Exercícios de Clustering (Agrupamentos)

1 Procurando por clusters visualmente

Dado um vetor de pontos de dimensão 300 X 2, em que cada linha contém as coordenadas (x,y) de um ponto no mapa. Faça um gráfico de dispersão (scatter plot) desses pontos e use-o para estimar quantos clusters existem. Use o dataset CH1EX1.csv.

1) Carregue o dataset
2) Import o matplotlib.pyplot como plt
3)
    a) Crie um vetor chamado xs que contenha os valores dos pontos [:,0], ou seja, uma coluna 0 de pontos;
    b) Crie um vetor chamado ys que contenha os valores dos pontos [:,1], ou seja, uma coluna 1 de pontos;
4) Faça um scatter plot passando xs e ys para a função plt.scatter()
5) Chame a função plt.show() para exibir a plotagem

2 Agrupando pontos em duas dimensões

No *scatter plot* do exercício anterior, viu-se que os pontos parecem dividir-se entre 3 agrupamentos (clusters). Agora crie um modelo K-means para descobrir estes 3 clusters, e ajustá-los aos pontos do exercício anterior. Depois que o modelo for ajustado, determine os rótulos dos clusters para os pontos originais e também para os novos pontos, usando o método .predict( ). Use o dataset CH1EX2.csv.

1) Carregue o dataset
2) Import KMeans de sklearn.cluster
3) Usando KMeans( ), crie uma instância KMeans com o nome de *modelo* para achar 3 clusters. Use a palavra-chave *n_clusters* como argumento para especificar o número de clusters.
4) Use o método .fit( ) de *modelo* para ajustar o modelo ao vetor de pontos original.
5) Use o método .predict( ) de *modelo* para predizer os rótulos dos clusters dos pontos originais, atribuindo o resultado a um vetor *rotulos*.
6) Imprima os rótulos e observe-os. (No próximo exercício veremos como visualizar melhor os agrupamentos).
7) Use o método .predict( ) de modelo para predizer os rótulos dos clusters dos pontos novos, atribuindo o resultado a um vetor rotulos_novos. Note que KMeans pode atribuir previamente pontos nunca vistos a clusters que já foram determinados.

3 Inspecione seu agrupamento

Vamos inspecionar a clusterização que você realizou no exercício anterior. Use o dataset CH1EX2.csv.

1) Carregue o dataset
2) Execute a solução do exercício anterior
3) Import matplotlib.pyplot como plt
4) Atribua a coluna 0 dos pontos originais a xs e a coluna 1 dos mesmos pontos a ys
5) Faça um scatter plot de xs e ys, especificando o argumento da palavra chave *c = labels* para colorir os pontos com o rótulo de seu cluster. Você verá que o KMeans fez um bom trabalho em identificar os clusters.
6) Calcule as coordenadas dos centróides usando o atributo .cluster_center de modelo. Guarde-as em centroides.
7) Salve a coluna 0 de centroides em centroides_x e a coluna 1 de centroides a centroides_y.
8) Em uma única célula, crie dois *scatterplots* (isto fará com que um seja mostrado sobre o outro). Chame *plt.show( )* apenas uma vez, no final. Primeiro, faça o *scatterplot* que foi feito acima. Em seguida, faça o scatterplot de centroides_x e centroides_y, usando uma cruz ('X') como marcador, especificando o parâmetro *marker* para tal. Faça o tamanho dos marcadores ser 200 usando *s=200*.

4 Quantos clusters de grãos ?

Nos slides da aula vimos como escolher um número ótimo de clusters para um determinado dataset usando um gráfico de inércia em k-means. Aqui é dado um dataset contendo medidas de amostras de grãos (seeds.csv) . Qual é o número de clusters bom para esta caso?

1) Carregue o dataset
2) Mostre o DataFrame para inspecionar os dados. Note que existem 7 colunas - logo, cada amostra de um grão é um ponto em um espaço 7D ! Scatterplots não vão ser úteis aqui.
3) Extraia as medidas do DataFrame usando seu atributo values.
4) Meça a qualidade da clusterização com diferentes quantidades de clusters usando a inércia (inertia). Para cada um dos valores de k, execute os seguintes passos:
    a) crie uma instância KMeans com k clusters, chame-a de modelo
    b) ajuste o modelo às amostras dos dados de grãos
    c) inclua o valor do atributo inertia_ do modelo na lista inercias
5) Plote os valores das inércias para verificar qual número de clusters é o melhor. Lembre-se: quanto mais baixo o valor da inércia, melhor!

5 Avaliando a clusterização dos grãos

No exercício anterior, observamos na plotagem das inércias que 3 é um número bom de clusters para os dados dos grãos. Realmente, as amostras dos grãos vêm de uma mistura de três variedades de sementes: Kama, Rosa e Canadian. Neste exercício, agrupe as amostras em três e compare os clusters às variedades de grãos usando uma tabulação cruzada (cross-tabulation). Use o dataset seeds.csv.

1) Carregue o dataset. Há os arrays de amostras de grãos e uma lista de variedades indicando a variedade para cada amostra
2) Import KMeans
3) Crie um modelo KMeans, chame-o modelo com 3 clusters.
4) Use o método .fit_predict( ) de modelo para ajustar as amostras e inferir os rótulos dos grupos. Chamar .fit_predict( ) é o mesmo que chamar .fit( ) e em seguida chamar .predict ( ).
5) Crie um DataFrame df com duas colunas chamadas "rótulos" e “variedades”, usando os rótulos do dataset e os valores da lista de variedades, respectivamente para os valores das colunas.
6) Use a função pd.crosstab( ) em df['rotulos'] e em df['variedades'], para contar o número de vezes em que cada variedade de grão coincide com cada rótulo do cluster. Guarde o resultado em ct.
7) Mostre ct avaliando-o e inspecione sua tabulação cruzada. Note que sua clusterização está muito boa!

6 Pondo em escala dados de peixes para agrupá-los

É dado um conjunto de amostras de medições de peixes. Cada linha representa um único peixe. As medidas, tais como o peso em gramas, comprimento em centímetros e a proporção percentual entre altura e largura, estão em escalas muito diferentes. A fim de efetivamente agrupar estes dados, precisaremos antes padronizar essas características. Neste exercício, deve-se construir um pipeline para padronizar e agrupar os dados. Use o dataset FISH.csv.

1) Carregue o dataset.
2) Use **df.head( )** para inspecionar o dataset
3) Extraia todas as medidas na forma de um vetor NumPy de duas dimensões, armazenando-o em ***amostras*** (dica: use o atributo **.values** do DataFrame)
4) Execute as seguintes importações:
    a) **make_pipeline** de **sklearn.pipeline**
    b) **StandardScaler** de **sklearn.preprocessing**
    c) **KMeans** de **sklearn.cluster**
5) Crie uma instância de **StandardScaler** chamado ***escala***
6) Crie uma instância de **KMeans** com 4 clusters chamado ***kmeans***
7) Crie um pipeline chamado ***pipeline*** que conecte ***escala*** a ***kmeans***. Para tal, é necessário apenas passá-los como argumentos para **make_pipeline( )**