1️⃣

Lab001 (Clustering)

Agrupamento (Clustering)

Fundamentos

Agrupamento (Clustering) é uma técnica de aprendizado não supervisionado que visa dividir um conjunto de dados em grupos (clusters) de forma que os dados dentro de cada grupo sejam mais semelhantes entre si do que com os dados em outros grupos. A similaridade entre os dados é frequentemente determinada por cálculos de distância. Exemplos: Agrupamento de clientes com base em seus comportamentos de compra; Segmentação de imagens em partes distintas.

Clusters

- **Clusters** são os grupos formados durante o processo de agrupamento.
- Cada cluster deve conter objetos similares, e diferentes clusters devem conter objetos dissimilares.

Cálculos de Distância

- **City Block (Manhattan)**: Soma das diferenças absolutas entre as coordenadas correspondentes dos pontos.
- **Euclidiana**: Raiz quadrada da soma das diferenças quadradas entre as coordenadas correspondentes dos pontos.
- **Levenshtein**: Número mínimo de operações necessárias para transformar uma string em outra, considerando inserções, deleções e substituições de caracteres.

K-Means

K-Means é um dos algoritmos de agrupamento mais populares. Ele tenta dividir um conjunto de dados em k clusters, onde k é um parâmetro definido pelo usuário.

- **Passos do K-Means**:
    1. Escolha k centróides iniciais aleatoriamente.
    2. Atribua cada ponto de dados ao centróide mais próximo, formando k clusters.
    3. Recalcule os centróides de cada cluster.
    4. Repita os passos 2 e 3 até que os centróides não mudem mais significativamente.

Método de Elbow

Método de Elbow é usado para determinar o número ótimo de clusters k em K-Means. Ele envolve:

- Rodar o algoritmo K-Means para diferentes valores de k.
- Para cada k, calcular a soma das distâncias dos pontos ao centróide do cluster ao qual pertencem (WCSSE - Within-Cluster Sum of Squared Errors).
- Plotar o WCSSE em relação a k. O ponto onde a diminuição do WCSSE começa a se estabilizar (formando um "cotovelo") é escolhido como o k ótimo.

Avaliação do Modelo

WCSSE (Within-Cluster Sum of Squared Errors) é uma métrica que mede a variabilidade dentro dos clusters:

- Calcula a soma das distâncias quadradas de cada ponto ao centróide do cluster ao qual ele pertence.
- Um valor menor de WCSSE indica clusters mais compactos e, geralmente, uma melhor qualidade do agrupamento.

UA06 / LABORATÓRIO # 1

Importação das Bibliotecas

```
from sklearn.cluster import KMeans
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
```

- `sklearn.cluster.KMeans`: Para realizar a técnica de clustering K-Means.
- `pandas`: Para manipulação e análise de dados.
- `sklearn.preprocessing.MinMaxScaler`: Para normalização dos dados (não usado no código, mas importado).
- `matplotlib.pyplot`: Para visualização dos dados.

Carregamento e Visualização Inicial dos Dados

```
df = pd.read_csv("Mall_Customers.csv")
df.head()
```

Carrega os dados de clientes de um arquivo CSV e visualizar as primeiras linhas do DataFrame.

- `pd.read_csv`: Carrega o arquivo CSV em um DataFrame.
- `df.head()`: Exibe as primeiras 5 linhas do DataFrame.

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

Remoção da Coluna CustomerID e Informação sobre o DataFrame

```
df = df.drop('CustomerID', axis=1)
df.info()
```

Remove a coluna `CustomerID` do DataFrame e exibir informações sobre os dados.

- `df.drop`: Remove a coluna `CustomerID`.
- `df.info()`: Exibe informações como número de entradas, tipos de dados e valores nulos.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Gender                  200 non-null    object
 1   Age                     200 non-null    int64
 2   Annual Income (k$)      200 non-null    int64
 3   Spending Score (1-100)  200 non-null    int64
dtypes: int64(3), object(1)
memory usage: 6.4+ KB
```

Conversão de Dados Categóricos

```
df['Gender'] = df['Gender'].map({'Male': 0, 'Female': 1})
```

Converte a coluna categórica `Gender` em valores numéricos para facilitar a análise.

- `df['Gender'].map`: Mapeia `Male` para 0 e `Female` para 1.

Estatísticas Descritivas

```
df.describe()
```

Gera estatísticas descritivas para entender melhor a distribuição dos dados.

- `df.describe()`: Calcula estatísticas como média, desvio padrão, valores mínimo e máximo, e percentis.

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	0.560000	38.850000	60.560000	50.200000
std	0.497633	13.969007	26.264721	25.823522
min	0.000000	18.000000	15.000000	1.000000
25%	0.000000	28.750000	41.500000	34.750000
50%	1.000000	36.000000	61.500000	50.000000
75%	1.000000	49.000000	78.000000	73.000000
max	1.000000	70.000000	137.000000	99.000000

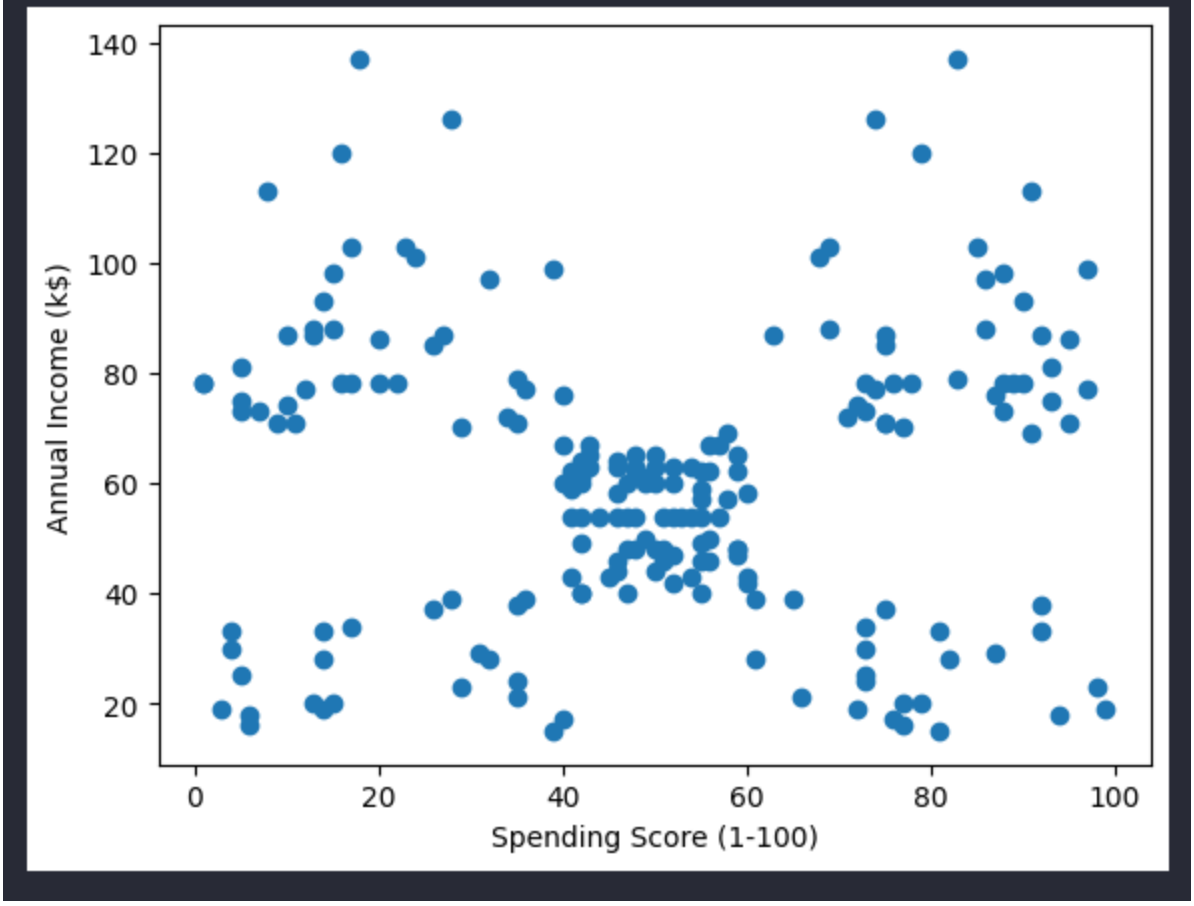
Visualização de Dados

```
plt.scatter(df['Spending Score (1-100)'],df['Annual Income (k$)
plt.xlabel("Spending Score (1-100)")
plt.ylabel('Annual Income (k$)')
```

Cria um gráfico de dispersão para visualizar a relação entre `Spending Score (1-100)` e `Annual Income (k$)`.

- `plt.scatter`: Plota um gráfico de dispersão.
- `plt.xlabel` e `plt.ylabel`: Adicionam rótulos aos eixos x e y.

```
Text(0, 0.5, 'Annual Income (k$)')
```



O gráfico de dispersão visualiza a relação entre `Spending Score (1-100)` (Pontuação de Gastos) e `Annual Income (k$)` (Renda Anual) dos clientes. As principais observações são:

1. **Distribuição Geral**: Os clientes apresentam uma ampla variedade de pontuações de gastos e rendas anuais.
2. **Concentração de Pontos**: Há uma alta densidade de pontos no centro do gráfico, indicando muitos clientes com renda anual entre 60 e 80 mil dólares e pontuação de gastos entre 40 e 60.
3. **Segmentação Visual**: Diferentes segmentos de clientes são sugeridos, como:
    - Clientes de baixa renda e baixa pontuação de gastos.
    - Clientes de alta renda e alta pontuação de gastos.

- Clientes de renda média com pontuação de gastos variável.

4. **Variabilidade**: Alta variabilidade nas pontuações de gastos em todas as faixas de renda.
5. **Outliers**: Alguns pontos isolados podem representar comportamentos atípicos de clientes.

Implicações

- **Clustering**: O gráfico sugere que aplicar técnicas de clustering, como K-Means, pode ser útil para identificar segmentos distintos de clientes.
- **Método Elbow**: Confirmar o número ótimo de clusters para melhor segmentação.
- **K-Means**: Aplicar K-Means para identificar e visualizar os clusters, ajudando a entender melhor os padrões de comportamento dos clientes.

Método Elbow para Determinação do Número Ótimo de Clusters

```
sse = []
k_rng = range(1,10)
for k in k_rng:
    km = KMeans(n_clusters=k)
    km.fit(df[['Spending Score (1-100)','Annual Income (k$)']])
    sse.append(km.inertia_)
print(sse)
```

Utiliza o Método Elbow para determinar o número ótimo de clusters.

- Calcula o WCSSE (Within-Cluster Sum of Squared Errors) para diferentes valores de k (de 1 a 9).

  kk

- `km.inertia_`: Adiciona o valor de inércia (WCSSE) à lista `sse`.

Output: [269981.28, 184740.3962731454, 106348.37306211119, 73679.78903948834, 44454.47647967974, 37455.98455516028,

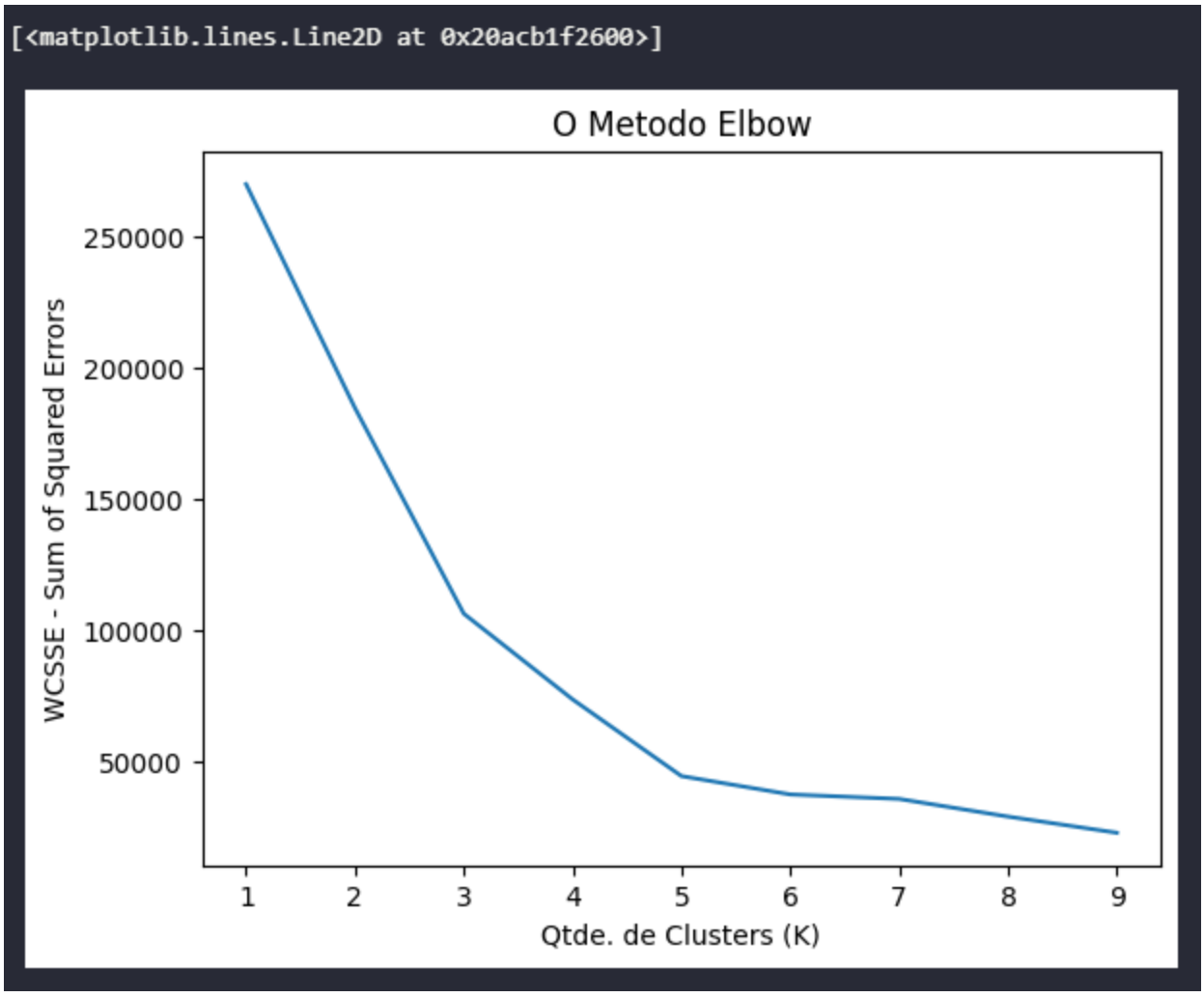
35747.895340240706, 29024.568499838882, 22858.490863027135]

Visualização do Método Elbow

```
plt.title('O Metodo Elbow')
plt.xlabel('Qtde. de Clusters (K)')
plt.ylabel('WCSSE - Sum of Squared Errors')
plt.plot(k_rng,sse)
```

Plotar o gráfico do Método Elbow para ajudar a visualizar o número ótimo de clusters.

- `plt.plot`: Cria o gráfico do Método Elbow.



É possível observar como a soma dos erros quadrados dentro dos clusters (WCSSE - Within-Cluster Sum of Squared Errors) diminui à medida que o número de clusters (K) aumenta. Esse método é frequentemente utilizado para ajudar a determinar o número ótimo de clusters em que um conjunto de dados deve ser dividido ao aplicar o algoritmo de K-Means.

Análise do Gráfico:

- **Declínio Acentuado:** O gráfico mostra um declínio acentuado do WCSSE conforme o número de clusters aumenta de 1 para 2 e de 2 para 3. Isso indica que cada adição de cluster nesse estágio reduz significativamente a variância dentro dos clusters.
- **Ponto de Cotovelo:** O "cotovelo", ou seja, o ponto após o qual o declínio no WCSSE se torna menos acentuado, parece ocorrer em torno de K=3 ou K=4. Este é o ponto onde aumentar o número de clusters deixa de resultar em ganhos significativos na redução do WCSSE, sugerindo que o benefício marginal de adicionar mais clusters é reduzido.
- **Estabilização:** Após K=4, a curva começa a se achatar, indicando que adicionar mais clusters não melhora substancialmente a compactação dos grupos.

Conclusões:

- **Número Ótimo de Clusters:** Com base no gráfico, pode-se considerar que o número ótimo de clusters para esse conjunto de dados específico está entre 3 e 4. Escolher um desses valores para K permitiria uma boa segmentação do conjunto de dados enquanto mantém uma quantidade razoável de WCSSE.
- **Seleção de K:** A escolha entre K=3 ou K=4 pode depender de outras considerações, como a interpretabilidade dos clusters ou restrições específicas do problema que está sendo resolvido.

Este método fornece uma maneira visual e intuitiva de estimar o número de clusters que pode ser mais apropriado para o conjunto de dados em análise, facilitando a tomada de decisão na aplicação de técnicas de clustering.

Aplicação do K-Means com Número Ótimo de Clusters

```
km = KMeans(n_clusters=4)
y_predicted = km.fit_predict(df[['Gender','Age', 'Spending Score
y_predicted
```

Aplica o algoritmo K-Means com 4 clusters (com base na análise do Elbow) e prever os clusters para os dados.

- `km.fit_predict`: Ajusta o modelo e prevê os clusters.

```
array([1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 3, 3, 2, 3, 2, 3, 2, 3, 2, 3,
       2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3,
       2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3,
       2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3,
       2, 3])
```

Visualização dos Centros dos Clusters

```
km.cluster_centers_
```

Obtém os centros dos clusters após o ajuste do modelo.

- `km.cluster_centers_`: Retorna as coordenadas dos centros dos clusters.

```
array([[ 0.5942029 , 52.05797101, 39.88405797, 46.42028986],
       [ 0.58490566, 25.05660377, 62.62264151, 40.73584906],
       [ 0.47368421, 40.39473684, 18.63157895, 87.        ],
       [ 0.55      , 32.875     , 81.525     , 86.1       ]])
```

Adicionando Informações dos Clusters ao DataFrame

```
df['cluster']=y_predicted
df.head()
```

Adiciona a coluna `cluster` ao DataFrame original para indicar a qual cluster cada ponto pertence.

- `df['cluster']`: Cria uma nova coluna `cluster` com os valores previstos.
- `df.head()`: Exibe as primeiras 5 linhas do DataFrame atualizado.

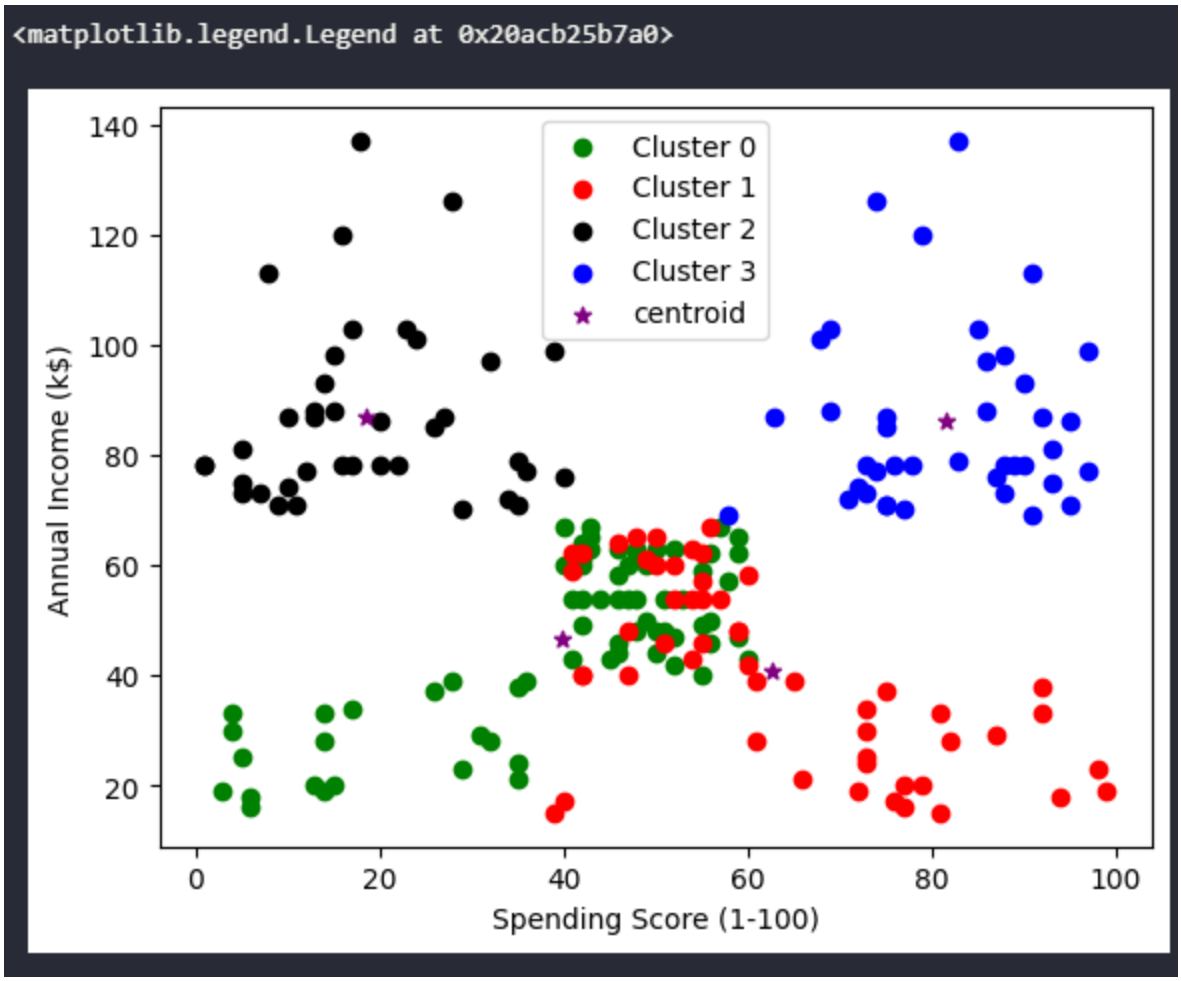
	Gender	Age	Annual Income (k$)	Spending Score (1-100)	cluster
0	0	19	15	39	1
1	0	21	15	81	1
2	1	20	16	6	0
3	1	23	16	77	1
4	1	31	17	40	1

Visualização dos Clusters

```
df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]
df4 = df[df.cluster==3]
plt.scatter(df1['Spending Score (1-100)'],df1['Annual Income (k$
plt.scatter(df2['Spending Score (1-100)'],df2['Annual Income (k$
plt.scatter(df3['Spending Score (1-100)'],df3['Annual Income (k$
plt.scatter(df4['Spending Score (1-100)'],df4['Annual Income (k$
plt.scatter(km.cluster_centers_[:,2],km.cluster_centers_[:,3],co
plt.xlabel("Spending Score (1-100)")
plt.ylabel('Annual Income (k$)')
plt.legend()
```

Visualiza os clusters e seus centróides em um gráfico de dispersão.

- Filtra o DataFrame por clusters (`df1` , `df2` , `df3` , `df4`).
- Plota os pontos de cada cluster em cores diferentes.
- Adiciona os centróides ao gráfico como estrelas (`*`).
- Adiciona rótulos e legenda para melhor entendimento do gráfico.



O gráfico mostra a visualização dos clusters formados pelo algoritmo K-Means, os dados estão coloridos de acordo com o cluster ao qual cada ponto pertence. Os centróides de cada cluster são marcados com estrelas roxas.

Análise dos Clusters:

- **Cluster 0 (Cor Verde):** Este cluster agrupa clientes com uma pontuação de gastos moderada e uma renda anual baixa. A maioria dos pontos está concentrada entre uma pontuação de gastos de 40 a 60 e renda anual de menos de 40k$.
- **Cluster 1 (Cor Vermelha):** Formado por clientes com pontuação de gastos alta e renda anual baixa. Os pontos estão concentrados principalmente entre uma pontuação de gastos de 60 a 100 e renda anual abaixo de 40k$.
- **Cluster 2 (Cor Preta):** Este cluster inclui clientes com uma pontuação de gastos baixa e renda anual variável, mas geralmente mais alta do que os outros clusters. A pontuação de gastos varia de 0 a 40, enquanto a renda anual vai de 40k$ a cerca de 140k$.
- **Cluster 3 (Cor Azul):** Engloba clientes com pontuação de gastos alta e renda anual alta. A pontuação de gastos varia de 60 a 100, e a renda anual é consistentemente acima de 60k$.

Características dos Centróides:

- **Localização dos Centróides:** Os centróides (marcados com estrelas roxas) indicam o "centro" de cada cluster, representando a média da pontuação de gastos e da renda anual para os membros de cada cluster. Os centróides de cada cluster estão localizados de maneira a minimizar a distância total de todos os pontos no cluster até este ponto central.

Implicações:

- **Segmentação de Cliente:** O agrupamento demonstra uma segmentação clara do mercado baseada na renda anual e na pontuação de gastos dos clientes. Isso pode ser usado por um shopping ou varejista para desenvolver estratégias de marketing e vendas direcionadas.
- **Estratégias de Marketing:** Por exemplo, o cluster 1 pode ser alvo de promoções e ofertas especiais para incentivar compras maiores, visto que têm alta pontuação de gastos, enquanto o cluster 2 pode requerer estratégias para aumentar a frequência de compras ou o valor gasto.

Conclusão:

O gráfico fornece uma visualização clara e intuitiva de como o algoritmo K-Means agrupou os clientes com base em duas variáveis importantes. Essa segmentação permite que decisões estratégicas sejam tomadas com base nas características comuns dentro de cada cluster, potencializando a eficácia das campanhas de marketing e o entendimento do comportamento do consumidor.