentos.
3. Recalcula as distâncias entre os agrupamentos.
4. Repete os passos 2 e 3 até que se tenha apenas um agrupamento com **n** instâncias.
5. Escolhe o número de agrupamentos que serão considerados.

Para visualizarmos os agrupamentos formados, é utilizada uma representação chamada de *dendrograma*, que monta uma estrutura de árvore hierárquica, conforme podemos ver na Figura 3.

Figura 3 – Dendrograma de clientes agrupados por idade



Fonte: Sousa, 2020.

4.2 Usando agrupamento hierárquico

A utilização de um agrupamento hierárquico é semelhante ao uso que fizemos do k-means, sendo necessário executar a normalização das características. O código a seguir executa o processo de normalização do

dataset de clientes e o agrupamento das instâncias utilizando um algoritmo de agrupamento hierárquico.

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import pandas as pd
import numpy as np

# Lê o conjunto de dados
df_clientes = pd.read_csv("dados_clientes_compras.csv")

# Normaliza os dados
scaler = StandardScaler()
clientes_norm = scaler.fit_transform(df_clientes.to_numpy())

# Cria o modelo de agrupamento hierarquico
hcluster = AgglomerativeClustering(distance_threshold=0, n_clusters=None,
affinity='euclidean', linkage='ward',  compute_full_tree=True)
clusters = hcluster.fit_predict(clientes_norm)
df_clientes['cluster'] = clusters
```

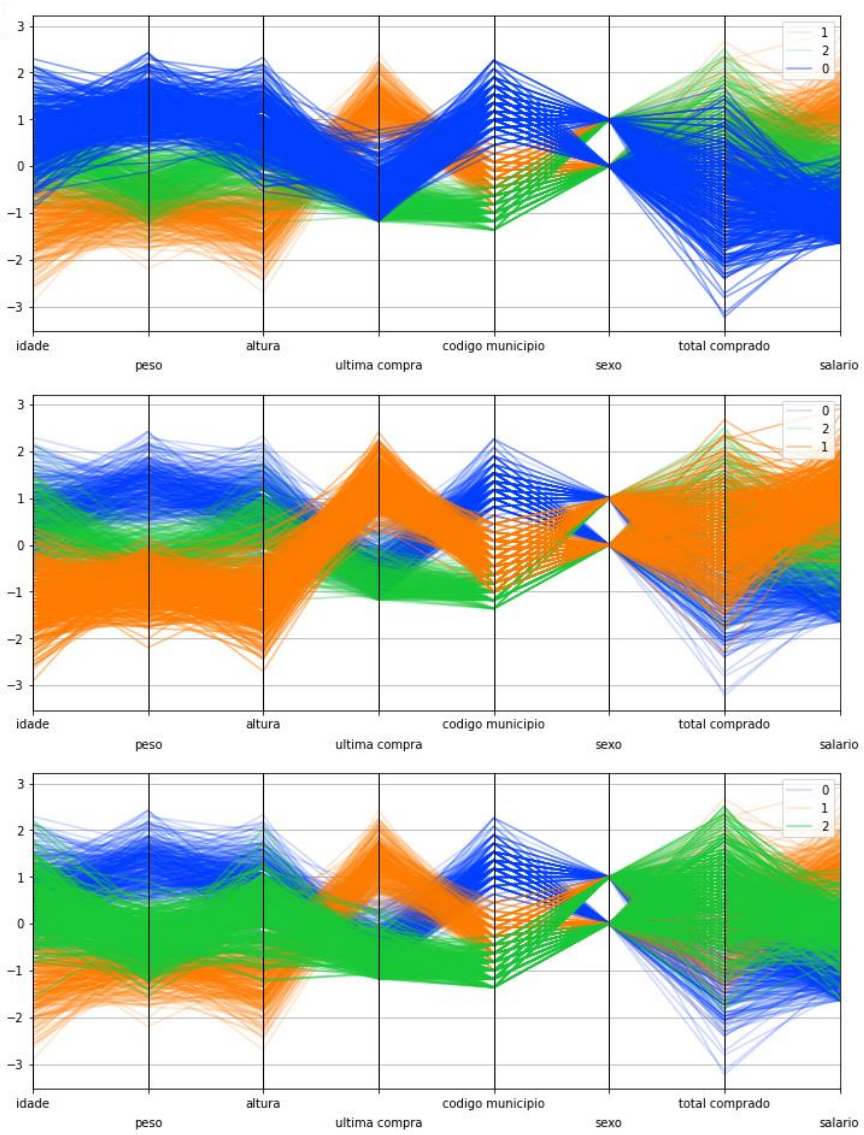
A utilização de clusterização hierárquica necessita da definição de uma métrica de distância (*affinity*) para ser utilizada e de um critério que define em que momento a distância entre os agrupamentos será calculada (*linkage*). No caso da implementação do *scikit-learn*, temos a possibilidade de indicar ao número de agrupamentos desejados (n_clusters), definindo assim a profundidade da árvore hierárquica que será construída.

Apesar de apresentar vantagens como a interpretabilidade, agrupamentos hierárquicos não devem ser aplicados em datasets muito grandes, pois sua complexidade computacional de tempo e espaço são elevadas.

4.3 Avaliando os agrupamentos

Após a obtenção dos agrupamentos, da mesma forma que nos outros métodos, necessitamos avaliar os agrupamentos para determinar os tipos de dados que foram agrupados e como estes podem ser utilizados. Para a avaliação podemos utilizar a plotagem de coordenadas paralelas, com resultando equivalente ao k-means, conforme podemos ver na Figura 4.

Figura 4 – Coordenadas paralelas para três agrupamentos



Fonte: Sousa, 2020.

O resultado dos algoritmos de agrupamento aparentemente não difere entre si, mas devemos nos atentar para o fato de que as instâncias que constituem cada agrupamento podem ser bem diferentes. Assim, se mais de um algoritmo for utilizado, deve-se considerar a possiblidade de comparar os resultados entre si.

TEMA 5 – OUTROS MÉTODOS NÃO SUPERVISIONADOS

O k-means apresenta vantagens como a sua facilidade de interpretação, escalabilidade para grandes conjuntos de dados e a garantia de convergência. Contudo, pode apresentar dificuldade no seu uso, em decorrência da sua capacidade de determinar apenas fronteiras lineares, da sua sensibilidade a

outliers e do fato de sua velocidade diminuir à medida que a quantidade de instâncias aumenta. Além dessas dificuldades, o k-means não é capaz de indicar sozinho o número ideal de agrupamentos e a obtenção desse valor pode ser algo difícil de determinar, mas, para os primeiros passos exploratórios sobre os dados, o modelo se apresenta como uma boa escolha.

Como forma de superar algumas das limitações do k-means, algumas propostas de alteração do próprio algoritmo podem ser utilizadas ou outros algoritmos não supervisionados de detecção de agrupamentos, que serão apresentados nesta seção.

5.1 Variações do k-means

Entre as variações do k-means, está o k-medoids, que apresenta o mesmo funcionamento. Entretanto, em vez de escolher um ponto no espaço de características como centroide, é escolhida uma instância – o medoide. Esse algoritmo é mais robusto que o k-means na presença de *outliers* e ruídos. O k-medoids também pode ser utilizado em combinação com o k-means, o que confere ao modelo final obtido maior capacidade de lidar com *outliers*.

A execução do k-means pode ter alto custo computacional e o tempo total para obtenção da posição final dos centroides pode ser muito alto à medida que o número de instância aumenta. Uma solução para obter maior velocidade é o k-means *mini-batches* que utiliza subconjuntos dos dados para reduzir o tempo de cálculo. Em diferentes etapas, um subconjunto diferente dos dados de entrada é escolhido e os centroides são recalculados. Isso faz com que a taxa de mudança da posição dos centroides seja reduzida, diminuindo o tempo de convergência. Contudo, a qualidade dos agrupamentos pode ser inferior ao k-means padrão.

Além das variações, algumas mudanças podem ser feitas no processo de inicialização do k-means ou nos seus parâmetros de execução como forma de melhor o desempenho. Por exemplo, o k-means++, que se trata da escolha inicial dos centroides, pode acelerar consideravelmente o processo de convergência. Assim, as variações do k-means, bem como a avaliação de diferentes parâmetros, pode ajudar na melhora do custo computacional e na qualidade dos agrupamentos obtidos.

5.2 Outros algoritmos não supervisionados

Além das variações do k-means, há vários outros algoritmos para determinação de agrupamentos. Entre alguns algoritmos comumente utilizados, podemos citar:

- *Density-Based Spatial Clustering of Applications with noise* (DBSCAN): funciona agrupando instâncias, baseando-se em uma métrica de distância e número mínimo de pontos, considerando a densidade da região do espaço em que cada instância se encontra. Dessa forma, instâncias em regiões com baixa densidade são consideradas como *outliers*. O DBSCAN não necessita do parâmetro contendo o número de agrupamentos inicialmente, sendo esta uma das suas características fortes, bem como o fato de conseguir determinar agrupamentos com organização não linear.
- *Balanced Iterative Reducing and Clustering Using Hierarchies* (Birch): funciona por meio da criação de uma estrutura de árvore de características para determinar os agrupamentos. Executa o treino do modelo de forma rápida e em uma única passagem e permite a identificação de *outliers*.
- Algoritmo de Maximização da Esperança: funciona de forma generativa. Em vez de, por meio das instâncias, determinar um agrupamento para cada instância com os modelos de cada agrupamento já determinado, avalia a qual agrupamento da instância se adequa melhor.

Além dos algoritmos aqui apresentados, há várias outras propostas de detecção de agrupamentos e variações, cada uma delas atendendo às necessidades específicas. Como sempre, é bom deixarmos claro que em aprendizagem de máquina não existe uma única solução para o problema, e nem sempre uma solução que funciona para uma situação funcionará para outras. Para fazer uma melhor escolha, devemos avaliar as soluções mais simples e, por meio delas, tentar identificar o que precisa ser melhorado.

FINALIZANDO

Nesta aula, apresentamos o funcionamento de alguns dos principais algoritmos de classificação não supervisionada, com especial atenção para o k-

means, que apesar de simples e ter algumas limitações ao lidar com dados de alta dimensionalidade, nos permite interpretar os resultados obtidos. Também apresentamos um algoritmo de agrupamento por aglomeração – agrupamento hierárquico – que pode ser utilizado em situações nas quais o k-means, por conta de suas limitações, não consiga obter os resultados esperados.

Foram apresentados conceitos e diferentes métricas de distância, cuja escolha é fundamental para algoritmos não supervisionados. No entanto, também é utilizado em aprendizagem de máquina como um todo, pois permite visualizar as conformações espaciais das instâncias dos dados em altas dimensões. Também exibimos como utilizar o PCA para executar uma redução da dimensionalidade dos dados e visualizarmo-nos melhor, bem como os agrupamentos obtidos.

Agora estamos preparados para lidar com problemas em que tenhamos informações de grupos ou classes nos quais os dados se inserem, bem como com situações em que não temos nenhuma informação desse tipo e dispomos apenas dos dados. Ainda, aprendemos como identificar o que os agrupamentos obtidos representam e como interpretar seu relacionamento com as características dos dados originais, sendo uma importante ferramenta para os mais diversos fins.

REFERÊNCIAS

CUESTA, H. **Practical data analysis**. Birmingham: Packt Publishing Ltd, 2013.

GÉRON, A. **Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow**: Concepts, tools, and techniques to build intelligent systems. Sebastopol: O'Reilly Media, 2019.

MCKINNEY, W. **Python para análise de dados**: tratamento de dados com Pandas, NumPy e IPython. São Paulo: Novatec, 2019.

RICHERT, W. **Building machine learning systems with Python**. Birmingham: Packt Publishing Ltd, 2013.

RUSSEL, S. J.; NORVIG, P. **Inteligência Artificial**: uma abordagem moderna. 3. ed. Tradução de Regina Célia Simille. Rio de Janeiro: Elsevier, 2013.