

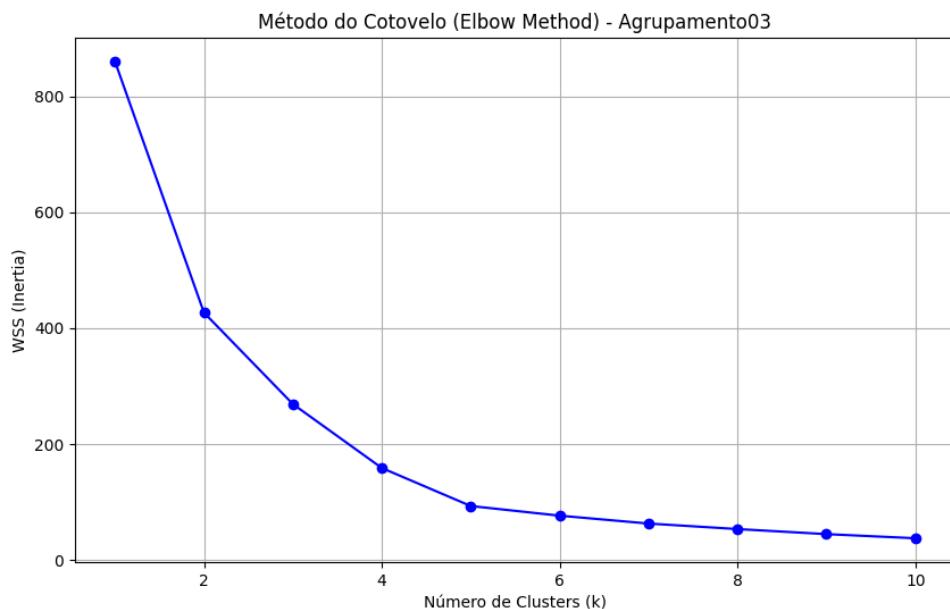
Segundo Trabalho Prático sobre Agrupamento

Aluno: Pedro Henrique Vilaça Valverde

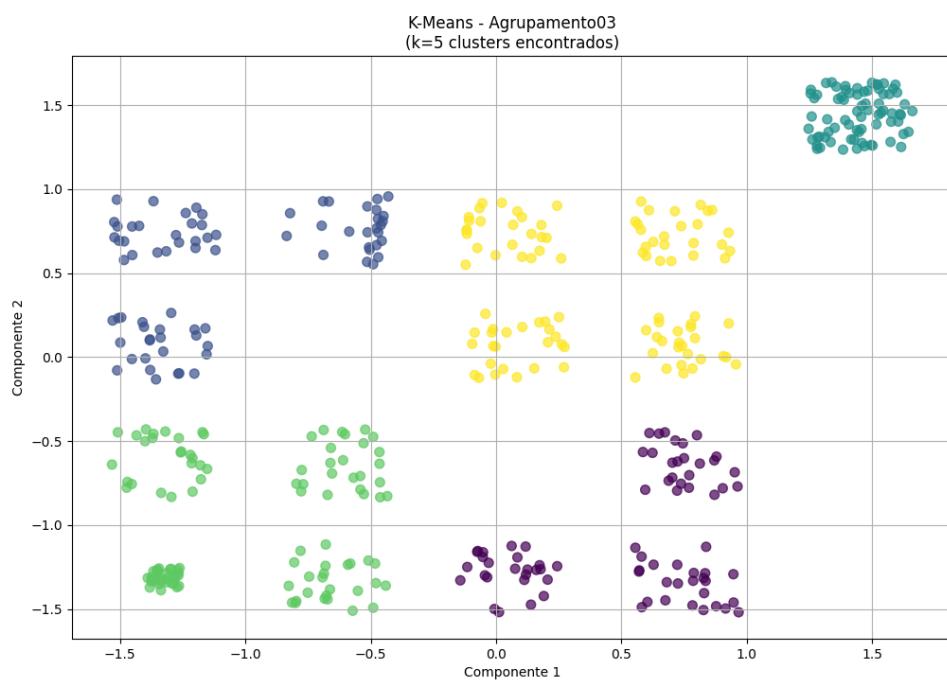
Disciplina: Inteligência Artificial

Problema 1: Agrupamento03.txt

Metodo do Cotovelo (Elbow Method)

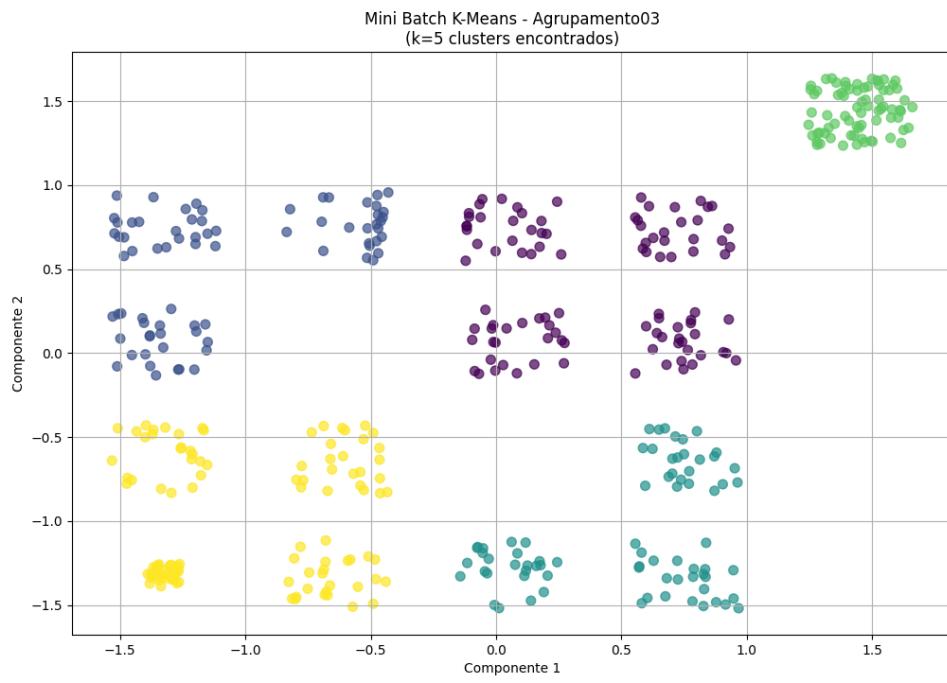


1.1) K-Means

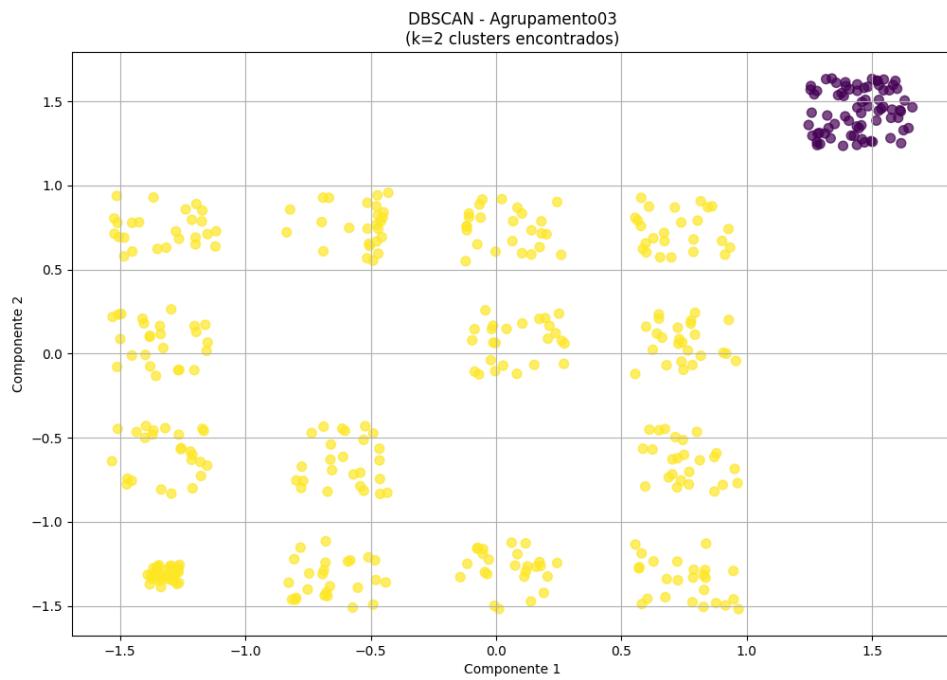


1.2) Mini Batch K-Means

Segundo Trabalho Prático sobre Agrupamento

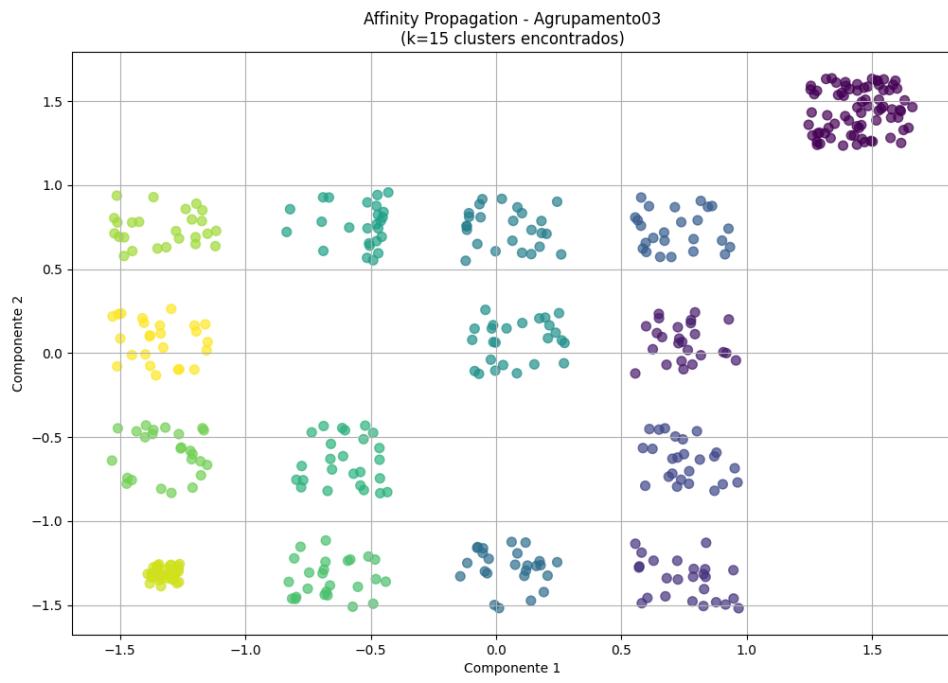


1.3) DBSCAN

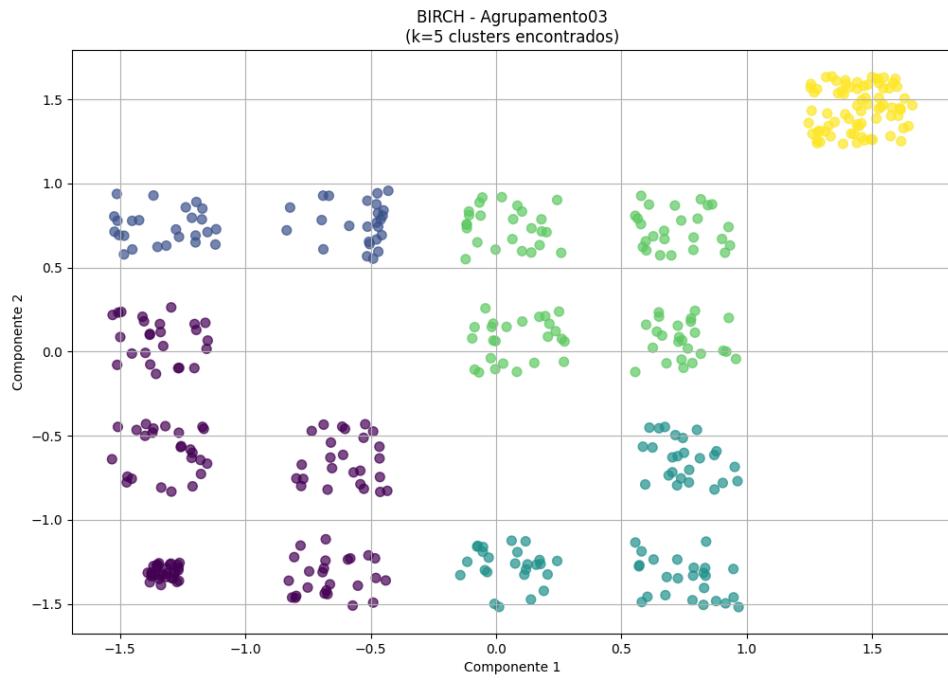


1.4) Affinity Propagation

Segundo Trabalho Prático sobre Agrupamento



1.5) BIRCH

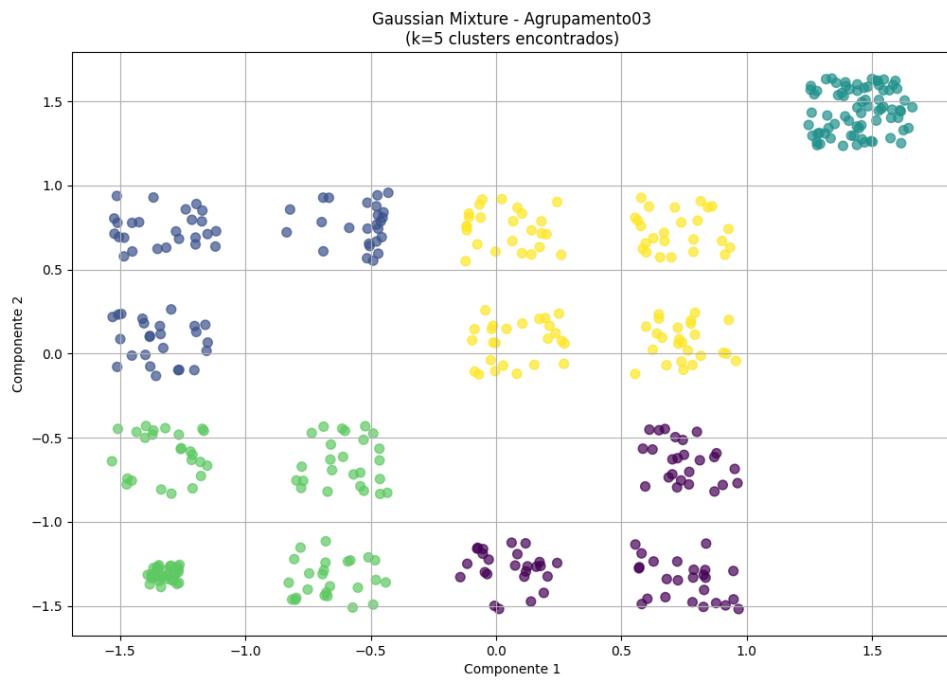


1.6) Agglomerative Clustering

Segundo Trabalho Prático sobre Agrupamento

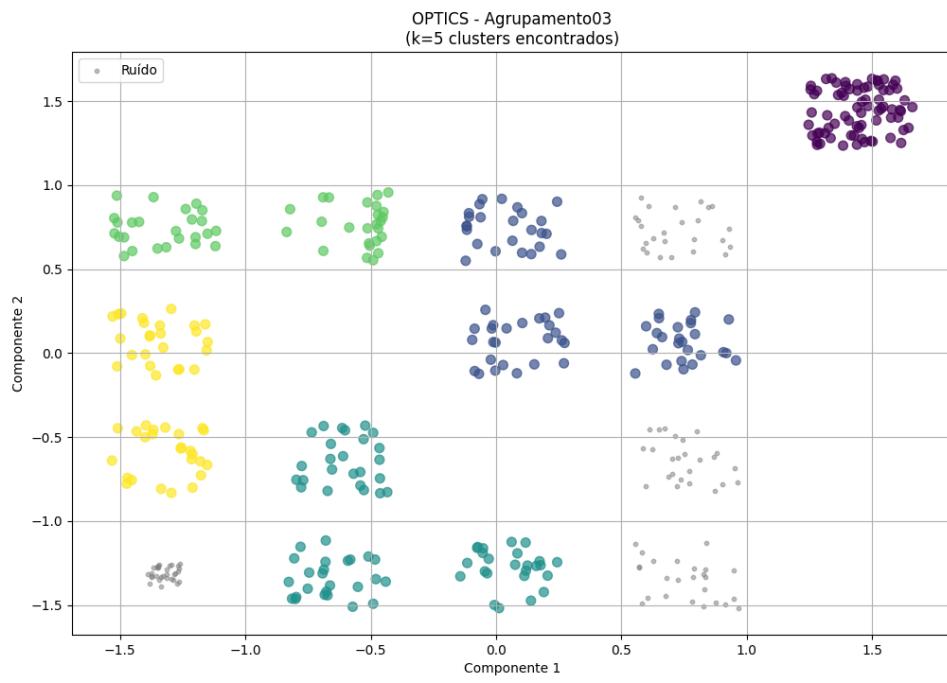


1.7) Gaussian Mixture

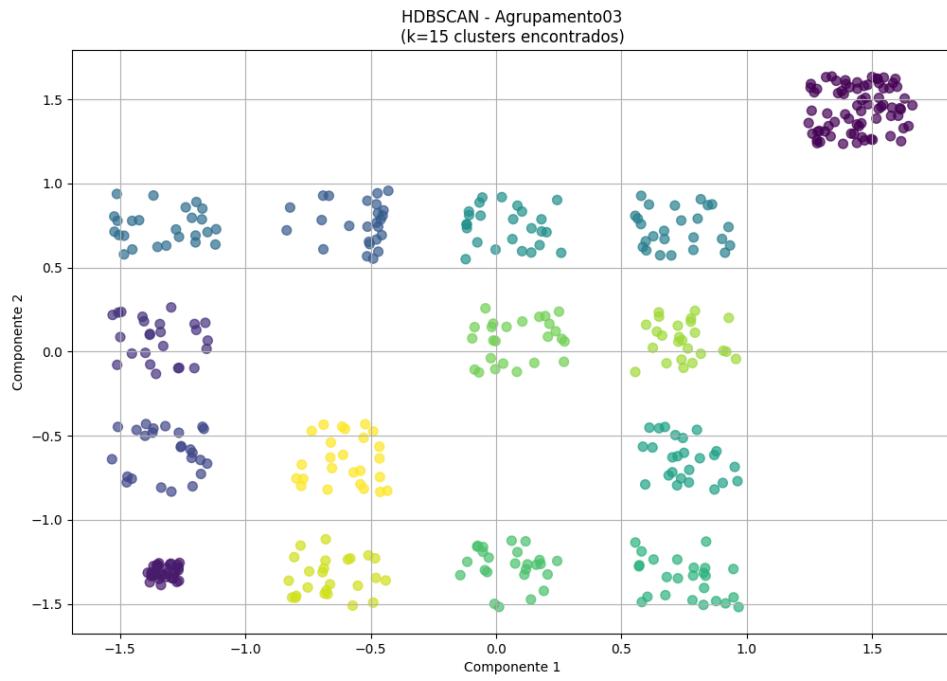


1.8) OPTICS

Segundo Trabalho Prático sobre Agrupamento

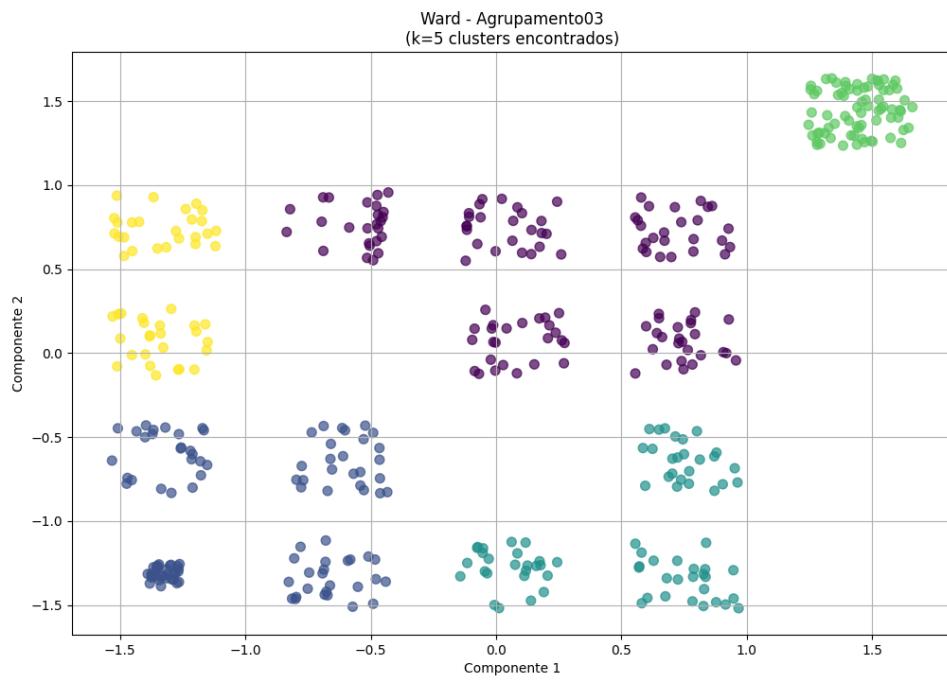


1.9) HDBSCAN

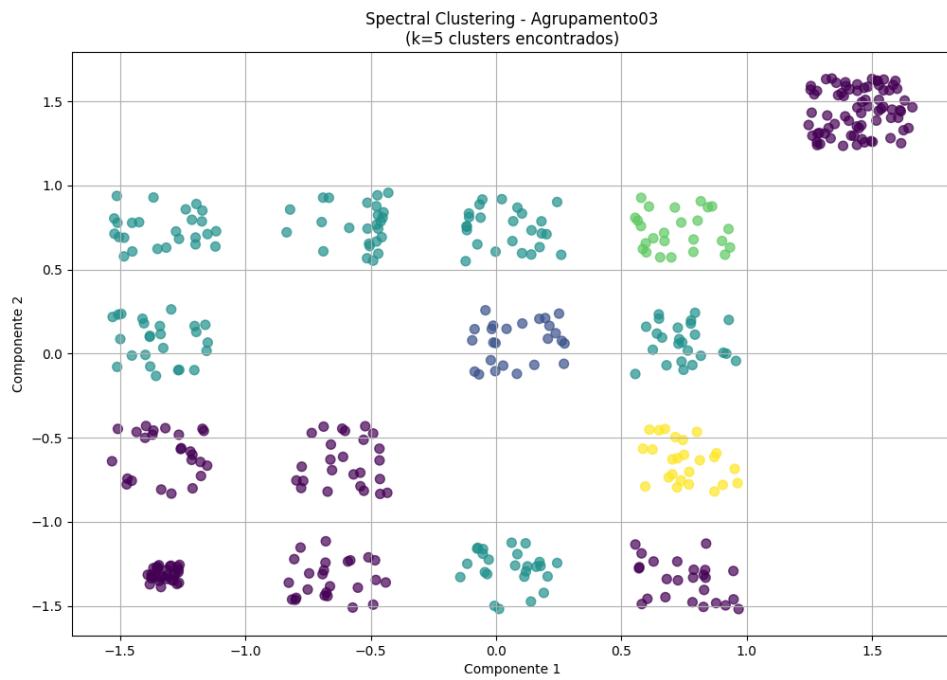


1.10) Ward

Segundo Trabalho Prático sobre Agrupamento

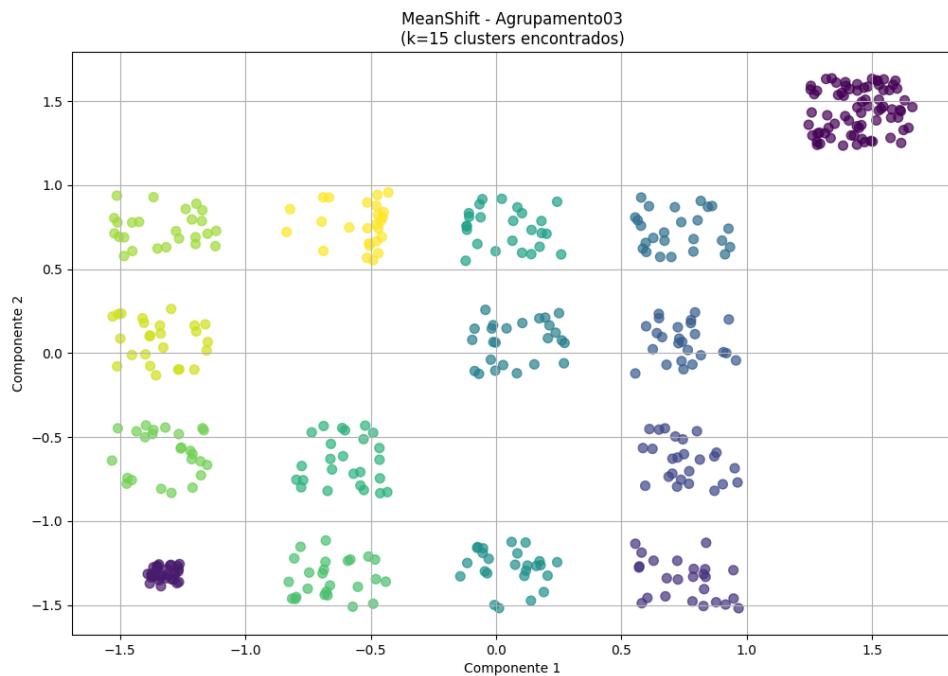


1.11) Spectral Clustering



1.12) MeanShift

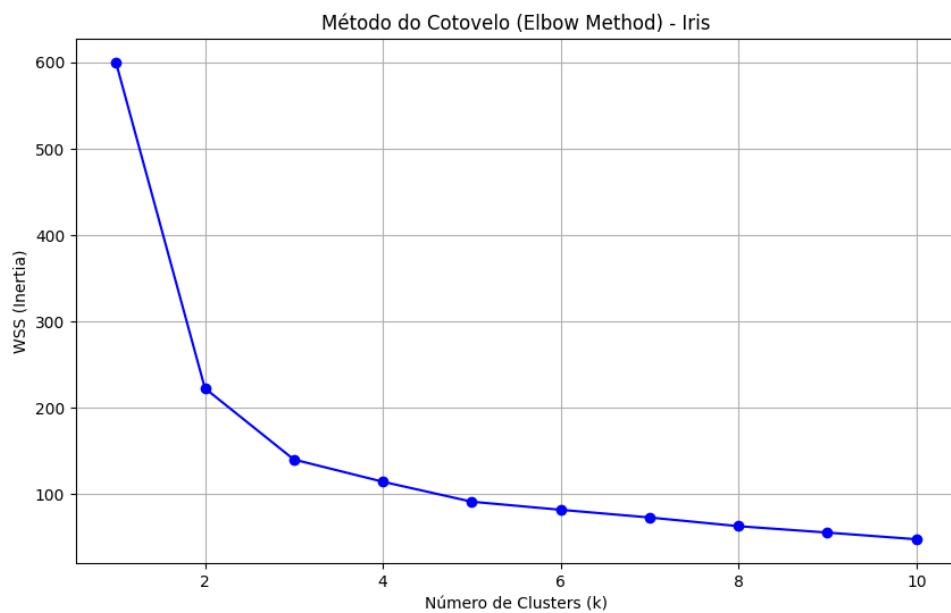
Segundo Trabalho Prático sobre Agrupamento



Segundo Trabalho Prático sobre Agrupamento

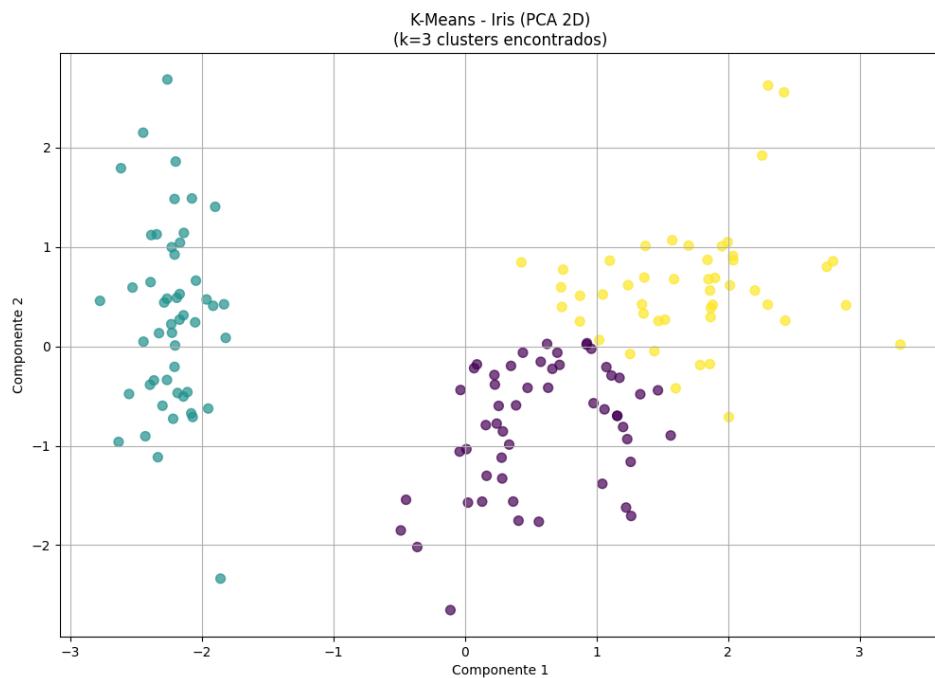
Problema 2: iris_cluster.txt

Metodo do Cotovelo (Elbow Method)



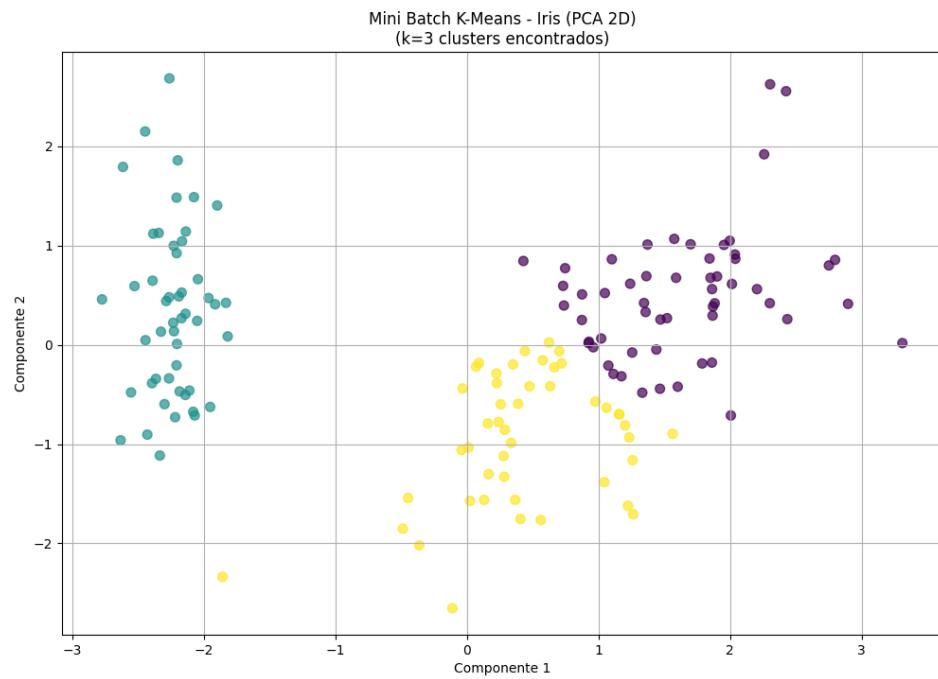
Nota: Os gráficos a seguir são plotados usando os 2 primeiros Componentes Principais (PCA) para visualização 2D.

2.1) K-Means

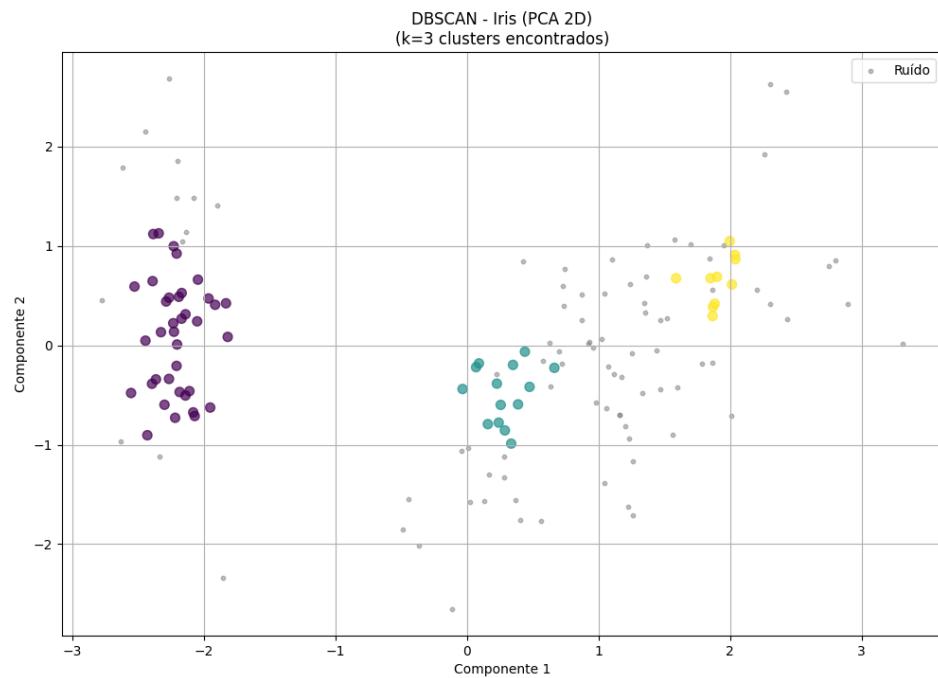


2.2) Mini Batch K-Means

Segundo Trabalho Prático sobre Agrupamento

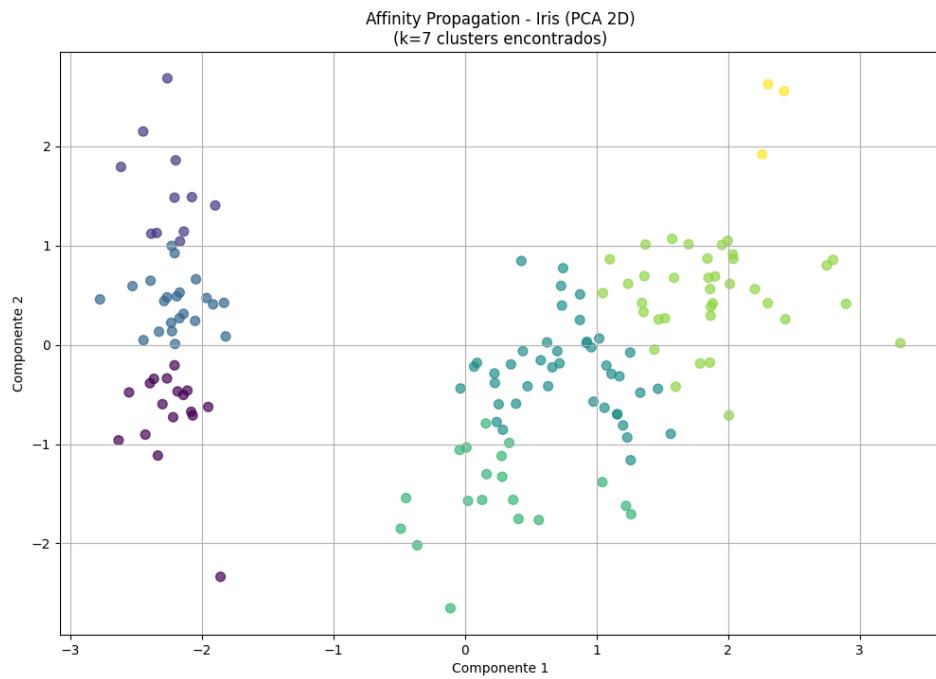


2.3) DBSCAN

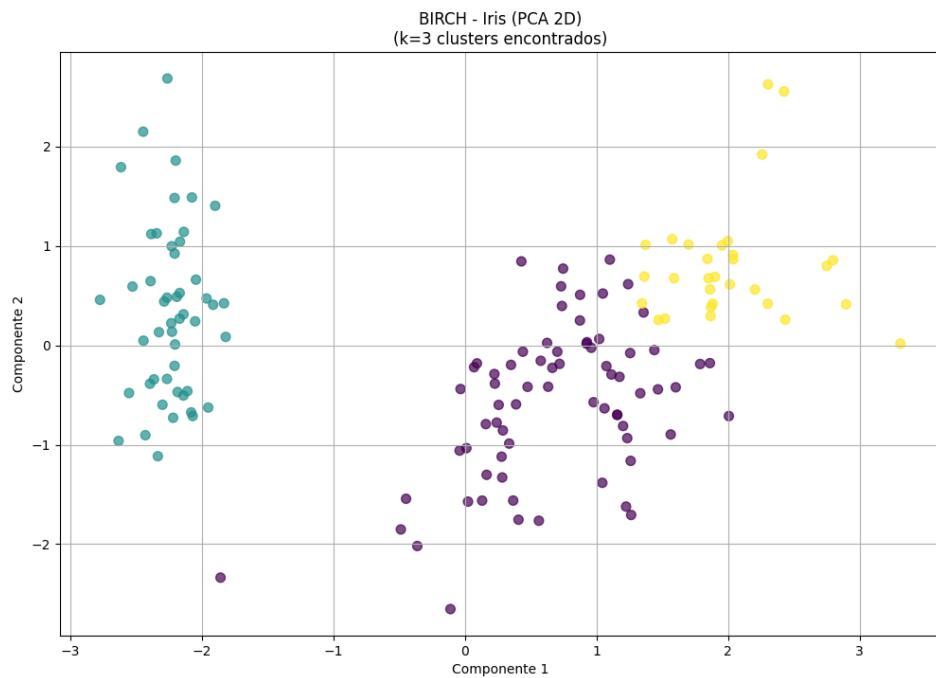


2.4) Affinity Propagation

Segundo Trabalho Prático sobre Agrupamento

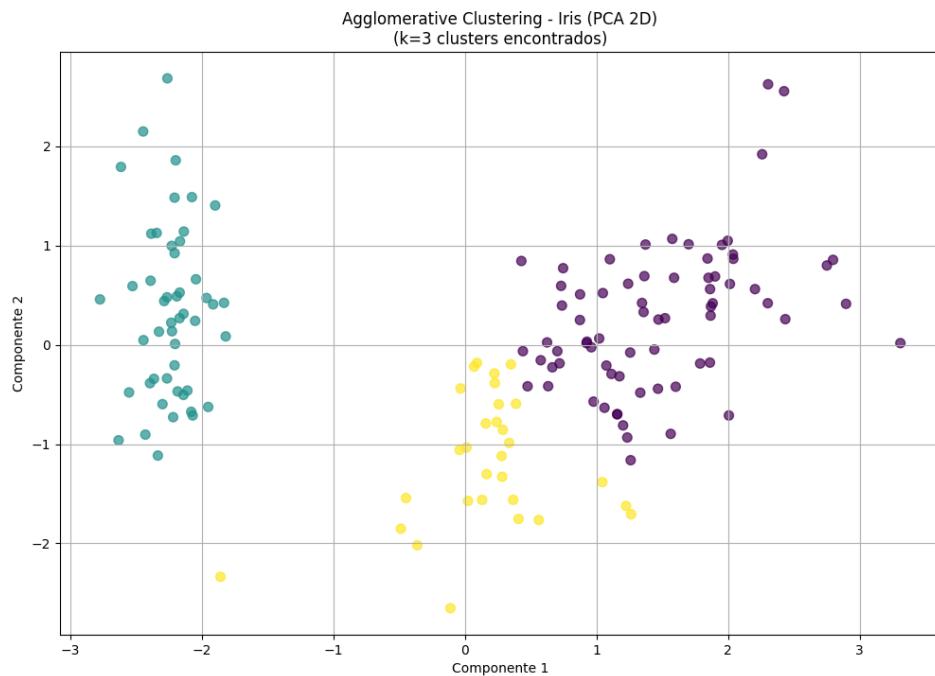


2.5) BIRCH

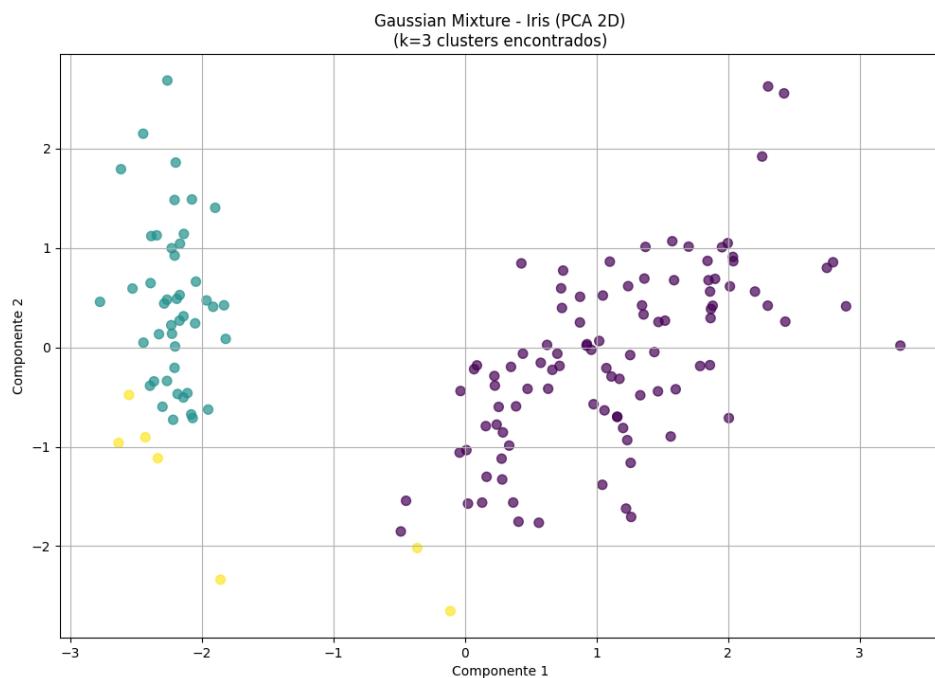


2.6) Agglomerative Clustering

Segundo Trabalho Prático sobre Agrupamento

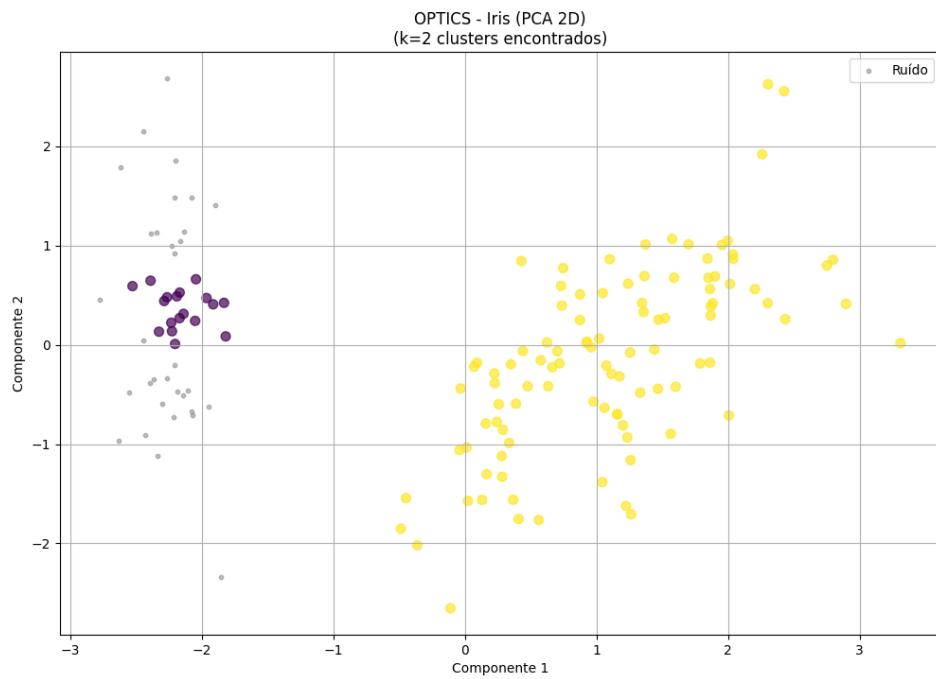


2.7) Gaussian Mixture

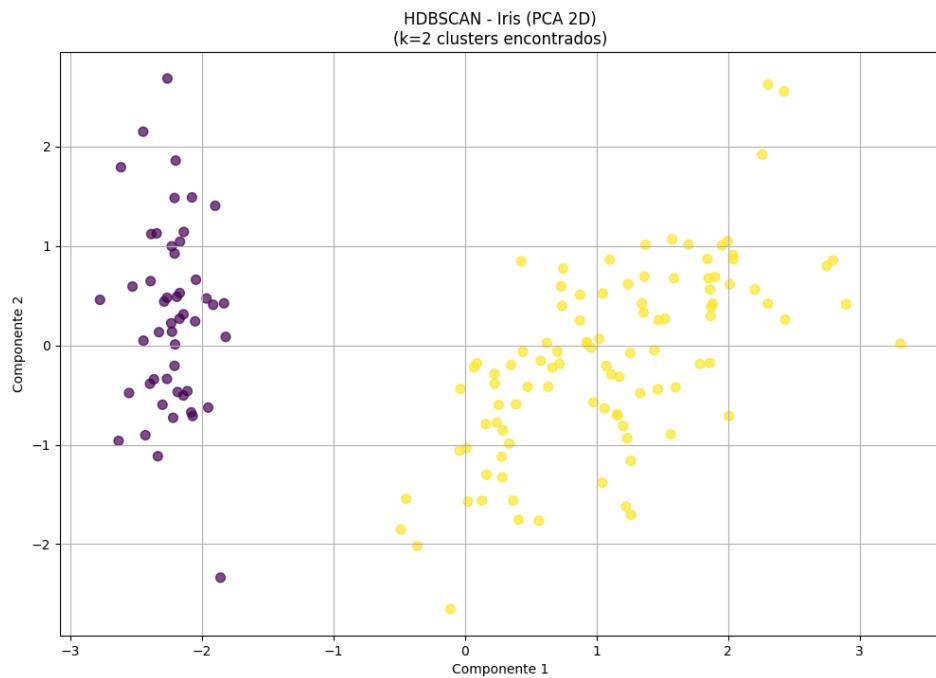


2.8) OPTICS

Segundo Trabalho Prático sobre Agrupamento

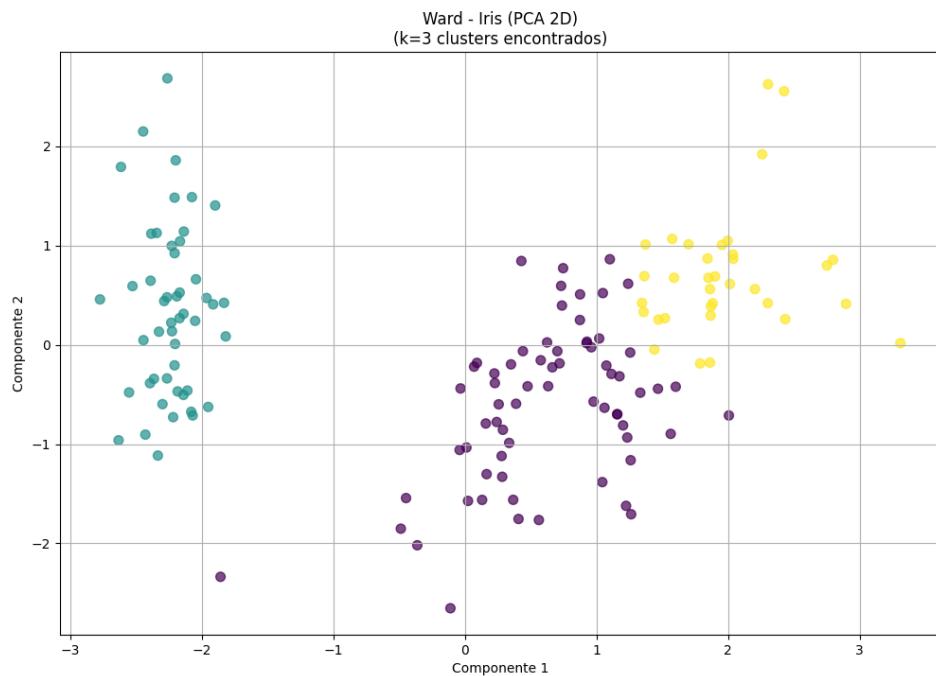


2.9) HDBSCAN

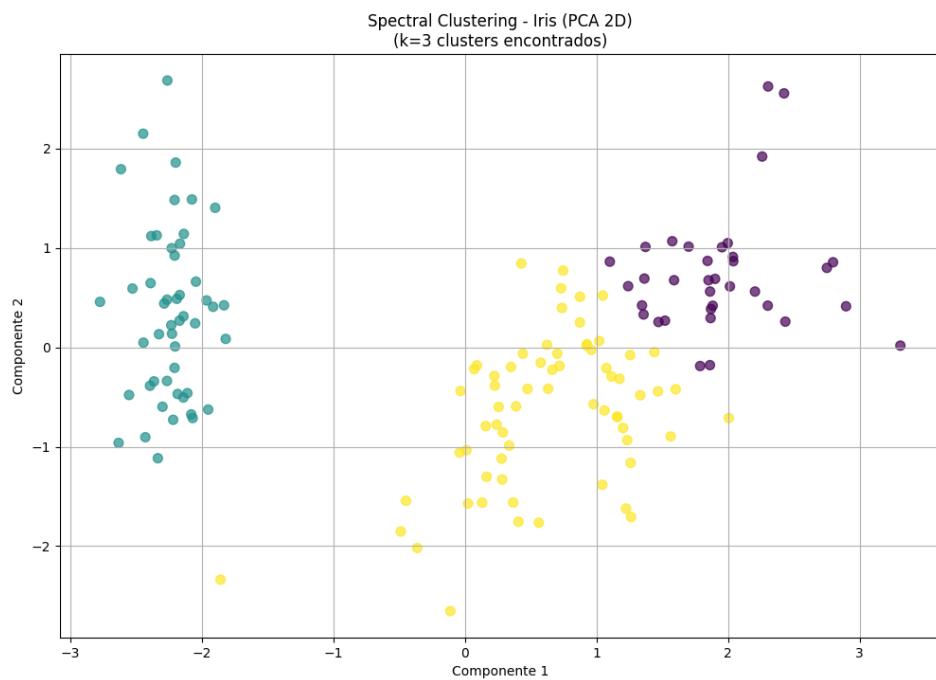


2.10) Ward

Segundo Trabalho Prático sobre Agrupamento

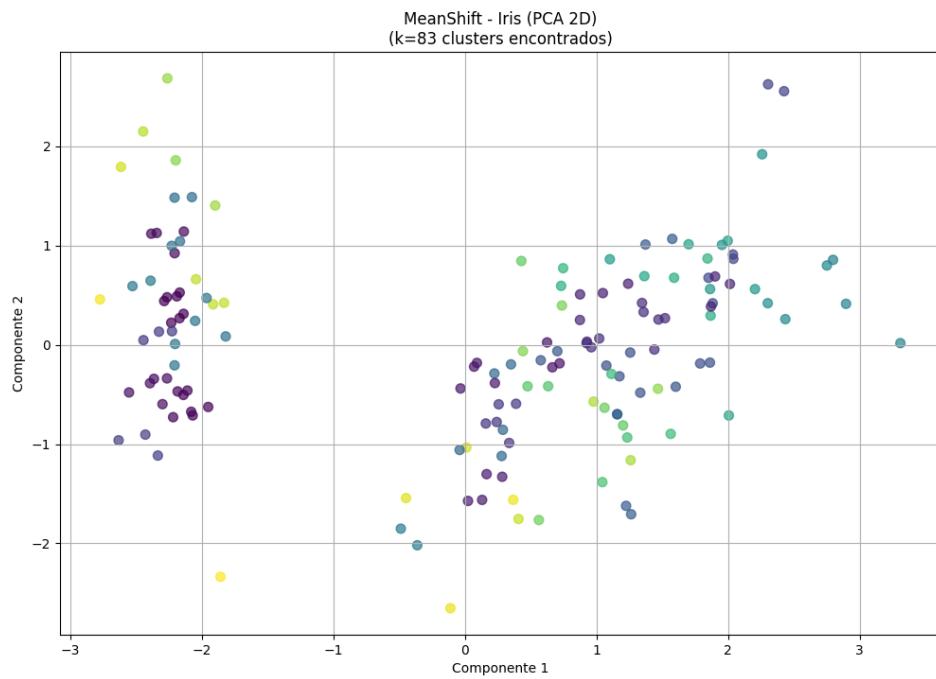


2.11) Spectral Clustering



2.12) MeanShift

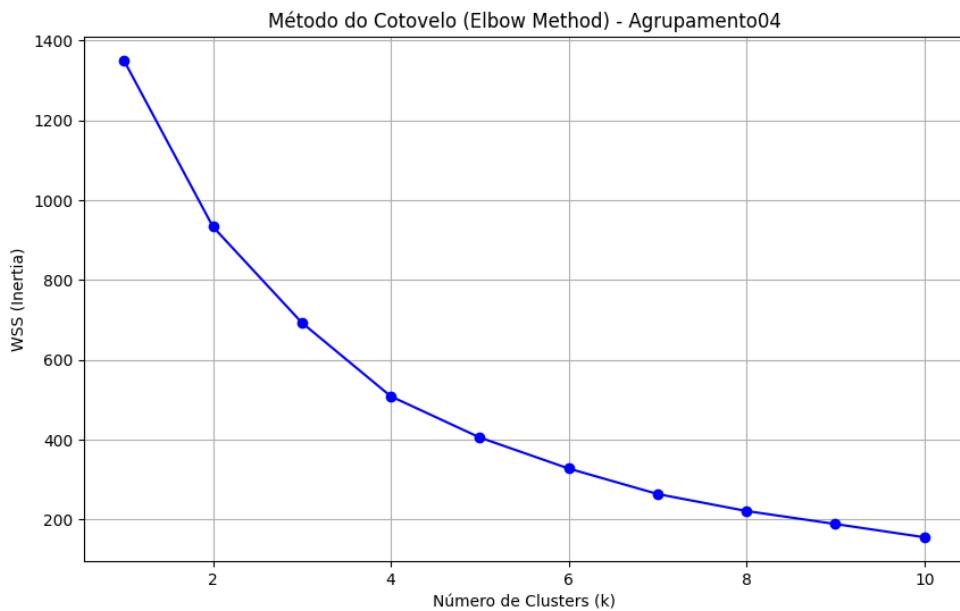
Segundo Trabalho Prático sobre Agrupamento



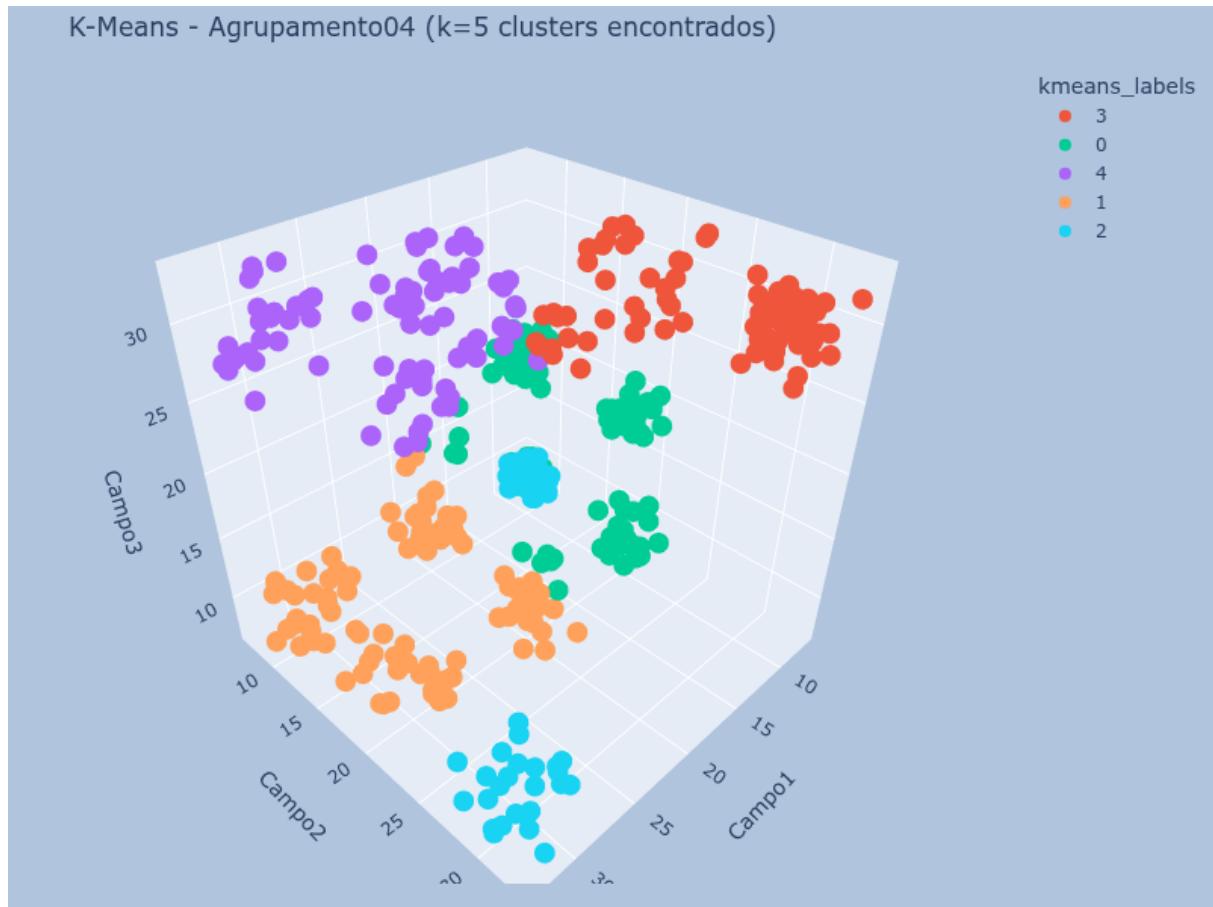
Segundo Trabalho Prático sobre Agrupamento

Problema 3: Agrupamento04.txt

Metodo do Cotovelo (Elbow Method)



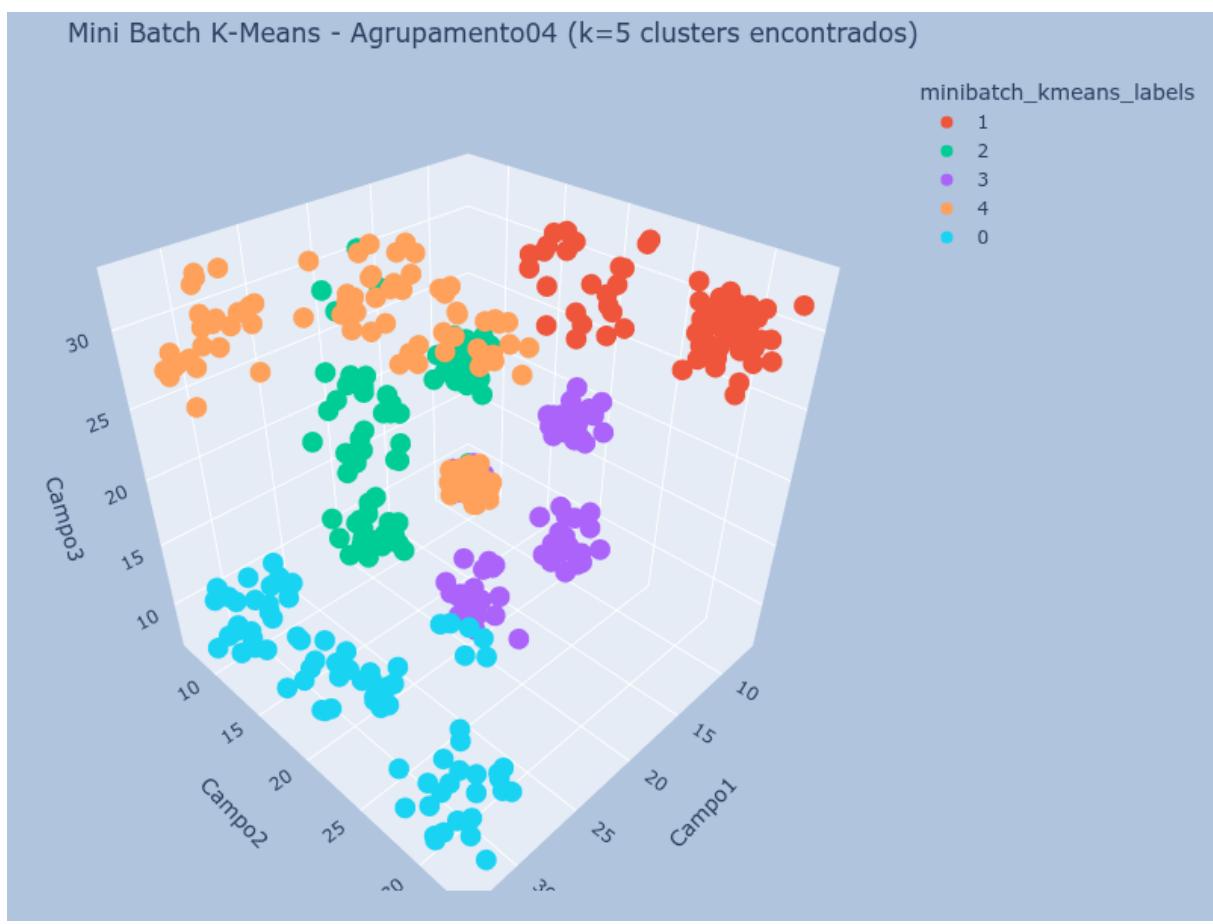
3.1) K-Means



[Abrir versão interativa \(p3_kmeans.html\)](#)

Segundo Trabalho Prático sobre Agrupamento

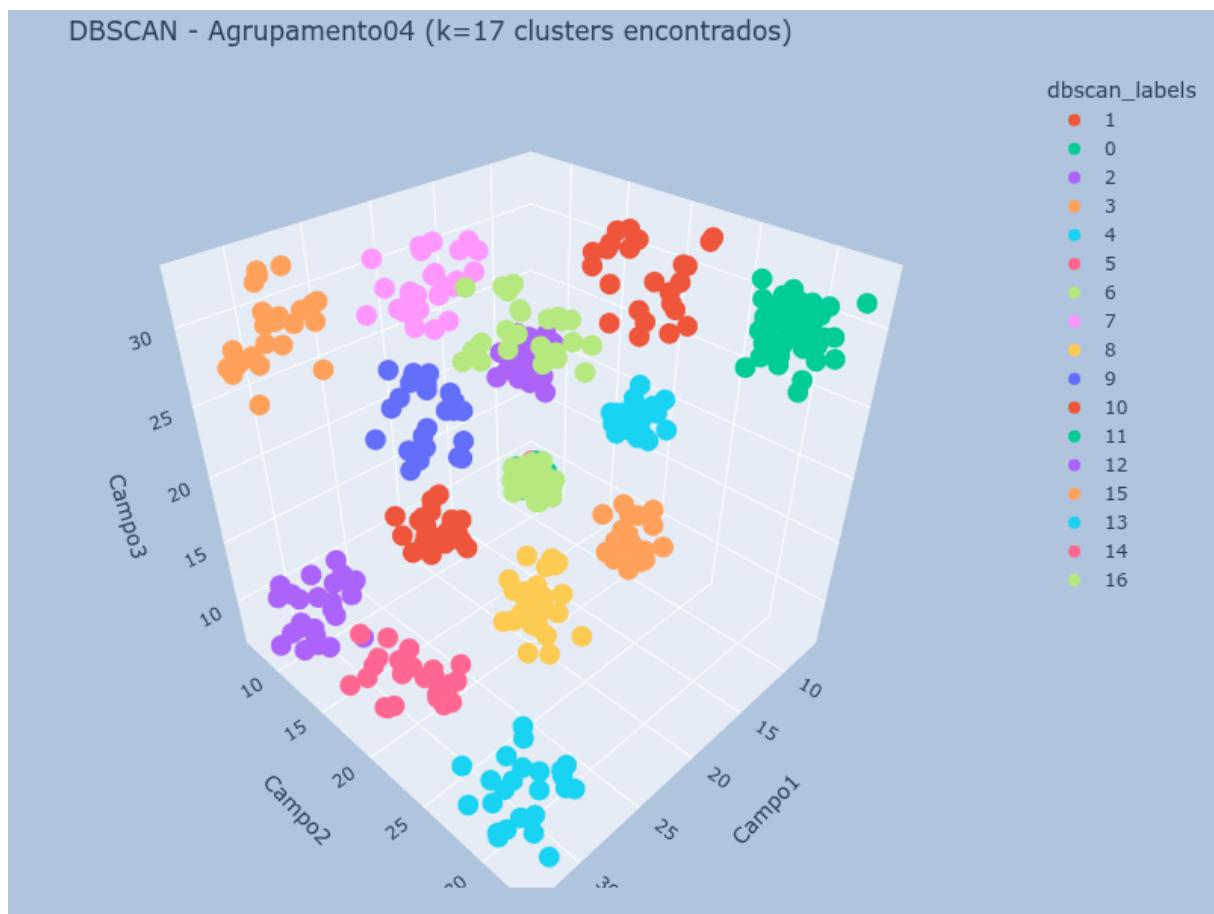
3.2) Mini Batch K-Means



[Abrir versão interativa \(p3_minibatch_kmeans.html\)](#)

3.3) DBSCAN

Segundo Trabalho Prático sobre Agrupamento

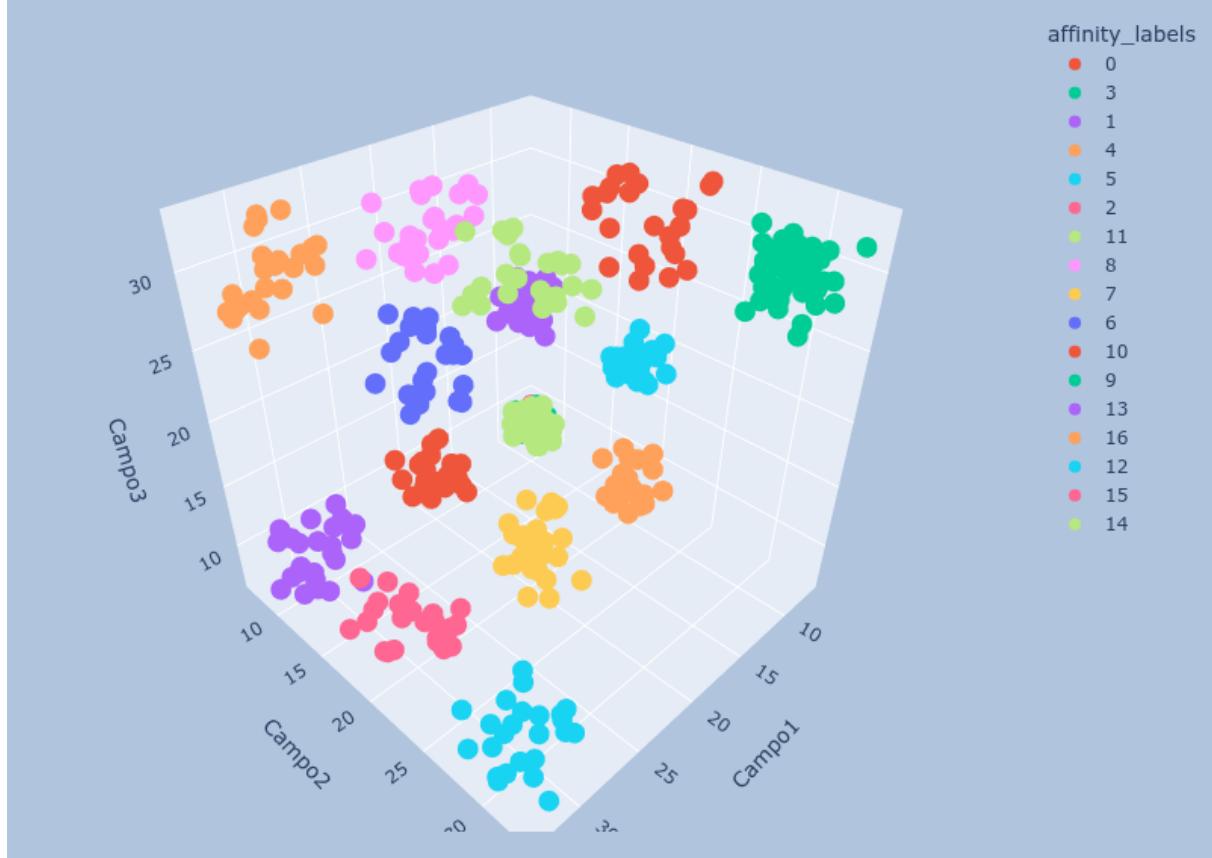


[Abrir versão interativa \(p3_dbSCAN.html\)](#)

3.4) Affinity Propagation

Segundo Trabalho Prático sobre Agrupamento

Affinity Propagation - Agrupamento04 (k=17 clusters encontrados)

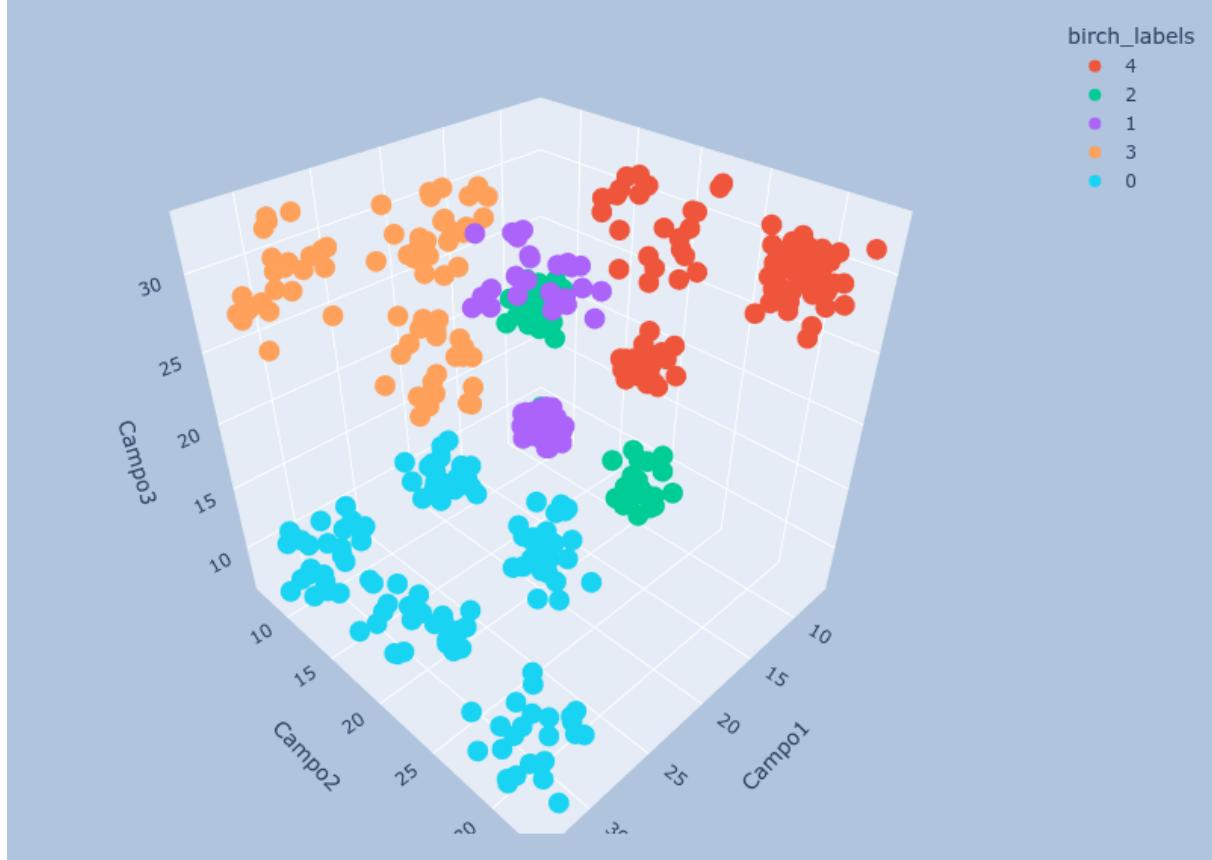


[Abrir versão interativa \(p3_affinity.html\)](#)

3.5) BIRCH

Segundo Trabalho Prático sobre Agrupamento

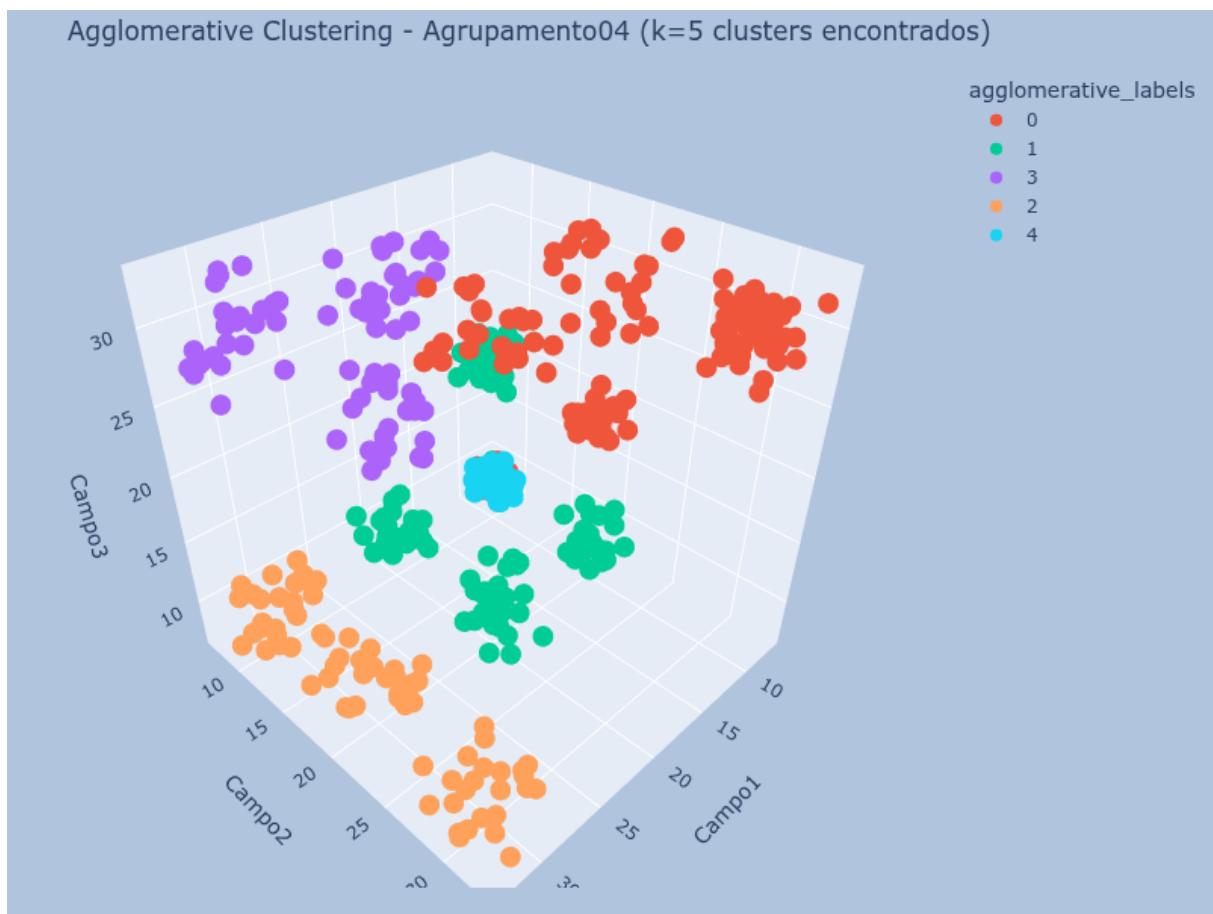
BIRCH - Agrupamento04 (k=5 clusters encontrados)



[Abrir versão interativa \(p3_birch.html\)](#)

3.6) Agglomerative Clustering

Segundo Trabalho Prático sobre Agrupamento

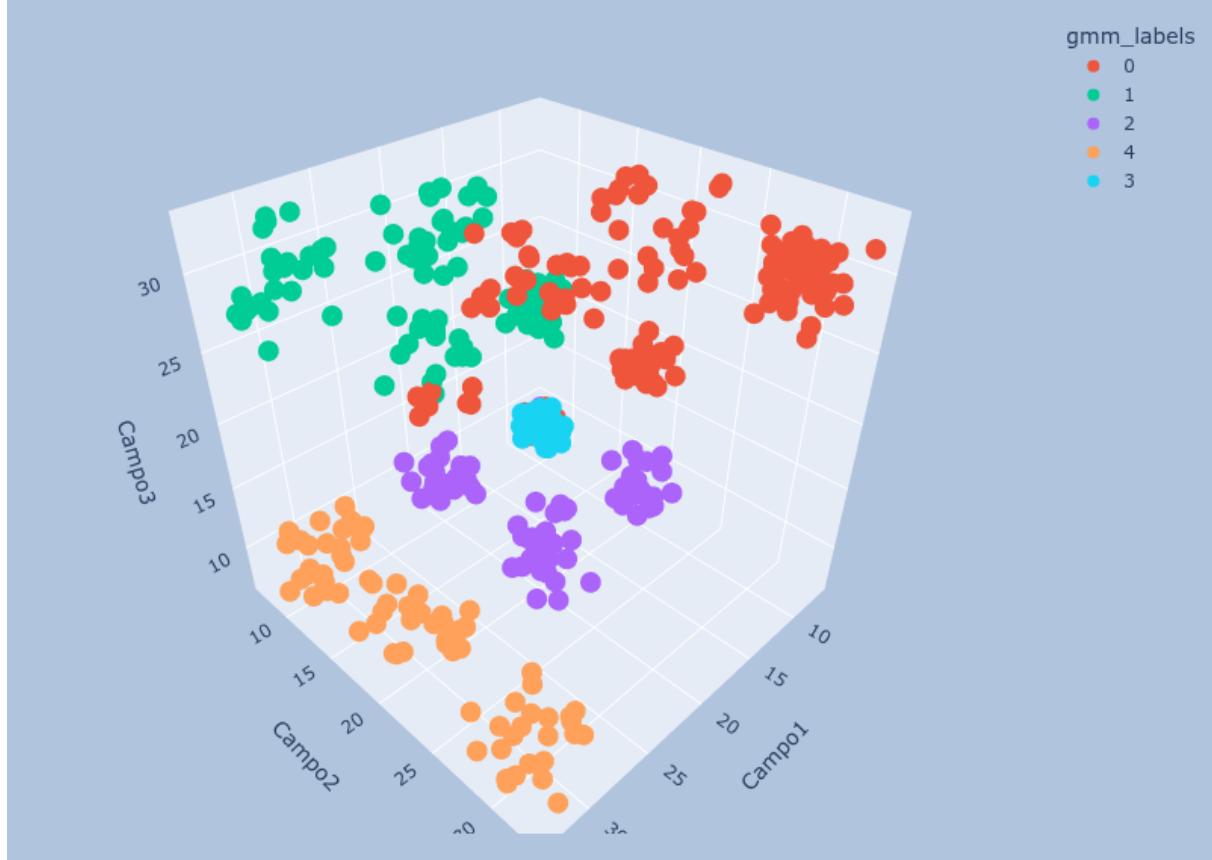


[Abrir versão interativa \(p3_agglomerative.html\)](#)

3.7) Gaussian Mixture

Segundo Trabalho Prático sobre Agrupamento

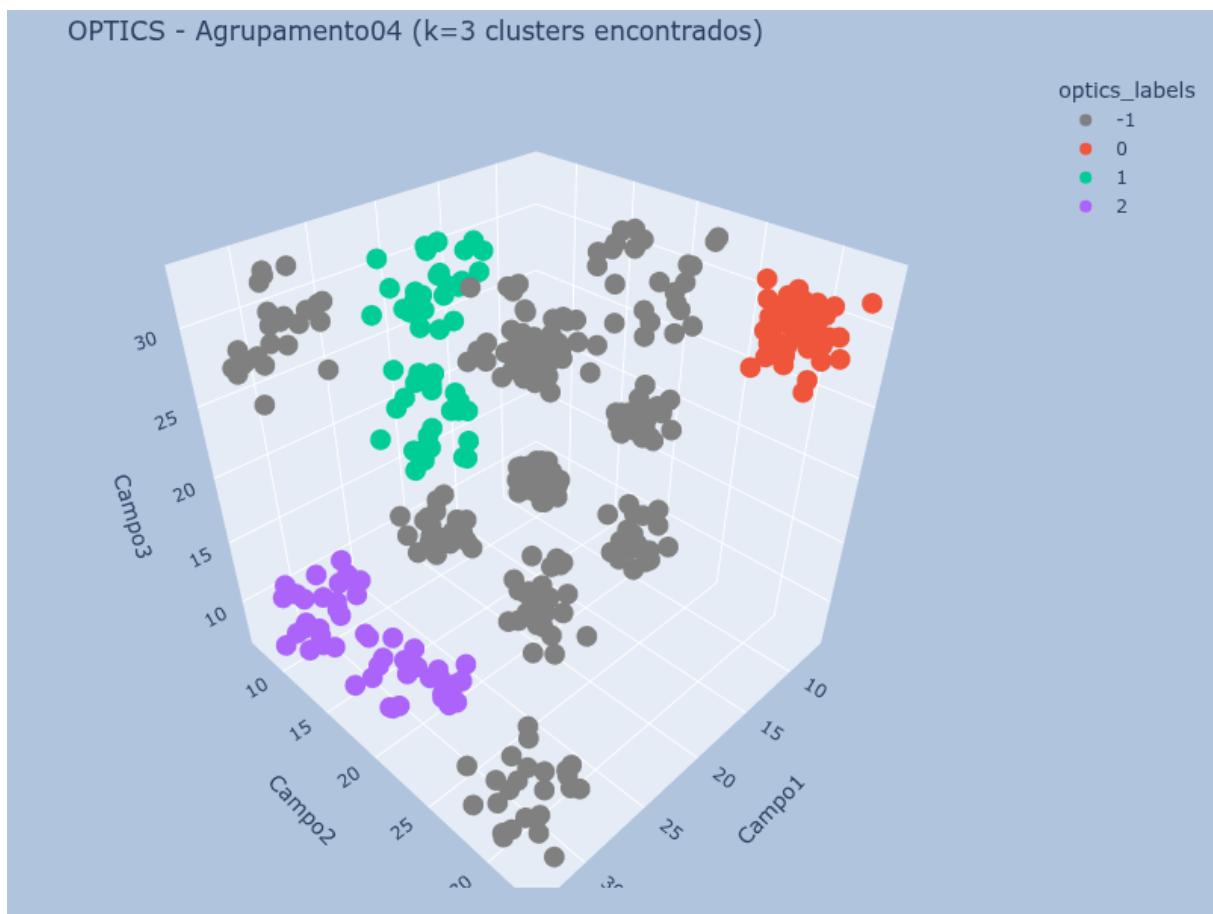
Gaussian Mixture - Agrupamento04 (k=5 clusters encontrados)



[Abrir versão interativa \(p3_gmm.html\)](#)

3.8) OPTICS

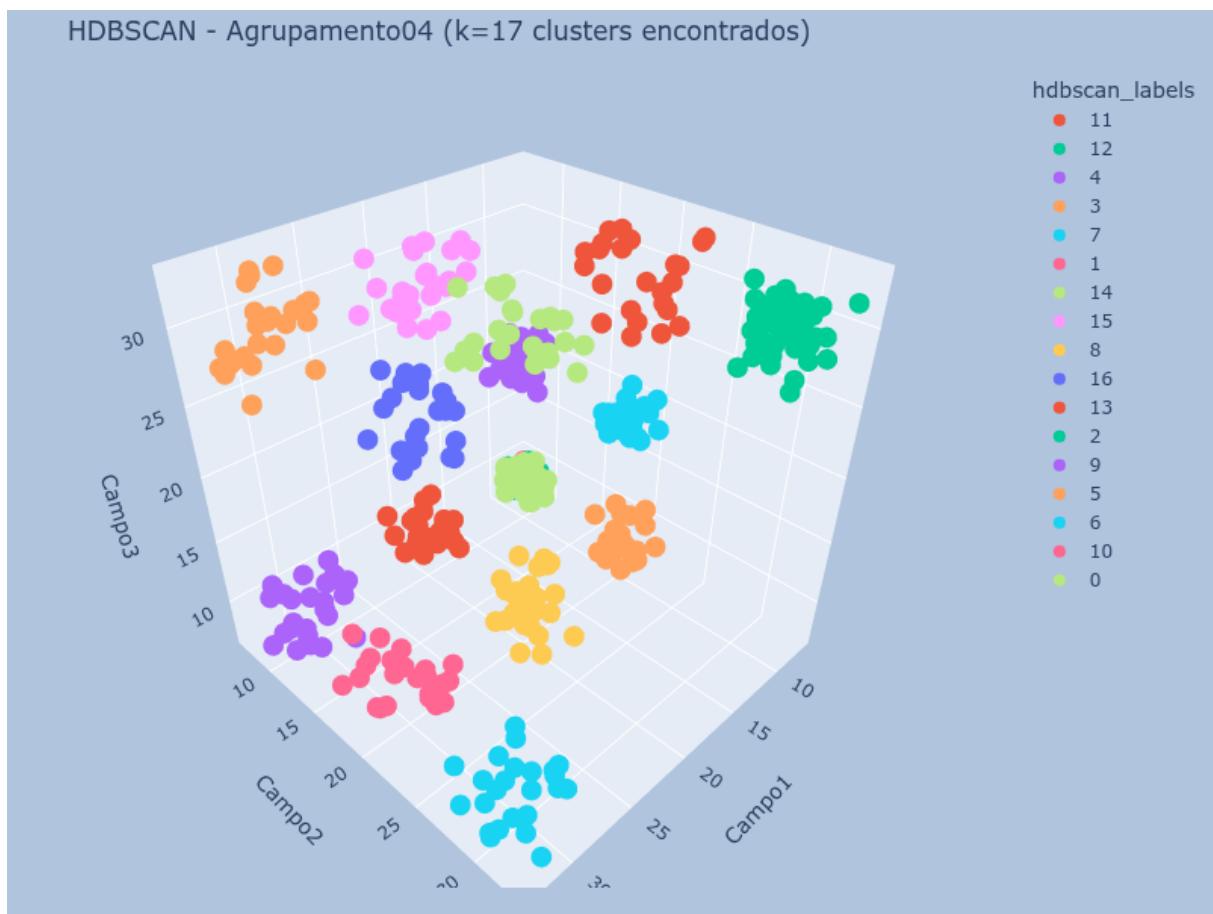
Segundo Trabalho Prático sobre Agrupamento



[Abrir versão interativa \(p3_optics.html\)](#)

3.9) HDBSCAN

Segundo Trabalho Prático sobre Agrupamento

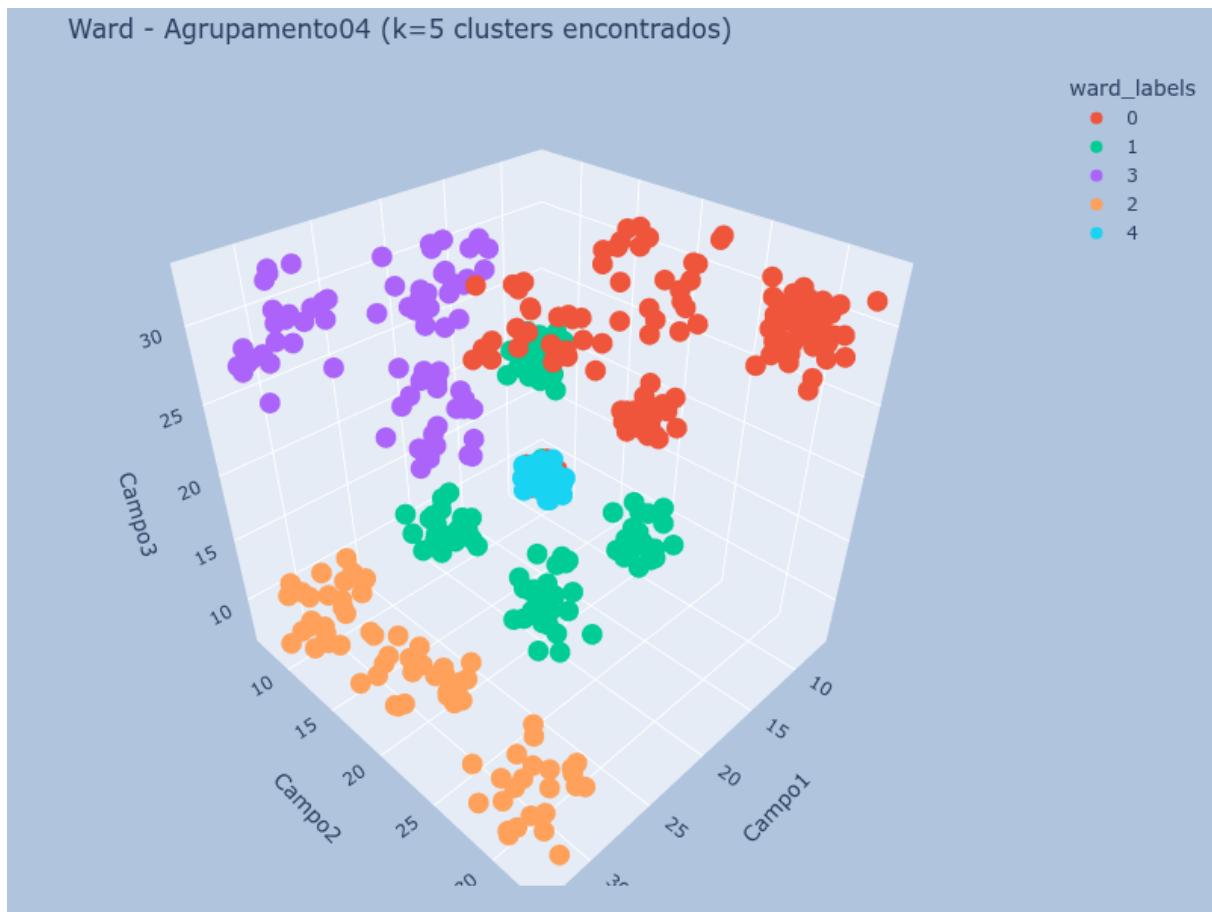


[Abrir versão interativa \(p3_hdbscan.html\)](#)

3.10) Ward

Segundo Trabalho Prático sobre Agrupamento

Ward - Agrupamento04 (k=5 clusters encontrados)

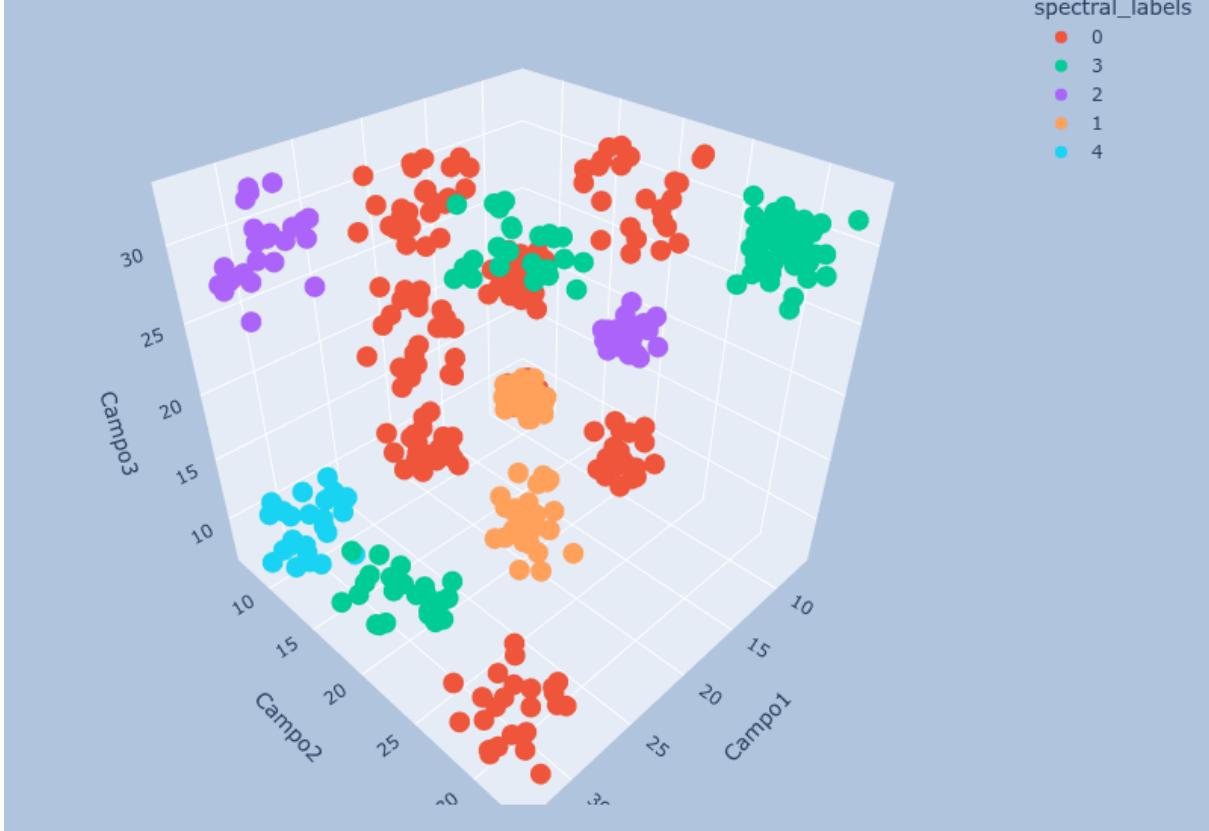


[Abrir versão interativa \(p3_ward.html\)](#)

3.11) Spectral Clustering

Segundo Trabalho Prático sobre Agrupamento

Spectral Clustering - Agrupamento04 (k=5 clusters encontrados)

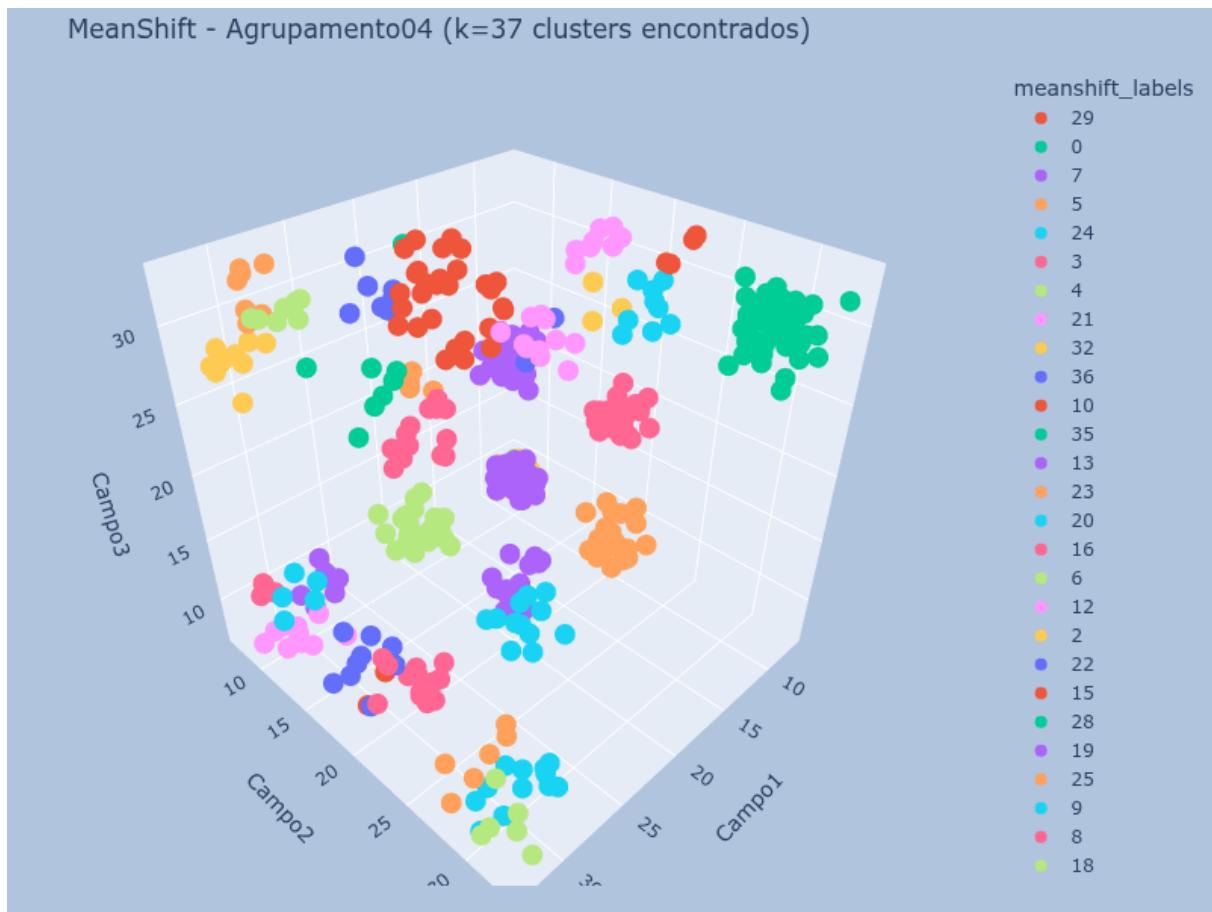


[Abrir versão interativa \(p3_spectral.html\)](#)

3.12) MeanShift

Segundo Trabalho Prático sobre Agrupamento

MeanShift - Agrupamento04 (k=37 clusters encontrados)

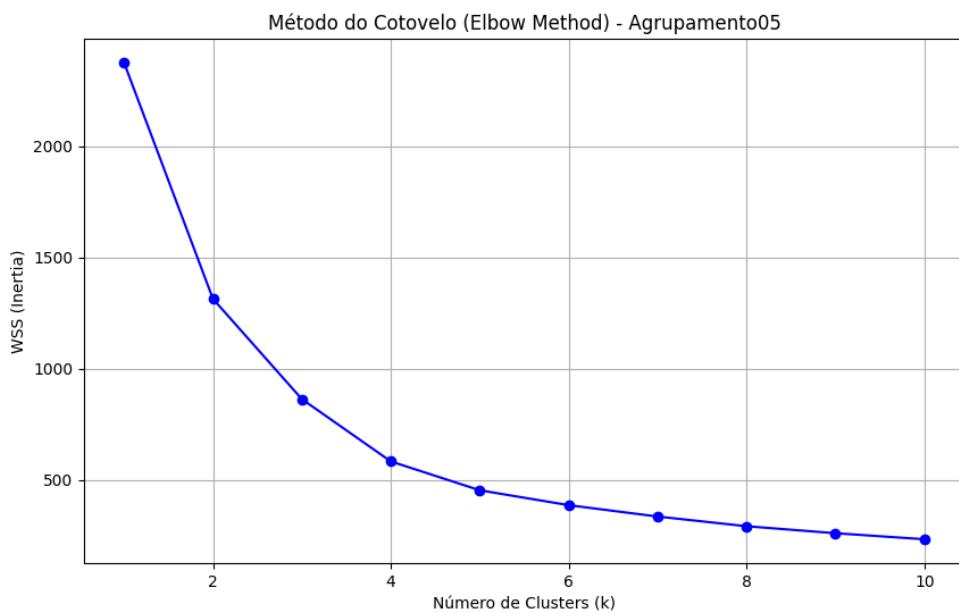


[Abrir versão interativa \(p3_meanshift.html\)](#)

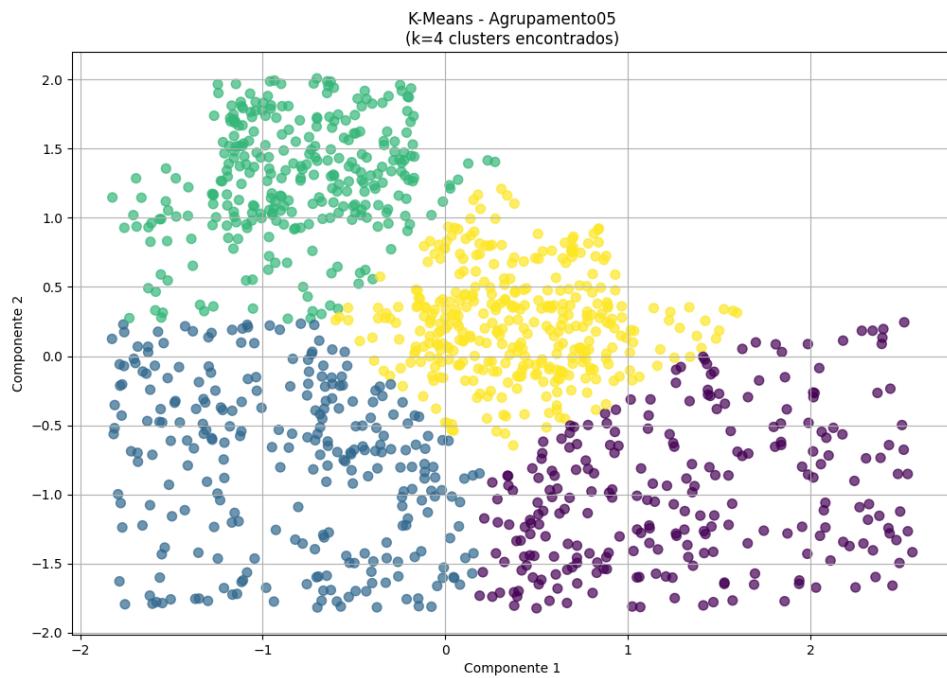
Segundo Trabalho Prático sobre Agrupamento

Problema 4: Agrupamento05.txt

Metodo do Cotovelo (Elbow Method)

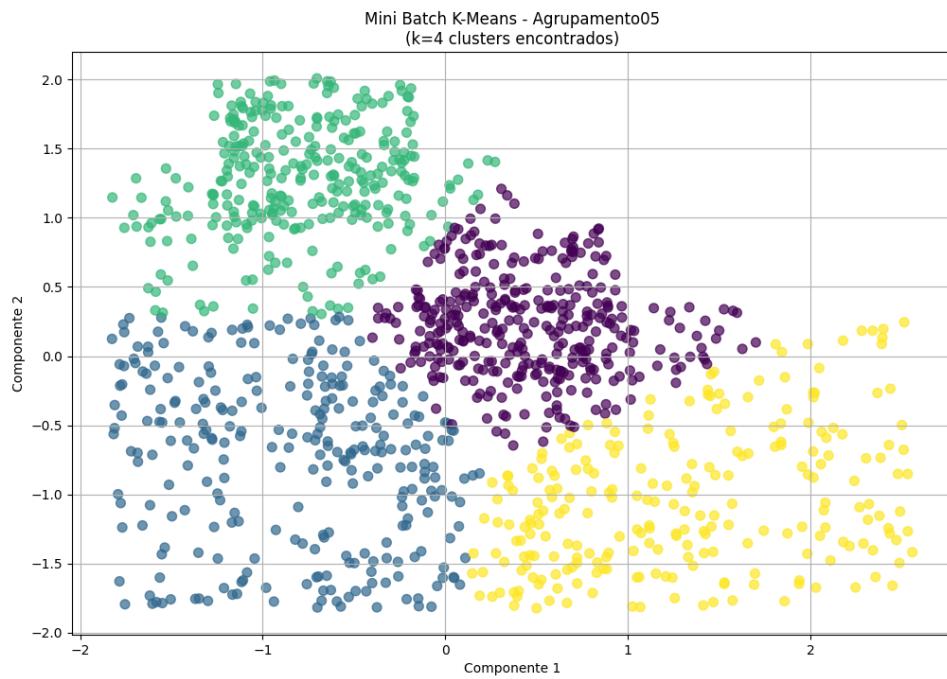


4.1) K-Means

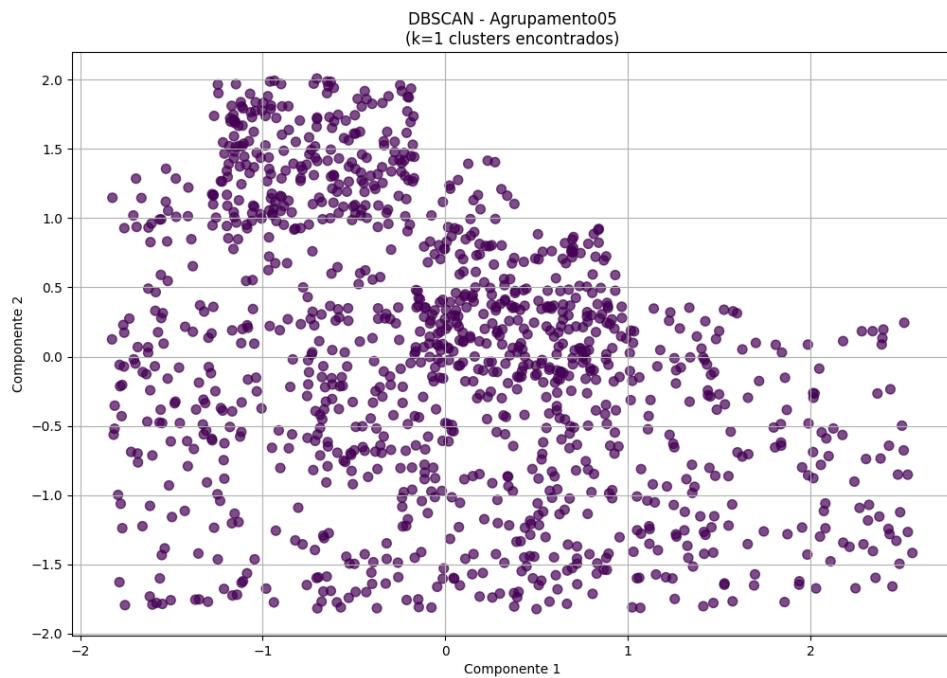


4.2) Mini Batch K-Means

Segundo Trabalho Prático sobre Agrupamento

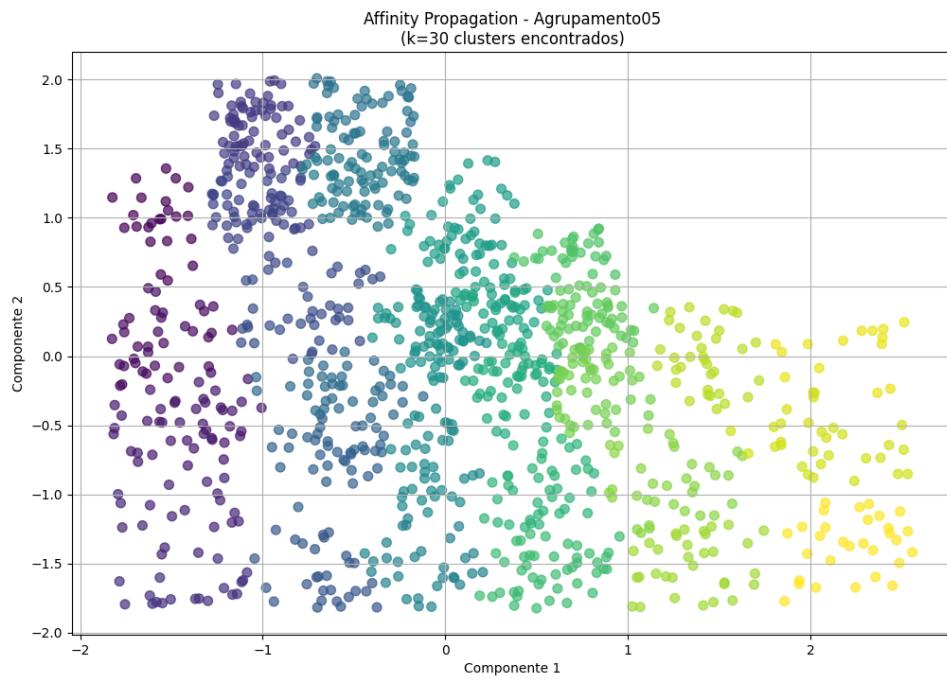


4.3) DBSCAN

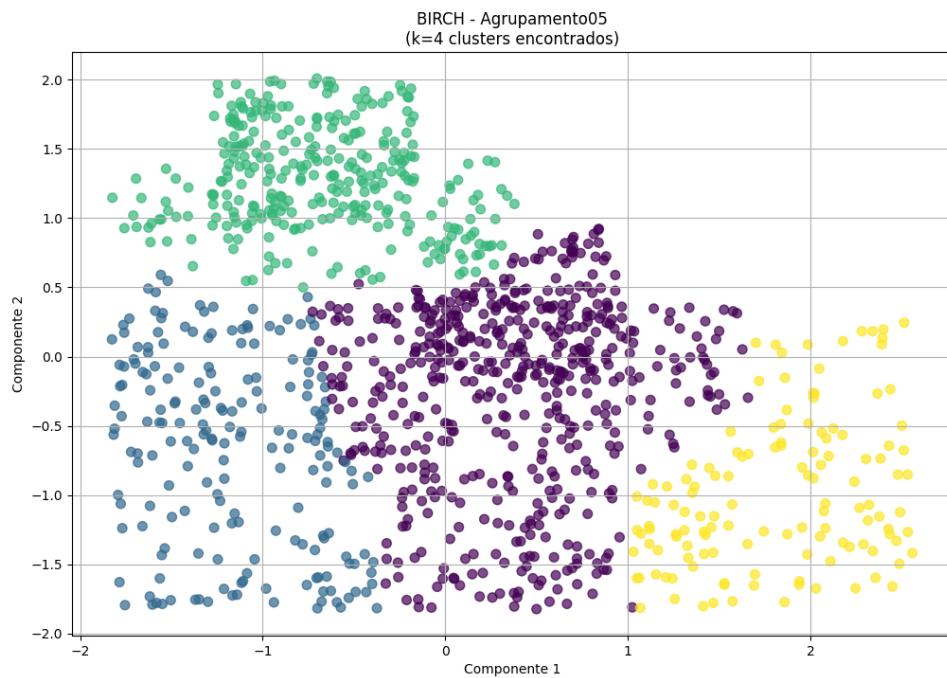


4.4) Affinity Propagation

Segundo Trabalho Prático sobre Agrupamento

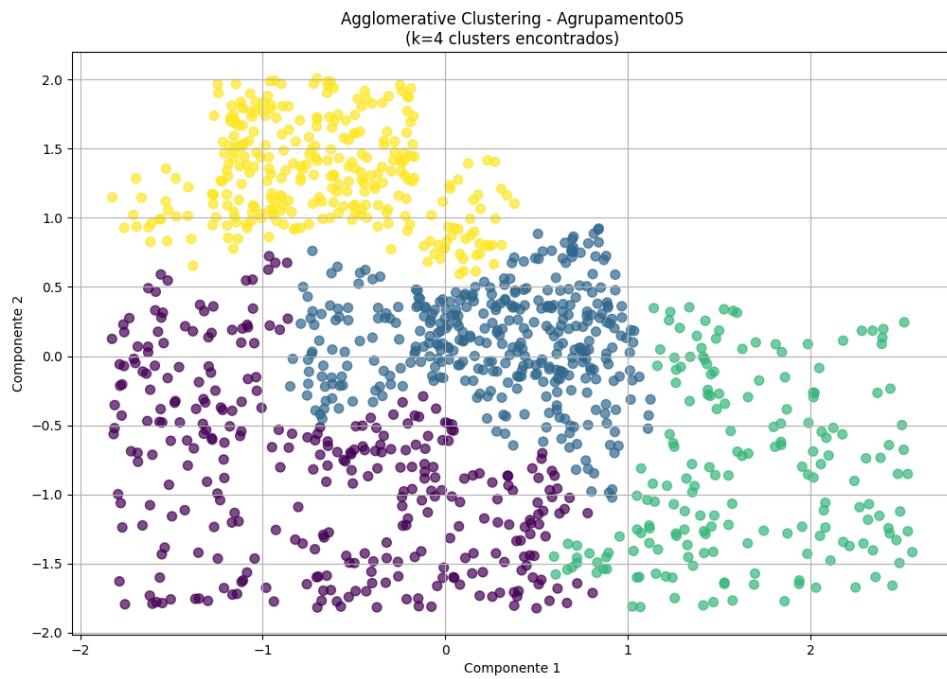


4.5) BIRCH

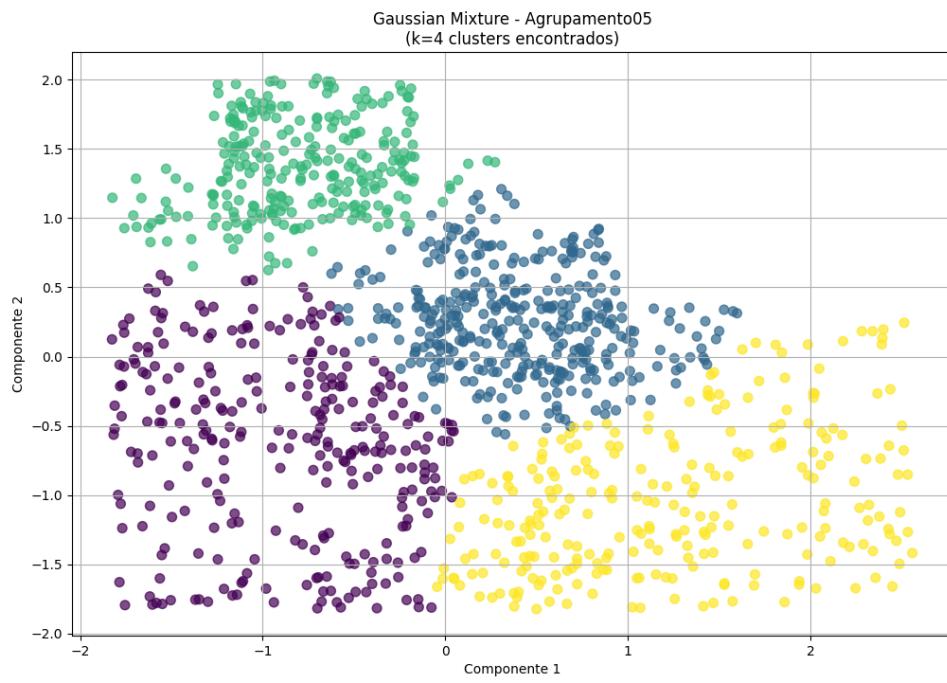


4.6) Agglomerative Clustering

Segundo Trabalho Prático sobre Agrupamento

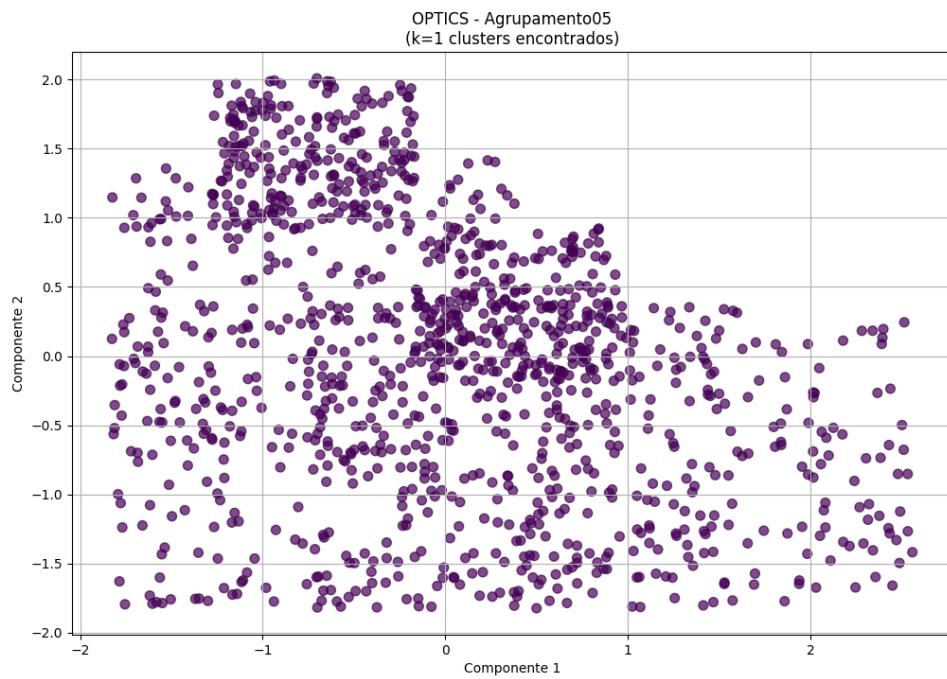


4.7) Gaussian Mixture

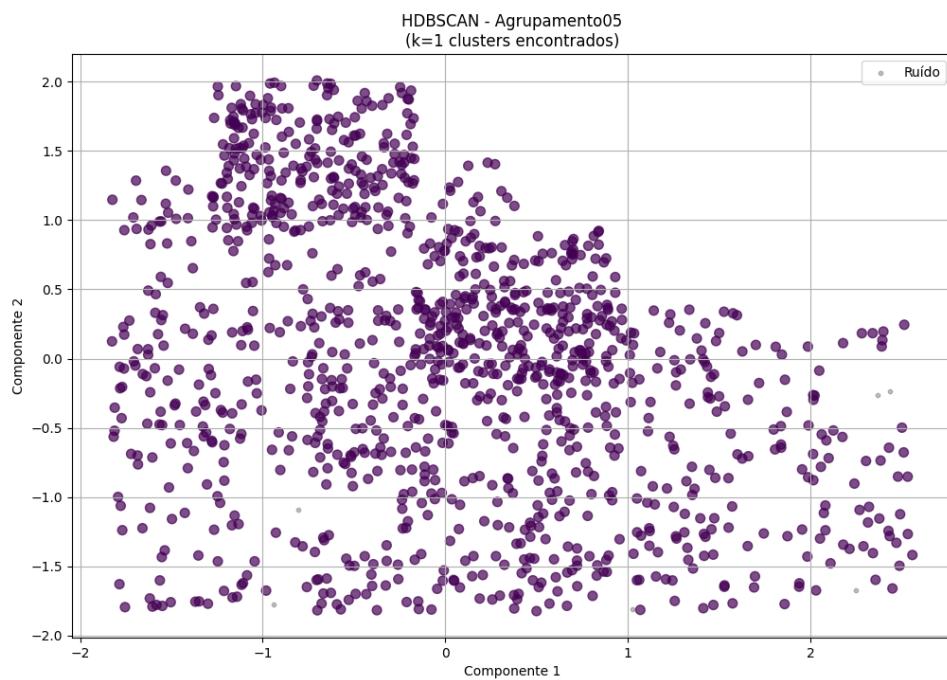


4.8) OPTICS

Segundo Trabalho Prático sobre Agrupamento

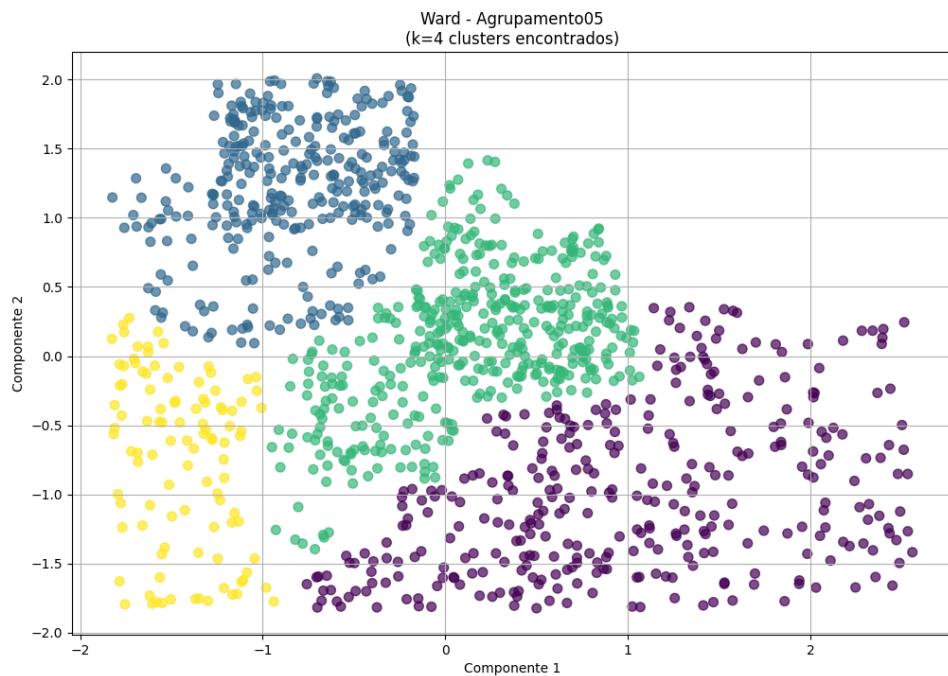


4.9) HDBSCAN

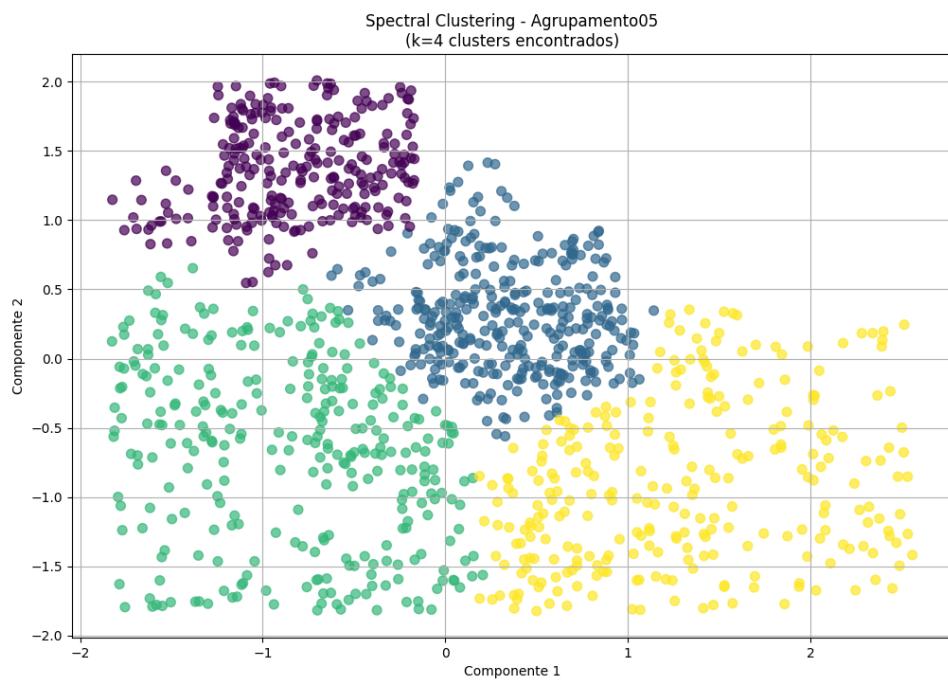


4.10) Ward

Segundo Trabalho Prático sobre Agrupamento

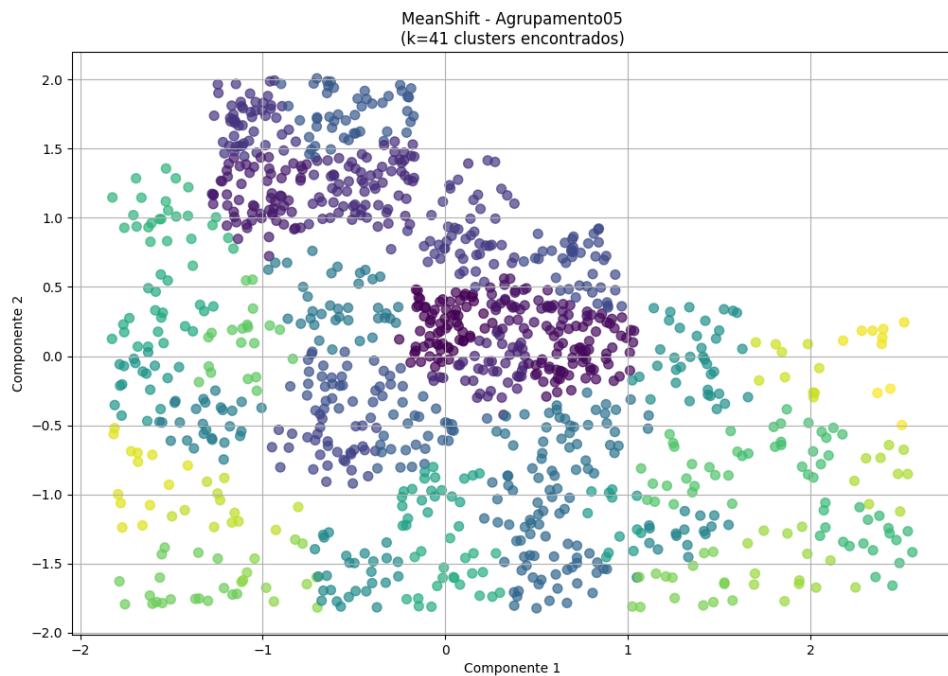


4.11) Spectral Clustering



4.12) MeanShift

Segundo Trabalho Prático sobre Agrupamento



Segundo Trabalho Prático sobre Agrupamento

Anexo: Código Fonte (resolver_trabalho.py)

```
# -*- coding: utf-8 -*-
import os
import warnings
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import (
    KMeans,
    MiniBatchKMeans,
    DBSCAN,
    AffinityPropagation,
    Birch,
    AgglomerativeClustering,
    OPTICS,
    MeanShift,
    SpectralClustering,
    HDBSCAN,
)
from sklearn.mixture import GaussianMixture
from sklearn.neighbors import kneighbors_graph
from sklearn.utils._testing import ignore_warnings # Para ignorar UserWarnings
from sklearn.exceptions import ConvergenceWarning

# Importar kaleido não é necessário, mas ele precisa estar instalado
# import kaleido

# Ignorar avisos para uma saída mais limpa
warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", category=ConvergenceWarning)

# --- Constantes de Diretório ---
INPUT_DIR = "./content"
OUTPUT_DIR = "./result"

def plotar_grafico_cotovelo(X_scaled, title_suffix, output_basename):
    """
    Calcula e salva o gráfico do Método do Cotovelo (WSS).

    """
    print(f" Calculando WSS para o Método do Cotovelo ({title_suffix})...")
    wss = []
    K_range = range(1, 11)

    for k in K_range:
        kmeans_elbow = KMeans(n_clusters=k, random_state=42, n_init=10)
        kmeans_elbow.fit(X_scaled)
        wss.append(kmeans_elbow.inertia_)

    plt.figure(figsize=(10, 6))
    plt.plot(K_range, wss, "bo-")
    plt.xlabel("Número de Clusters (k)")
    plt.ylabel("WSS (Inertia)")
```

Segundo Trabalho Prático sobre Agrupamento

```
plt.title(f"Método do Cotovelo (Elbow Method) - {title_suffix}")
plt.grid(True)
cotovelos_path = os.path.join(OUTPUT_DIR, f"{output_basename}_grafico_cotovelos.png")
plt.savefig(cotovelos_path)
plt.close()
print(f" Gráfico do Cotovelo salvo em: {cotovelos_path}")

def plotar_grafico_2d(X_plot, labels, title, output_path, centers=None):
    """
    Função auxiliar para criar e salvar gráficos 2D (PNG) com Matplotlib.
    """
    plt.figure(figsize=(12, 8))
    # 'labels == -1' é para ruído (comum em DBSCAN, OPTICS, HDBSCAN)
    # Damos a eles uma cor cinza e tamanho menor
    if -1 in labels:
        noise_mask = labels == -1
        plt.scatter(
            X_plot[noise_mask, 0],
            X_plot[noise_mask, 1],
            c="gray",
            s=10,
            alpha=0.5,
            label="Ruído",
        )
        # Plota os pontos não-ruído
        core_mask = ~noise_mask
        plt.scatter(
            X_plot[core_mask, 0],
            X_plot[core_mask, 1],
            c=labels[core_mask],
            s=50,
            cmap="viridis",
            alpha=0.7,
        )
    else:
        # Plota normalmente se não houver ruído
        plt.scatter(
            X_plot[:, 0],
            X_plot[:, 1],
            c=labels,
            s=50,
            cmap="viridis",
            alpha=0.7,
        )

    # Plota centróides, se fornecidos
    if centers is not None:
        plt.scatter(
            centers[:, 0],
            centers[:, 1],
            c="red",
            s=250,
            marker="X",
            alpha=0.9,
            label="Centróides",
        )

# Métodos de agrupamento
```

Segundo Trabalho Prático sobre Agrupamento

```
plt.title(title)
plt.xlabel("Componente 1" if X_plot.shape[1] > 1 else X_plot.columns[0])
plt.ylabel("Componente 2" if X_plot.shape[1] > 1 else X_plot.columns[1])
if -1 in labels or centers is not None:
    plt.legend()
plt.grid(True)
plt.savefig(output_path)
plt.close()

def plotar_grafico_3d(df_plot, labels_col, title, output_basename):
    """
    Função auxiliar para criar e salvar gráficos 3D interativos (HTML)
    e estáticos (PNG). (Reutilizada do TP1)
    """
    print(f" Gerando gráfico 3D: {title}...")

    # Converte labels para string para Plotly tratar cores discretas
    df_plot[labels_col] = df_plot[labels_col].astype(str)

    fig = px.scatter_3d(
        df_plot,
        x=df_plot.columns[0],
        y=df_plot.columns[1],
        z=df_plot.columns[2],
        color=labels_col,
        title=title,
        color_discrete_map={"-1": "grey"}, # Mapeia ruído para cinza
    )

    fig.update_layout(
        margin=dict(l=20, r=20, t=40, b=20), paper_bgcolor="LightSteelBlue"
    )

    html_file_path = f"{output_basename}.html"
    fig.write_html(html_file_path)

    png_file_path = f"{output_basename}.png"
    try:
        fig.write_image(png_file_path, width=800, height=600)
    except Exception as e:
        print(f" AVISO: Não foi possível salvar a imagem estática {png_file_path}.")
        print(" Certifique-se que 'kaleido' está instalado: pip install kaleido")
        print(f" Erro: {e}")

def executar_e_plotar_algoritmos(
    X_scaled, X_plot, dataset_name, problem_prefix, k_ideal, is_3d=False
):
    """
    Executa todos os 12 algoritmos de clusterização e salva seus gráficos.
    """
    print(f"\nIniciando execução dos 12 algoritmos para {dataset_name}...")

    # X_plot pode ser um DataFrame (3D) ou np.array (2D)
    df_plot = None
    if is_3d:
        df_plot = X_plot.copy()
```

Segundo Trabalho Prático sobre Agrupamento

```
# Parâmetros (alguns baseados no Colab do professor)
params = {
    "quantile": 0.3,
    "eps": 0.5,
    "min_samples": 10,
    "damping": 0.9,
    "preference": -200,
    "n_neighbors": 3,
    "hdbscan_min_cluster_size": 15,
    "hdbscan_min_samples": 3,
    "optics_min_samples": 10,
    "optics_xi": 0.05,
    "optics_min_cluster_size": 0.1,
}

# Estrutura dos algoritmos
# (Nome Amigável, nome_arquivo, instância_do_algoritmo)

# Conectividade para Ward e Agglomerative
connectivity = kneighbors_graph(
    X_scaled, n_neighbors=params["n_neighbors"], include_self=False
)
connectivity = 0.5 * (connectivity + connectivity.T)

# Largura de banda para MeanShift
bandwidth = 0.3 # Valor padrão, estimate_bandwidth pode ser lento
# try:
#     bandwidth = cluster.estimate_bandwidth(X_scaled, quantile=params["quantile"])
# except Exception:
#     print(" Aviso: estimate_bandwidth falhou, usando padrão 0.3.")
#     bandwidth = 0.3

algoritmos = [
    ("K-Means", "kmeans", KMeans(n_clusters=k_ideal, random_state=42, n_init=10)),
    (
        "Mini Batch K-Means",
        "minibatch_kmeans",
        MiniBatchKMeans(n_clusters=k_ideal, random_state=42, n_init=10),
    ),
    (
        "DBSCAN",
        "dbSCAN",
        DBSCAN(eps=params["eps"], min_samples=params["min_samples"]),
    ),
    (
        "Affinity Propagation",
        "affinity",
        AffinityPropagation(damping=params["damping"], random_state=42),
    ),
    ("BIRCH", "birch", Birch(n_clusters=k_ideal)),
    (
        "Agglomerative Clustering",
        "agglomerative",
        AgglomerativeClustering(n_clusters=k_ideal),
    ),
    (
        "Gaussian Mixture",
        "gaussian_mixture",
        GaussianMixture(n_components=k_ideal, covariance_type="full"),
    ),
]
```

Segundo Trabalho Prático sobre Agrupamento

```
"gmm",
GaussianMixture(n_components=k_ideal, random_state=42),
),
(
    "OPTICS",
    "optics",
    OPTICS(
        min_samples=params[ "optics_min_samples" ],
        xi=params[ "optics_xi" ],
        min_cluster_size=params[ "optics_min_cluster_size" ],
    ),
),
(
    "HDBSCAN",
    "hdbscan",
    HDBSCAN(
        min_cluster_size=params[ "hdbscan_min_cluster_size" ],
        min_samples=params[ "hdbscan_min_samples" ],
        allow_single_cluster=True,
    ),
),
(
    "Ward",
    "ward",
    AgglomerativeClustering(
        n_clusters=k_ideal, linkage="ward", connectivity=connectivity
    ),
),
(
    "Spectral Clustering",
    "spectral",
    SpectralClustering(
        n_clusters=k_ideal, affinity="nearest_neighbors", random_state=42
    ),
),
("MeanShift", "meanshift", MeanShift(bandwidth=bandwidth, bin_seeding=True)),
]

for nome_amigavel, nome_arquivo, algoritmo in algoritmos:
    print(f"  Executando ({nome_amigavel})...")

    output_basename = os.path.join(OUTPUT_DIR, f"{problem_prefix}_{nome_arquivo}")

    # Ajuste para GMM que não tem 'fit_predict'
    if hasattr(algoritmo, "fit_predict"):
        labels = algoritmo.fit_predict(X_scaled)
    else:
        algoritmo.fit(X_scaled)
        labels = algoritmo.predict(X_scaled)

    # Ajuste para Affinity Propagation e K-Means que têm centróides
    centers = None
    if hasattr(algoritmo, "cluster_centers_"):
        # Centros estão no espaço normalizado, precisamos "desnormalizar"
        # Se X_plot é 2D (PCA), precisamos transformar os centros para PCA
        if not is_3d and X_scaled.shape[1] != X_plot.shape[1]:
            # Caso especial do PCA (Problema 2)
            # Encontrar um 'scaler' de PCA para os centros
```

Segundo Trabalho Prático sobre Agrupamento

```
# Isso é complexo, vamos omitir os centros do PCA por simplicidade
centers = None
elif not is_3d:
    # Caso 2D (Problemas 1 e 4)
    # Precisamos de um 'scaler' 2D.
    # Vamos simplificar: plotar centros no espaço escalado (PCA=False)
    # O X_plot deve ser o X_scaled para isso funcionar
    # Vamos re-plotar no espaço escalado para P1 e P4
    centers = None # Simplificando para não plotar centros no 2D
else:
    centers = None # Não plotamos centros no 3D

n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
title = (
    f"{{nome_amigavel}} - {{dataset_name}}\n{k={n_clusters} clusters encontrados}"
)

if is_3d:
    labels_col_name = f"{{nome_arquivo}}_labels"
    df_plot[labels_col_name] = labels
    plotar_grafico_3d(df_plot, labels_col_name, title, output_basename)
else:
    # Para P1 e P4, X_plot = X_scaled
    # Para P2, X_plot = X_pca
    output_png_path = f"{{output_basename}}.png"
    plotar_grafico_2d(X_plot, labels, title, output_png_path, centers=centers)

print(f" {{dataset_name}} concluído.")

# ---
# PROBLEMA 1: Agrupamento03.txt (2D)
# ---
def resolver_problema_1(input_dir, output_dir):
    print("\nIniciando Problema 1 (Agrupamento03.txt)...")
    file_path = os.path.join(input_dir, "Agrupamento03.txt")
    try:
        df = pd.read_csv(file_path)
    except FileNotFoundError:
        print(f"ERRO: Arquivo não encontrado em {file_path}")
        return

    X = df.iloc[:, 0:2]

    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    plotar_grafico_cotovelo(X_scaled, "Agrupamento03", "p1")

    k_ideal_p1 = 5 # Valor chutado após ver o cotovelo P1

    # Para P1, plotamos os dados escalados (X_scaled)
    executar_e_plotar_algoritmos(
        X_scaled, X_scaled, "Agrupamento03", "p1", k_ideal_p1, is_3d=False
    )

# ---
```

Segundo Trabalho Prático sobre Agrupamento

```
# PROBLEMA 2: iris_cluster.txt (4D)
# ---
def resolver_problema_2(input_dir, output_dir):
    print("\nIniciando Problema 2 (iris_cluster.txt)...")
    file_path = os.path.join(input_dir, "iris_cluster.txt")
    try:
        df = pd.read_csv(file_path)
    except FileNotFoundError:
        print(f"ERRO: Arquivo não encontrado em {file_path}")
        return

    # Ignora a coluna 'variety' que é o rótulo real
    X = df.drop("variety", axis=1)

    scaler = StandardScaler()
    X_scaled_4d = scaler.fit_transform(X)

    plotar_grafico_cotovelo(X_scaled_4d, "Iris", "p2")

    k_ideal_p2 = 3 # Dataset Iris classicamente tem 3 clusters

    print(" Aplicando PCA para visualização 2D...")
    pca = PCA(n_components=2, random_state=42)
    X_pca_2d = pca.fit_transform(X_scaled_4d)

    # Treinamos no 4D (X_scaled_4d), mas plotamos no 2D (X_pca_2d)
    executar_e_plotar_algoritmos(
        X_scaled_4d, X_pca_2d, "Iris (PCA 2D)", "p2", k_ideal_p2, is_3d=False
    )

# ---
# PROBLEMA 3: Agrupamento04.txt (3D)
# ---
def resolver_problema_3(input_dir, output_dir):
    print("\nIniciando Problema 3 (Agrupamento04.txt)...")
    file_path = os.path.join(input_dir, "Agrupamento04.txt")
    try:
        df = pd.read_csv(file_path)
    except FileNotFoundError:
        print(f"ERRO: Arquivo não encontrado em {file_path}")
        return

    X = df.iloc[:, 0:3]

    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    plotar_grafico_cotovelo(X_scaled, "Agrupamento04", "p3")

    k_ideal_p3 = 5 # Valor chutado após ver o cotovelo P3

    # Para P3, passamos o DataFrame original para plotagem 3D
    executar_e_plotar_algoritmos(
        X_scaled, X, "Agrupamento04", "p3", k_ideal_p3, is_3d=True
    )
```

Segundo Trabalho Prático sobre Agrupamento

```
# ---
# PROBLEMA 4: Agrupamento05.txt (2D)
# ---

def resolver_problema_4(input_dir, output_dir):
    print("\nIniciando Problema 4 (Agrupamento05.txt)...")
    file_path = os.path.join(input_dir, "Agrupamento05.txt")
    try:
        df = pd.read_csv(file_path)
    except FileNotFoundError:
        print(f"ERRO: Arquivo não encontrado em {file_path}")
        return

    X = df.iloc[:, 0:2]

    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    plotar_grafico_cotovelos(X_scaled, "Agrupamento05", "p4")

    k_ideal_p4 = 4 # Valor chutado após ver o cotovelo P4

    # Para P4, plotamos os dados escalados (X_scaled)
    executar_e_plotar_algoritmos(
        X_scaled, X_scaled, "Agrupamento05", "p4", k_ideal_p4, is_3d=False
    )

# --- Bloco de Execução Principal ---
if __name__ == "__main__":
    os.makedirs(OUTPUT_DIR, exist_ok=True)
    os.makedirs(INPUT_DIR, exist_ok=True)

    # Lembrete para o usuário colocar os arquivos
    print(f"Iniciando Segundo Trabalho Prático...")
    print(f"Verificando arquivos de entrada em: {os.path.abspath(INPUT_DIR)}")
    print(
        "Certifique-se que 'Agrupamento03.txt', 'iris_cluster.txt', 'Agrupamento04.txt', 'Agrupamento05.txt' estão lá."
    )
    print(f"Resultados serão salvos em: {os.path.abspath(OUTPUT_DIR)}\n")

    resolver_problema_1(INPUT_DIR, OUTPUT_DIR)
    resolver_problema_2(INPUT_DIR, OUTPUT_DIR)
    resolver_problema_3(INPUT_DIR, OUTPUT_DIR)
    resolver_problema_4(INPUT_DIR, OUTPUT_DIR)

    print("\n--- Processo Concluído ---")
    print(f"Todos os gráficos (PNG e HTML) foram salvos no diretório '{OUTPUT_DIR}' .")
```