1 - A) Considerando centros aleatórios:

```
Centros iniciais selecionados (índices): [ 0 17 15]
Centros iniciais (coordenadas):
[[ 7.82 -4.58 -3.97]
 [-6.38 -1.74  1.43]
 [ 6.81  0.17 -4.15]]
Convergência alcançada após 6 iterações.
Centroides finais:
[[ 5.80428571 -0.41        4.03285714]
 [-6.84111111  0.39333333 -0.45444444]
 [ 6.905      -1.1025     -3.735     ]]

Distribuição dos pontos por cluster:
Cluster 1: [[ 4.36  2.19  2.09]
 [ 6.72  0.88  2.8 ]
 [ 4.47 -2.62  5.76]
 [ 6.73 -2.01  4.18]
 [ 6.18 -2.81  5.82]
 [ 8.09  0.2   2.25]
 [ 4.08  1.3   5.33]]
Cluster 2: [[-6.68  3.16  2.71]
 [-8.64 -3.06  3.5 ]
 [-6.87  0.57 -5.45]
 [-7.71  2.34 -6.33]
 [-6.91 -0.49 -5.68]
 [-6.25 -0.26  0.56]
 [-6.94 -1.22  1.13]
 [-5.19  4.24  4.04]
 [-6.38 -1.74  1.43]]
Cluster 3: [[ 7.82 -4.58 -3.97]
 [ 6.72 -0.93 -4.04]
 [ 6.81  0.17 -4.15]
 [ 6.27  0.93 -2.78]]
```

B) Considerando os centros $m_1 = (0, 0, 0)^t$, $m_2 = (0, -1, -1)^t$, $m_3 = (1, 1, 2)^t$.

```
Convergência alcançada após 3 iterações.
Centroides finais:
[[-6.84111111  0.39333333 -0.45444444]
 [ 6.905      -1.1025     -3.735     ]
 [ 5.80428571 -0.41        4.03285714]]

Distribuição dos pontos por cluster:
Cluster 1: [[-6.68  3.16  2.71]
 [-8.64 -3.06  3.5 ]
 [-6.87  0.57 -5.45]
 [-7.71  2.34 -6.33]
 [-6.91 -0.49 -5.68]
 [-6.25 -0.26  0.56]
 [-6.94 -1.22  1.13]
 [-5.19  4.24  4.04]
 [-6.38 -1.74  1.43]]
Cluster 2: [[ 7.82 -4.58 -3.97]
 [ 6.72 -0.93 -4.04]
 [ 6.81  0.17 -4.15]
 [ 6.27  0.93 -2.78]]
Cluster 3: [[ 4.36  2.19  2.09]
 [ 6.72  0.88  2.8 ]
 [ 4.47 -2.62  5.76]
 [ 6.73 -2.01  4.18]
 [ 6.18 -2.81  5.82]
 [ 8.09  0.2   2.25]
 [ 4.08  1.3   5.33]]
```

C) Considerando os centros $m_1 = (-0.1, 0, -0.1)^t$, $m_2 = (0, -2, -3)^t$, $m_3 = (1, -3, -5)^t$.

```
Convergência alcançada após 6 iterações.
Centroides finais:
[[ 5.80428571 -0.41        4.03285714]
 [-6.84111111  0.39333333 -0.45444444]
 [ 6.905      -1.1025     -3.735     ]]

Distribuição dos pontos por cluster:
Cluster 1: [[ 4.36  2.19  2.09]
 [ 6.72  0.88  2.8 ]
 [ 4.47 -2.62  5.76]
 [ 6.73 -2.01  4.18]
 [ 6.18 -2.81  5.82]
 [ 8.09  0.2   2.25]
 [ 4.08  1.3   5.33]]
Cluster 2: [[-6.68  3.16  2.71]
 [-8.64 -3.06  3.5 ]
 [-6.87  0.57 -5.45]
 [-7.71  2.34 -6.33]
 [-6.91 -0.49 -5.68]
 [-6.25 -0.26  0.56]
 [-6.94 -1.22  1.13]
 [-5.19  4.24  4.04]
 [-6.38 -1.74  1.43]]
Cluster 3: [[ 7.82 -4.58 -3.97]
 [ 6.72 -0.93 -4.04]
 [ 6.81  0.17 -4.15]
 [ 6.27  0.93 -2.78]]
```

A letra “a” considera centros iniciais aleatórios, o que pode gerar inconsistência nos resultados, levando a resultados piores ou melhores do que a letra “c”, no entanto não temos garantia do que vai ocorrer, enquanto na letra “c” já temos pontos determinados, com resultados consistentes, no entanto, seria interessante avaliar o desempenho dos pontos fixos, uma vez que, escolhido de forma errada, podem levar à resultados ruins, que diferente, da opção “a” não tem a possibilidade de mudança.

2) Temos portanto 4 distribuições Gaussianas, C1,C2,C3 e C4, que devem ser gerados a partir dos centroides:
m1 = (0,0,0,0,0,0,0,0)
m2 = (4,0,0,0,0,0,0,0)
m3 = (0,0,0,4,0,0,0,0)
m4 = (0,0,0,0,0,0,0,4)

A associação assumida foi de C1 para m1, C2 para m2 e assim por diante. Sendo assim, geramos as 4 distribuições considerando 100 pontos para cada uma delas, como podemos ver na Figura X abaixo.

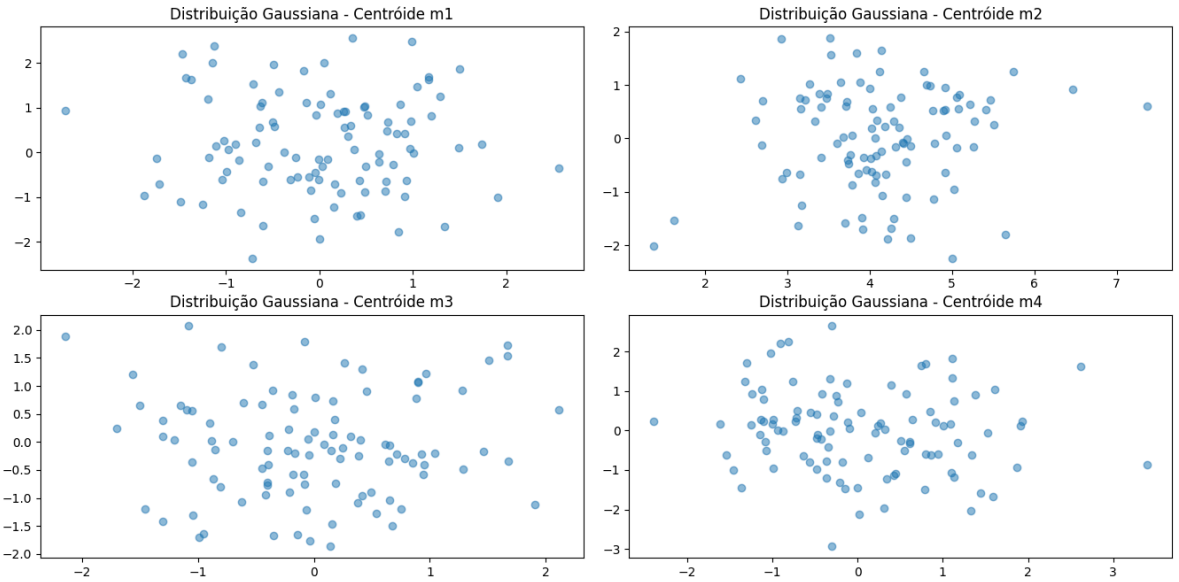

Figura X - Distribuições Gaussianas

Ao todo foram gerados 400 pontos. Após a geração das distribuições podemos criar o SOM, para isso consideramos uma grade com 100 neurônios (10x10) e 10000 iterações para treinar os dados. O SOM gerado pode ser visualizado na Figura X abaixo.

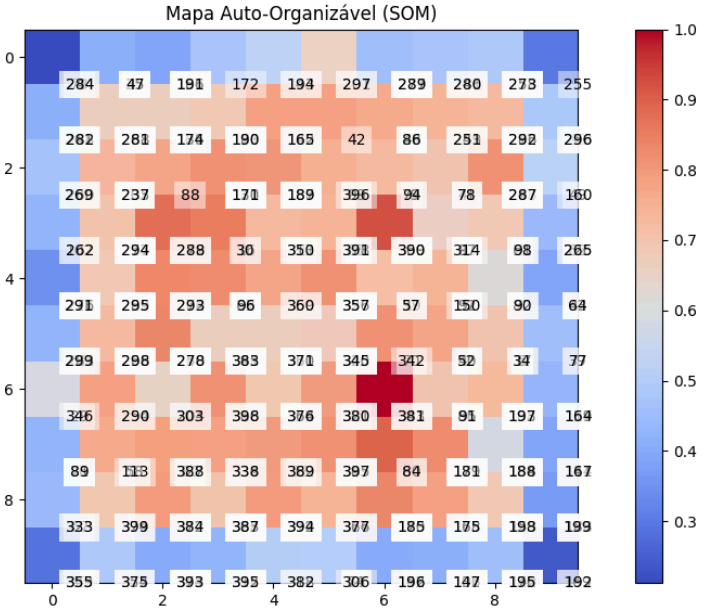

Figura X - SOM

3) Após implementar a clusterização hierárquica pudemos gerar a seguinte imagem contendo os clusters:

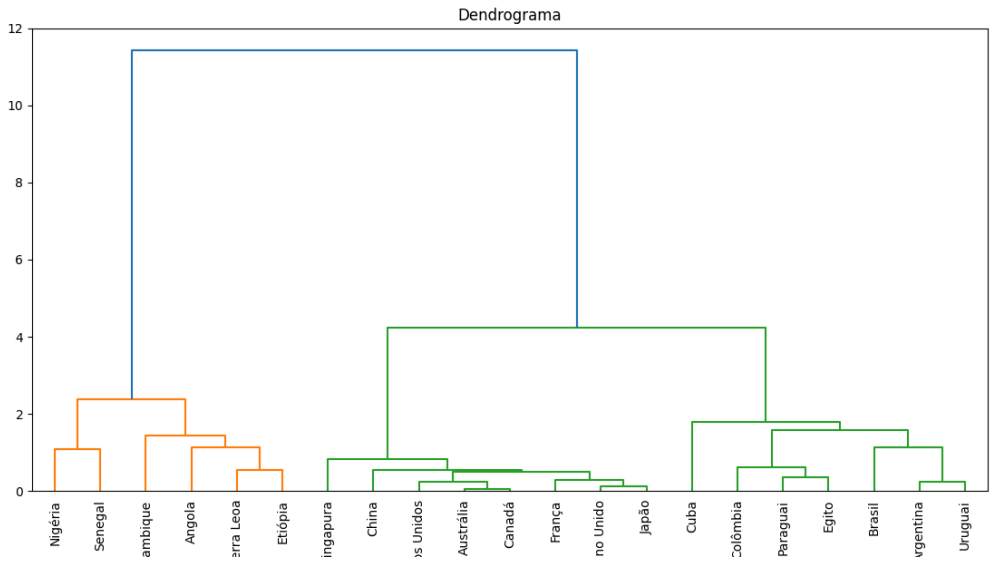

Figura 1 - Dendrograma dos clusters

A partir da Figura 1 podemos perceber todos os clusters com ênfase nos clusters:

Primeiro Cluster:

Composto pelos países: Nigéria, Senegal, Moçambique, Angola, Serra Leoa, Etiópia. Esses países têm características socioeconômicas e geográficas semelhantes. Eles podem compartilhar desafios e oportunidades similares em termos de desenvolvimento e estabilidade.

Segundo Cluster:

Composto pelos países: Singapura, China, Estados Unidos, Austrália, Canadá, França, Reino Unido, Japão.Esses países são geralmente mais desenvolvidos e têm uma influência global significativa. Eles podem ter altos índices de expectativa de vida, educação e PIB, além de maior estabilidade política.

Terceiro Cluster:

Composto pelos países: Cuba, Colômbia, Paraguai, Egito, Brasil, Argentina, Uruguai. Esses países podem ter características socioeconômicas intermediárias. Eles podem compartilhar desafios comuns em termos de desenvolvimento econômico e social, mas também têm potencial de crescimento e desenvolvimento.

Com isso também pudemos obter os valores de média e desvio-padrão:

Média	0.68	0.77	0.68	0.18
Desvio Padrão	0.24	0.26	0.22	1.34

Por fim podemos gerar o dendrograma do SOM conforme mostra a Figura 2:

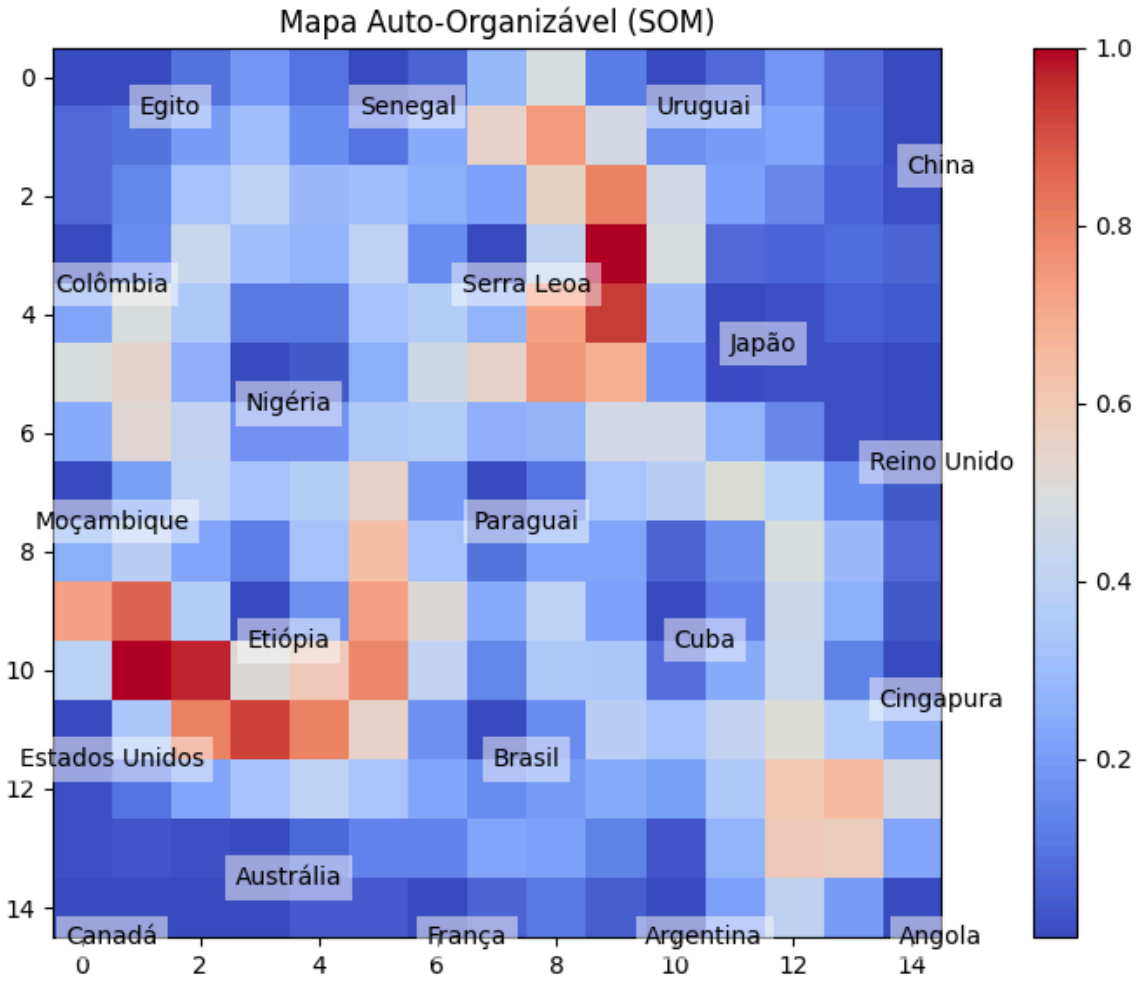

Figura 2 - Dendrograma SOM

Os dados foram treinados por 50000 iterações e também com uma grade de 225 neurônios (15x15), os pesos foram escolhidos de forma aleatória. O mapa mostra a proximidade ou similaridade entre os países, onde países com características semelhantes estão mais próximos uns dos outros na grade. Por exemplo, Serra Leoa e Etiópia estão em áreas de cor vermelha, indicando uma alta similaridade ou valor alto em alguma característica específica. Por outro lado, países como China e Japão estão em áreas de cor azul, indicando valores mais baixos ou menor similaridade com os países em vermelho.

4)

As técnicas de redução de dimensionalidade são técnicas utilizadas em ciência de dados e aprendizado de máquina, a fim de reduzir a complexidade de conjuntos de dados com muitas variáveis ou características, diminuindo suas dimensões ao mesmo que tempo que preserva as informações mais relevantes, melhorando a eficiência computacional, reduzindo as chances de overfitting e facilitando a visualização.

Um dos principais métodos de redução de dimensionalidade é a análise de componentes principais (PCA), que transforma o conjunto de dados para um novo sistema de coordenadas, usando combinações lineares das variáveis originais e eliminando redundâncias e correlações entre variáveis.

O PCA busca obter uma base ortogonal, de forma que a variância dos dados em cada direção seja maximizada, transformando os dados para um sistema de coordenadas onde a primeira componente é a direção de maior variância dos dados e a segunda componente é a direção de maior variância restante, sendo ortogonal a primeira e assim por diante. Um exemplo disso seria um conjunto de dados de altura e peso, que com o uso do PCA pode resultar em uma combinação linear das duas variáveis, representando 95% da variação total, como tamanho físico que pode relacionar a combinação das duas variáveis.

Outro método é a análise discriminante linear (LDA), que reduz as dimensões encontrando direções que maximizam a separação entre classes, ou seja, separando as características que melhor distinguem cada classe de objeto, sendo geralmente usado em problemas de classificação.

LDA é uma técnica supervisionada que tenta maximizar a distância média entre as classes e minimizar a dispersão dentro de cada classe, mantendo os grupos bem separados, como na identificação de dígitos escritos à mão, onde o LDA tenta encontrar dimensões que distinguem bem cada um dos números. Por isso é um método bastante utilizado em problemas de classificação, como reconhecimento de imagens.

Outras técnicas são as fatorações de matriz como NMF (Non-Negative Matrix Factorization) ou SVD (Singular Value Decomposition), que fatoram os dados em matrizes menores ou decompõe a matriz em dados de valores singulares, respectivamente.

Os Self-Organizing Maps (SOMs), também são outras técnicas importantes, que consistem em redes neurais não supervisionadas que projetam dados de alta dimensionalidade em um espaço de dimensão menor, como um plano bidimensional.

O SOM utiliza uma rede neural não supervisionada, mapeando os dados de alta dimensionalidade para um espaço 2D ou 3D, assim, os neurônios da grade 2D ou 3D, formam clusters, que representam subconjuntos dos dados. Um exemplo disso, é o mapeando de perfis de clientes, que ao invés de considerar milhares de clientes diferentes, cria células que representan grupos de clientes similares

No geral as aplicações de técnicas de redução de dimensionalidade variam, reduzindo pixels de uma imagem enquanto preservam a informação, reduzindo textos sem que o sentido seja perdido ou até simplificando conjuntos de dados genéticos para pesquisa de alguma doença, por exemplo. A fim de exemplificar as aplicações e desempenho de cada técnica citada, o artigo **Redução de dimensionalidade em bigdata**.

O trabalho selecionado utiliza e avalia diferentes métodos para redução de dimensionalidade, como PCA, Kernel PCA e LDA. Além de descrever cada uma das técnicas, o artigo compara seu desempenho utilizando bases estruturadas e não estruturadas (imagens). Dessa forma, para uma base estruturada, o PCA mostrou-se mais eficiente preservando informações e melhorando o tempo de processamento, enquanto o

Kernel PCA foi menos eficiente e mais custoso computacionalmente e o LDA teve um bom desempenho em classificações, porém se limita pela sua baixa flexibilidade. Já para bases não estruturadas o PCA destacou-se na redução de imagens com perdas de qualidade mínimas.

5) O **aprendizado semi-supervisionado** trata-se de uma abordagem de aprendizado de máquina que utiliza tanto dados rotulados quanto não rotulados para treinar um modelo. Ele busca tirar proveito da grande quantidade de dados não rotulados disponíveis, complementando-os com um número menor de dados rotulados, que são mais caros e demorados de obter. Esse tipo de aprendizado é especialmente útil em cenários onde é difícil ou caro rotular grandes quantidades de dados, mas ainda há muitos dados não rotulados disponíveis.

Ele funciona a partir de dados rotulados e não rotulados e tem por objetivo usar os dados rotulados para aprender a estrutura do problema, e depois aproveitar os dados não rotulados para melhorar o modelo e torná-lo mais robusto, ajudando o modelo a generalizar melhor para novos dados. Para isso pode-se usar métodos como Label Propagation, Autoencoders e Redes Neurais ou também Modelos Generativos.

O treino consiste em 3 fases onde:

Fase 1: Treino com dados rotulados;
Fase 2: Treino com dados não rotulados;
Fase 3: Reforço e ajustes.

A aplicação encontrada está no artigo **Improving Activity Classification for Health Applications on Mobile Devices using Active and Semi-Supervised Learning**. Este artigo explora o uso de redes neurais e métodos de aprendizado semi-supervisionado no contexto da segmentação de imagens de células, um desafio comum em imagens biomédicas, especialmente na análise de imagens microscópicas. O artigo propõe uma nova abordagem para segmentação de células, combinando aprendizado supervisionado e semi-supervisionado, com o objetivo de melhorar a precisão e a robustez do modelo.

A introdução do artigo detalha o problema da segmentação de células em imagens microscópicas, que é fundamental para várias aplicações biomédicas, como a análise de tecidos e a detecção de doenças. Embora os métodos tradicionais de segmentação, como a limiarização e o clustering, sejam eficazes em cenários simples, eles se mostram ineficazes em imagens complexas, que contêm ruído e variabilidade. Por isso, os métodos baseados em aprendizado de máquina, especialmente redes neurais profundas, têm se mostrado promissores para lidar com essas complexidades.

O ponto central da pesquisa é a proposta de um método de aprendizado semi-supervisionado para melhorar a segmentação de imagens de células. A rede neural é treinada com uma combinação de aprendizado supervisionado (para as imagens rotuladas) e uma técnica de auto-supervisão (para as imagens não rotuladas). Essa combinação de abordagens permite que o modelo aprenda de maneira mais eficaz e generalize melhor, lidando com a variabilidade dos dados de imagens de células.

Os resultados experimentais apresentados no artigo demonstram que a abordagem semi-supervisionada supera os métodos totalmente supervisionados em termos de precisão

e robustez, especialmente quando há um número limitado de dados rotulados. O modelo semi-supervisionado consegue usar eficientemente os dados não rotulados para melhorar a segmentação, apresentando resultados mais consistentes em diferentes conjuntos de dados.

TABLE I
PERCENTAGE CHANGE FROM BASE CLASSIFIER WITH 480 NEW DATAPOINTS OVER EIGHT ITERATIONS AND A CONFIDENCE INTERVAL OF 95%

	Self-Learning	Active Learning	En-Co-Training		Democratic Co-learning	
Unlabeled	DT only	DT only	DT only	Democratic	DT only	Democratic
50%	$-1.27\% \pm 2.07\%$	$2.15\% \pm 2.85\%$	$-0.91\% \pm 2.15\%$	$-0.34\% \pm 2.67\%$	$-2.06\% \pm 3.08\%$	$-0.63\% \pm 2.85\%$
55%	$-5.35\% \pm 5.66\%$	$3.17\% \pm 4.87\%$	$-6.64\% \pm 6.46\%$	$0.67\% \pm 0.66\%$	$-1.46\% \pm 3.14\%$	$0.38\% \pm 0.87\%$
60%	$3.31\% \pm 4.41\%$	$17.13\% \pm 7.95\%$	$5.53\% \pm 5.29\%$	$13.05\% \pm 7.20\%$	$14.38\% \pm 8.31\%$	$15.07\% \pm 8.00\%$
65%	$0.05\% \pm 0.28\%$	$12.38\% \pm 7.28\%$	$0.88\% \pm 1.66\%$	$6.34\% \pm 3.43\%$	$8.59\% \pm 8.08\%$	$10.48\% \pm 6.34\%$
70%	$0.17\% \pm 0.54\%$	$9.35\% \pm 6.41\%$	$0.04\% \pm 0.58\%$	$5.04\% \pm 3.14\%$	$7.99\% \pm 5.76\%$	$8.41\% \pm 5.82\%$
75%	$3.31\% \pm 4.41\%$	$9.79\% \pm 6.44\%$	$1.65\% \pm 6.51\%$	$6.69\% \pm 4.61\%$	$9.03\% \pm 6.31\%$	$9.12\% \pm 6.31\%$
80%	$-0.02\% \pm 0.03\%$	$1.48\% \pm 2.31\%$	$-0.01\% \pm 0.03\%$	$1.14\% \pm 0.80\%$	$0.54\% \pm 1.40\%$	$1.03\% \pm 1.11\%$
85%	$1.38\% \pm 1.87\%$	$8.77\% \pm 6.57\%$	$0.23\% \pm 0.55\%$	$5.45\% \pm 3.51\%$	$7.80\% \pm 6.40\%$	$8.84\% \pm 6.12\%$
90%	$-0.63\% \pm 0.89\%$	$3.13\% \pm 4.50\%$	$0.10\% \pm 1.54\%$	$1.41\% \pm 1.56\%$	$0.51\% \pm 2.15\%$	$1.02\% \pm 1.95\%$
95%	$-1.74\% \pm 1.33\%$	$8.90\% \pm 5.03\%$	$1.82\% \pm 2.79\%$	$6.27\% \pm 4.08\%$	$8.72\% \pm 6.48\%$	$8.97\% \pm 6.56\%$

Uma das vantagens notáveis do aprendizado semi-supervisionado é que ele pode ser aplicado sem a necessidade de interação constante do usuário, o que é uma desvantagem do aprendizado ativo, que exige que o usuário forneça rótulos para os dados.

O estudo concluiu que o uso de métodos de aprendizado ativo e semi-supervisionado pode melhorar significativamente a classificação de atividades em dispositivos móveis, especialmente em contextos de monitoramento de saúde e fitness. As conclusões principais incluem:

- **Melhorias significativas com precisão inicial baixa:** Métodos como aprendizado ativo e "Democratic Co-learning" foram eficazes quando o classificador inicial tinha uma precisão baixa (75-80%), mas tiveram pouco impacto em classificadores com alta precisão (próxima de 90%).
- **Desempenho do "Democratic Co-learning":** Este método semi-supervisionado mostrou desempenho quase equivalente ao aprendizado ativo, mas sem a necessidade de interação do usuário, o que o torna mais viável para aplicações em dispositivos móveis, como os usados para monitoramento de saúde.
- **Possibilidade de melhorias sem riscos significativos:** A maioria dos métodos não causou queda significativa na precisão do classificador, o que sugere que eles podem ser implementados em sistemas reais sem risco substancial de degradação do desempenho.
- **Viabilidade para personalização:** A abordagem semi-supervisionada pode ser usada para adaptar os classificadores a cada usuário, melhorando a precisão à medida que o sistema aprende com os novos dados.

6) O estudo encontrado sobre aplicação da clusterização Fuzzy está no artigo: **The fuzzy clustering method: Applications in the air transport market in Taiwan.** Neste estudo é explorada a aplicação do método de clustering fuzzy no mercado de transporte aéreo, com foco na rota Taipei-Vancouver. O objetivo principal é identificar segmentos de mercado de passageiros e desenvolver estratégias de marketing personalizadas para companhias aéreas.

No contexto do transporte aéreo, os passageiros escolhem companhias com base em fatores diversos, como:

- Preço da passagem.
- Conveniência do horário e tempo de espera.
- Conforto e qualidade dos serviços a bordo.
- Programas de fidelidade.

Essas escolhas podem variar conforme o objetivo da viagem, condições socioeconômicas e preferências individuais. A aplicação de métodos de clustering tradicionais, como o método de Ward, falha em acomodar a possibilidade de que um mesmo passageiro pertença a múltiplos segmentos simultaneamente. Por exemplo, um passageiro pode preferir passagens baratas (segmento de preço), mas também valorizar conforto a bordo (segmento de qualidade).

Para resolver essa limitação, o artigo propõe o uso do **método de clustering fuzzy**, que permite atribuir graus de pertinência a diferentes clusters. Isso significa que os passageiros podem ser classificados em mais de um segmento de mercado com diferentes níveis de associação, proporcionando uma visão mais realista do comportamento dos consumidores.

A metodologia utilizada foi utilizar a técnica do Fuzzy Clustering (Fuzzy K-Means), no qual um modelo que calcula graus de pertinência para cada elemento em múltiplos clusters, possibilitando que um passageiro pertença a mais de um segmento. Com isso, a função objetivo minimizada considera a distância quadrada entre observações e centros de clusters, ponderada por graus de pertinência.

A parte de coleta de dados envolveu alguns questionários feitos para 2 grupos de passageiros (lazer e negócios) envolvendo questões como preço da passagem, conforto, segurança, entre outros. A amostra dos dados contou com 2560 questionários usados como base de dados.

A partir da aplicação foram percebidos 3 clusters principais e 2 mercados Fuzzy, sendo eles:

- **Mercado A:** Sensível a preço e tempo de trânsito.
- **Mercado B:** Focado em conforto no voo e programas de fidelidade.
- **Mercado C:** Priorização de pontualidade, segurança e eficiência de tempo.

Por fim os autores fazem uma breve comparação entre o clustering fuzzy e o tradicional ressaltando que o clustering fuzzy identificou sobreposições entre mercados, o que não é possível no método tradicional, por exemplo, um passageiro pode pertencer simultaneamente aos mercados A e B, dependendo de suas preferências por preço e conforto.

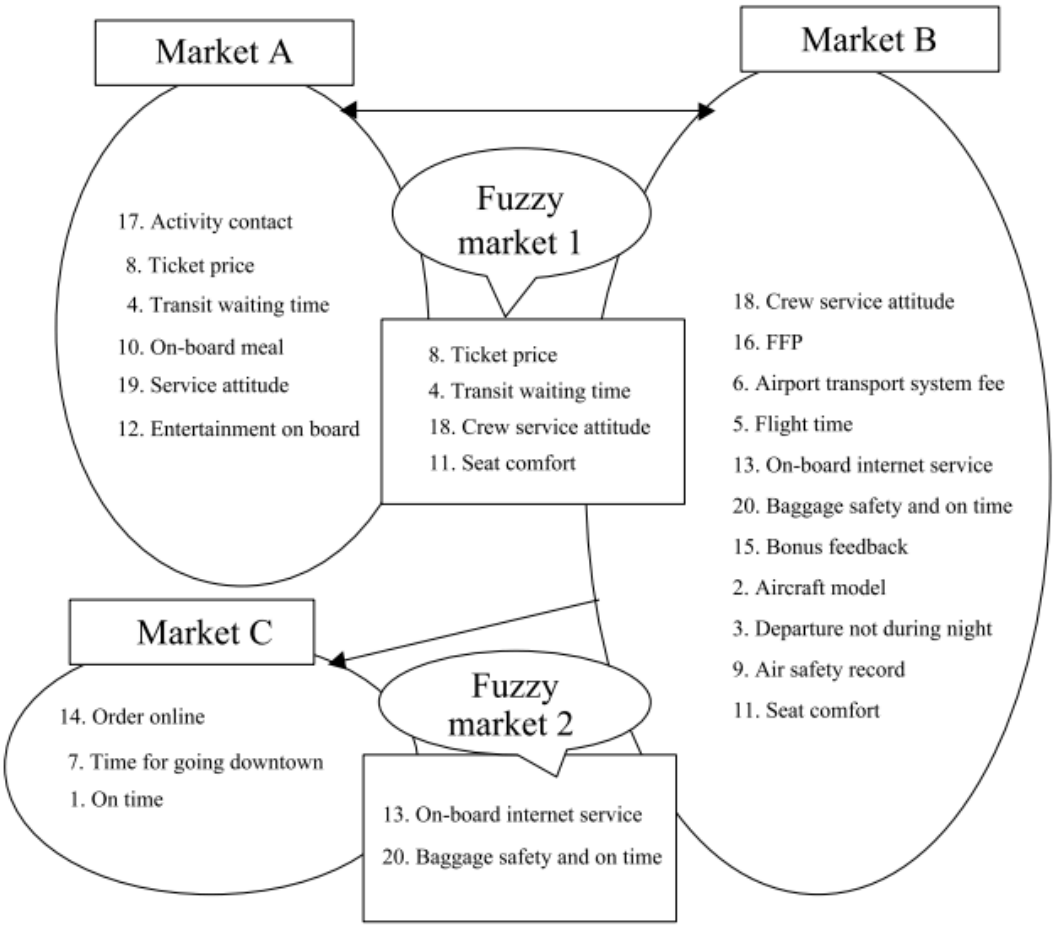

Figure 2: Fuzzy market segments for service qualities

Como mostrado na Figura 2, as variáveis favoráveis (qualidades de serviço) preferidas pelos viajantes nesta rota podem ser classificadas nos seguintes mercados:

- **Mercado A:** Serviço com valor. Se o viajante considera o preço do bilhete o fator mais importante, 36% dos viajantes também estão simultaneamente disponíveis para o mercado B; se o viajante prefere menor tempo de espera em conexões, 41% dos viajantes também estão simultaneamente disponíveis para o mercado B.
- **Mercado B:** Conforto da viagem e bônus. Viajantes podem pertencer simultaneamente ao mercado C — possivelmente 32% deles, se preferirem usar a internet a bordo; se preferirem segurança da bagagem e pontualidade no serviço, a possibilidade de estarem simultaneamente no mercado C também é de 32%.
- **Mercado C:** Efeito do tempo. A característica fuzzy deste segmento de mercado é pequena; portanto, ele não é distinguido na análise.

Em seguida, o cluster fuzzy divide este mercado em três grandes clusters, mas alguns viajantes pertencem a dois mercados. Como a Figura 2 mostra, entre o mercado A e o mercado B, existem quatro atributos que pertencem ao mercado fuzzy 1. No mercado A, com foco no preço do bilhete e no tempo de trânsito, os viajantes provavelmente também pertencem ao segmento de mercado B. No mercado B, com maior valor para a qualidade do

serviço dos atendentes a bordo e conforto moderado no voo, os viajantes simultaneamente pertencem ao mercado A. Nos mercados C e B, existem dois atributos no mercado fuzzy 2. Quando os viajantes pertencem ao mercado C e preferem uso de internet a bordo, segurança da bagagem e pontualidade no serviço, eles também pertencem ao mercado B.

Assim, a diferença entre o clustering fuzzy e o clustering tradicional está na rigidez tradicional da categorização de clusters em três segmentos de mercado. O clustering fuzzy consegue identificar, além dos três mercados principais, dois mercados fuzzy adicionais.

Códigos:

1 - a)

```
import numpy as np
import random

# Dados fornecidos (amostras)
data = np.array([
    [7.82, -4.58, -3.97],
    [-6.68, 3.16, 2.71],
    [4.36, 2.19, 2.09],
    [6.72, 0.88, 2.80],
    [-8.64, -3.06, 3.50],
    [-6.87, 0.57, -5.45],
    [4.47, -2.62, 5.76],
    [6.73, -2.01, 4.18],
    [-7.71, 2.34, -6.33],
    [-6.91, -0.49, -5.68],
    [6.18, -2.81, 5.82],
    [6.72, -0.93, -4.04],
    [-6.25, -0.26, 0.56],
    [-6.94, -1.22, 1.13],
    [8.09, 0.20, 2.25],
    [6.81, 0.17, -4.15],
    [-5.19, 4.24, 4.04],
```

```
    [-6.38, -1.74, 1.43],
    [4.08, 1.30, 5.33],
    [6.27, 0.93, -2.78]
])

# Selecionar 3 centros iniciais aleatórios
np.random.seed(42)
initial_indices = np.random.choice(range(len(data)), size=3, replace=False)
centroids = data[initial_indices]

print("Centros iniciais selecionados (índices):", initial_indices)
print("Centros iniciais (coordenadas):")
print(centroids)

def euclidean_distance(point, center):
    """Calcula a distância euclidiana entre um ponto e um centro."""
    return np.sqrt(np.sum((point - center) ** 2))

def kmeans(data, centroids, max_iter=100):
    """Algoritmo K-means para clusterização."""
    k = len(centroids)
    for iteration in range(max_iter):
        clusters = [[] for _ in range(k)]

        # Atribuir cada ponto ao cluster mais próximo
        for point in data:
            distances = [euclidean_distance(point, center) for center in centroids]
            cluster_idx = np.argmin(distances)
            clusters[cluster_idx].append(point)

        # Atualizar os centros como a média dos pontos em cada cluster
        new_centroids = []
        for cluster in clusters:
            if cluster:  # Evitar divisão por zero
                new_centroids.append(np.mean(cluster, axis=0))
            else:  # Se o cluster estiver vazio, manter o centro antigo
                new_centroids.append(centroids[len(new_centroids)])

        new_centroids = np.array(new_centroids)

        # Verificar se os centros mudaram
        if np.allclose(centroids, new_centroids, atol=1e-4):
            print(f"Convergência alcançada após {iteration + 1} iterações.")
            break

        centroids = new_centroids

    return centroids, clusters
```

```
# Executar o algoritmo K-means
final_centroids, final_clusters = kmeans(data, centroids)

# Exibir resultados
print("Centroides finais:")
print(final_centroids)
print("\nDistribuição dos pontos por cluster:")
for i, cluster in enumerate(final_clusters):
    print(f"Cluster {i + 1}: {np.array(cluster)}")
```

1 - b)

```
import numpy as np

# Dados fornecidos (amostras)
data = np.array([
    [7.82, -4.58, -3.97],
    [-6.68, 3.16, 2.71],
    [4.36, 2.19, 2.09],
    [6.72, 0.88, 2.80],
    [-8.64, -3.06, 3.50],
    [-6.87, 0.57, -5.45],
    [4.47, -2.62, 5.76],
    [6.73, -2.01, 4.18],
    [-7.71, 2.34, -6.33],
    [-6.91, -0.49, -5.68],
    [6.18, -2.81, 5.82],
    [6.72, -0.93, -4.04],
    [-6.25, -0.26, 0.56],
    [-6.94, -1.22, 1.13],
    [8.09, 0.20, 2.25],
    [6.81, 0.17, -4.15],
    [-5.19, 4.24, 4.04],
    [-6.38, -1.74, 1.43],
    [4.08, 1.30, 5.33],
    [6.27, 0.93, -2.78]
])

# Centros iniciais definidos no cenário b
centroids = np.array([
    [0, 0, 0],
    [0, -1, -1],
    [1, 1, 2]
])

def euclidean_distance(point, center):
```

```
    """Calcula a distância euclidiana entre um ponto e um centro."""
    return np.sqrt(np.sum((point - center) ** 2))

def kmeans(data, centroids, max_iter=100):
    """Algoritmo K-means para clusterização."""
    k = len(centroids)
    for iteration in range(max_iter):
        clusters = [[] for _ in range(k)]

        # Atribuir cada ponto ao cluster mais próximo
        for point in data:
            distances = [euclidean_distance(point, center) for center in centroids]
            cluster_idx = np.argmin(distances)
            clusters[cluster_idx].append(point)

        # Atualizar os centros como a média dos pontos em cada cluster
        new_centroids = []
        for cluster in clusters:
            if cluster:  # Evitar divisão por zero
                new_centroids.append(np.mean(cluster, axis=0))
            else:  # Se o cluster estiver vazio, manter o centro antigo
                new_centroids.append(centroids[len(new_centroids)])

        new_centroids = np.array(new_centroids)

        # Verificar se os centros mudaram
        if np.allclose(centroids, new_centroids, atol=1e-4):
            print(f"Convergência alcançada após {iteration + 1} iterações.")
            break

        centroids = new_centroids

    return centroids, clusters

# Executar o algoritmo K-means
final_centroids, final_clusters = kmeans(data, centroids)

# Exibir resultados
print("Centroides finais:")
print(final_centroids)
print("\nDistribuição dos pontos por cluster:")
for i, cluster in enumerate(final_clusters):
    print(f"Cluster {i + 1}: {np.array(cluster)}")
```

1 - c)

```
import numpy as np
```

```
# Dados fornecidos (amostras)
data = np.array([
    [7.82, -4.58, -3.97],
    [-6.68, 3.16, 2.71],
    [4.36, 2.19, 2.09],
    [6.72, 0.88, 2.80],
    [-8.64, -3.06, 3.50],
    [-6.87, 0.57, -5.45],
    [4.47, -2.62, 5.76],
    [6.73, -2.01, 4.18],
    [-7.71, 2.34, -6.33],
    [-6.91, -0.49, -5.68],
    [6.18, -2.81, 5.82],
    [6.72, -0.93, -4.04],
    [-6.25, -0.26, 0.56],
    [-6.94, -1.22, 1.13],
    [8.09, 0.20, 2.25],
    [6.81, 0.17, -4.15],
    [-5.19, 4.24, 4.04],
    [-6.38, -1.74, 1.43],
    [4.08, 1.30, 5.33],
    [6.27, 0.93, -2.78]
])

# Centros iniciais definidos no cenário c
centroids = np.array([
    [-0.1, 0, -0.1],
    [0, -2, -3],
    [1, -3, -5]
])

def euclidean_distance(point, center):
    """Calcula a distância euclidiana entre um ponto e um centro."""
    return np.sqrt(np.sum((point - center) ** 2))

def kmeans(data, centroids, max_iter=100):
    """Algoritmo K-means para clusterização."""
    k = len(centroids)
    for iteration in range(max_iter):
        clusters = [[] for _ in range(k)]

        # Atribuir cada ponto ao cluster mais próximo
        for point in data:
            distances = [euclidean_distance(point, center) for center in centroids]
            cluster_idx = np.argmin(distances)
            clusters[cluster_idx].append(point)
```

```
        # Atualizar os centros como a média dos pontos em cada cluster
        new_centroids = []
        for cluster in clusters:
            if cluster:  # Evitar divisão por zero
                new_centroids.append(np.mean(cluster, axis=0))
            else:  # Se o cluster estiver vazio, manter o centro antigo
                new_centroids.append(centroids[len(new_centroids)])

        new_centroids = np.array(new_centroids)

        # Verificar se os centros mudaram
        if np.allclose(centroids, new_centroids, atol=1e-4):
            print(f"Convergência alcançada após {iteration + 1} iterações.")
            break

        centroids = new_centroids

    return centroids, clusters

# Executar o algoritmo K-means
final_centroids, final_clusters = kmeans(data, centroids)

# Exibir resultados
print("Centroides finais:")
print(final_centroids)
print("\nDistribuição dos pontos por cluster:")
for i, cluster in enumerate(final_clusters):
    print(f"Cluster {i + 1}: {np.array(cluster)}")



2)
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from minisom import MiniSom

m1 = np.array([0, 0, 0, 0, 0, 0, 0, 0])
m2 = np.array([4, 0, 0, 0, 0, 0, 0, 0])
m3 = np.array([0, 0, 0, 4, 0, 0, 0, 0])
m4 = np.array([0, 0, 0, 0, 0, 0, 0, 4])

def generate_gaussian_data(mean, size=100):
    return np.random.normal(loc=mean, scale=1.0, size=(size, len(mean)))

data1 = generate_gaussian_data(m1)
data2 = generate_gaussian_data(m2)
data3 = generate_gaussian_data(m3)
```

```
data4 = generate_gaussian_data(m4)

data = np.vstack((data1, data2, data3, data4))

plt.figure(figsize=(12, 8))

plt.subplot(2, 2, 1)
plt.scatter(data1[:, 0], data1[:, 1], alpha=0.5)
plt.title('Distribuição Gaussiana - Centróide m1')

plt.subplot(2, 2, 2)
plt.scatter(data2[:, 0], data2[:, 1], alpha=0.5)
plt.title('Distribuição Gaussiana - Centróide m2')

plt.subplot(2, 2, 3)
plt.scatter(data3[:, 0], data3[:, 1], alpha=0.5)
plt.title('Distribuição Gaussiana - Centróide m3')

plt.subplot(2, 2, 4)
plt.scatter(data4[:, 0], data4[:, 1], alpha=0.5)
plt.title('Distribuição Gaussiana - Centróide m4')

plt.tight_layout()
plt.show()

scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)

som = MiniSom(10, 10, data_scaled.shape[1], sigma=1.0, learning_rate=0.5)
som.random_weights_init(data_scaled)
som.train_random(data_scaled, 10000)

plt.figure(figsize=(10, 10))
for i, x in enumerate(data_scaled):
    w = som.winner(x)
    plt.text(w[0] + .5, w[1] + .5, str(i), ha='center', va='center', bbox=dict(facecolor='white',
alpha=0.5, lw=0))

plt.imshow(som.distance_map().T, cmap='coolwarm', interpolation='none')
plt.colorbar()
plt.title('Mapa Auto-Organizável (SOM)')
plt.show()

3)
import pandas as pd
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt
```

```
from minisom import MiniSom

# Dados
data = {
    'Países': ['Reino Unido', 'Austrália', 'Canadá', 'Estados Unidos', 'Japão', 'França',
'Cingapura', 'Argentina', 'Uruguai', 'Cuba', 'Colômbia', 'Brasil', 'Paraguai', 'Egito', 'Nigéria',
'Senegal', 'Serra Leoa', 'Angola', 'Etiópia', 'Moçambique', 'China'],
    'Expectativa de Vida': [0.88, 0.90, 0.90, 0.87, 0.89, 0.89, 0.88, 0.81, 0.82, 0.75, 0.77, 0.80,
0.70, 0.70, 0.44, 0.47, 0.23, 0.34, 0.31, 0.24, 0.80],
    'Educação': [0.99, 0.99, 0.98, 0.98, 0.97, 0.97, 0.87, 0.92, 0.92, 0.94, 0.75, 0.73, 0.70,
0.62, 0.58, 0.37, 0.33, 0.36, 0.35, 0.37, 0.98],
    'PIB': [0.91, 0.93, 0.94, 0.97, 0.90, 0.85, 0.83, 0.70, 0.65, 0.40, 0.65, 0.82, 0.60, 0.61,
0.55, 0.45, 0.27, 0.51, 0.32, 0.20, 0.95],
    'Estabilidade Política': [1.10, 1.26, 1.24, 1.18, 1.04, 1.10, 1.41, 0.55, 0.70, 0.07, 0.50,
0.70, 0.40, 0.21, -1.36, -0.68, -1.56, -1.98, -2.0, -3.0, 1.09]
}

df = pd.DataFrame(data)

#normalização
scaler = StandardScaler()
data_scaled = scaler.fit_transform(df.iloc[:, 1:])

#clusterização hierárquica
linked = linkage(data_scaled, method='ward')
dendrogram(linked, labels=df['Países'].to_list(), leaf_rotation=90)
plt.title('Dendrograma')
plt.show()

#mapas Auto-Organizáveis (SOM)
som = MiniSom(15, 15, data_scaled.shape[1], sigma=1.0, learning_rate=0.5)
som.random_weights_init(data_scaled)
som.train_random(data_scaled, 50000)

# visualização do SOM
plt.figure(figsize=(7, 7))
for i, x in enumerate(data_scaled):
    w = som.winner(x)
    plt.text(w[0] + .5, w[1] + .5, df['Países'][i], ha='center', va='center',
bbox=dict(facecolor='white', alpha=0.5, lw=0))

#adicionando a visualização de distância
plt.imshow(som.distance_map().T, cmap='coolwarm', interpolation='none')
plt.colorbar()

plt.title('Mapa Auto-Organizável (SOM)')
plt.show()
```

Referências

CHIANG, Ian Wen-Yu; LIANG, Gin-Shuh; YAHALOM, Shmuel Z. **The fuzzy clustering method: applications in the air transport market in Taiwan.** *Database Marketing & Customer Strategy Management*, Henry Stewart Publications, v. 11, n. 2, p. 149–158, 2003. Disponível em: <https://link.springer.com/content/pdf/10.1057/palgrave.dbm.3240215.pdf>. Acesso em: 20 jan. 2025.

LONGSTAFF, Brent; REDDY, Sasank; ESTRIN, Deborah. **Improving Activity Classification for Health Applications on Mobile Devices using Active and Semi-Supervised Learning**. *Center for Embedded Networked Sensing*, University of California Los Angeles. Disponível em: http://www.ucla.edu. Acesso em: 20 jan. 2025.

MACHADO, Lucca; BORBA, Anderson**. Redução da dimensionalidade em Big Data**. 2023, Disponível em: https://adelpha-api.mackenzie.br/server/api/core/bitstreams/3abb8117-f5bd-4a50-9607-fa07d96df851/content. Acesso em: 20 jan. 2025.