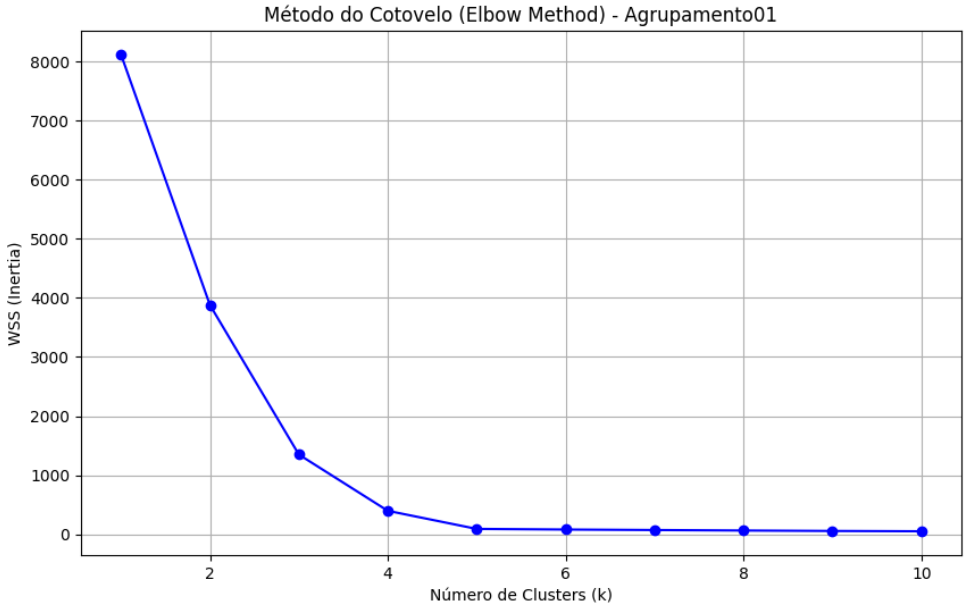
Primeiro Trabalho Prático sobre Agrupamento

Aluno: Pedro Henrique Vilaça Valverde

Disciplina: Inteligência Artificial

Problema 1: Agrupamento01.txt

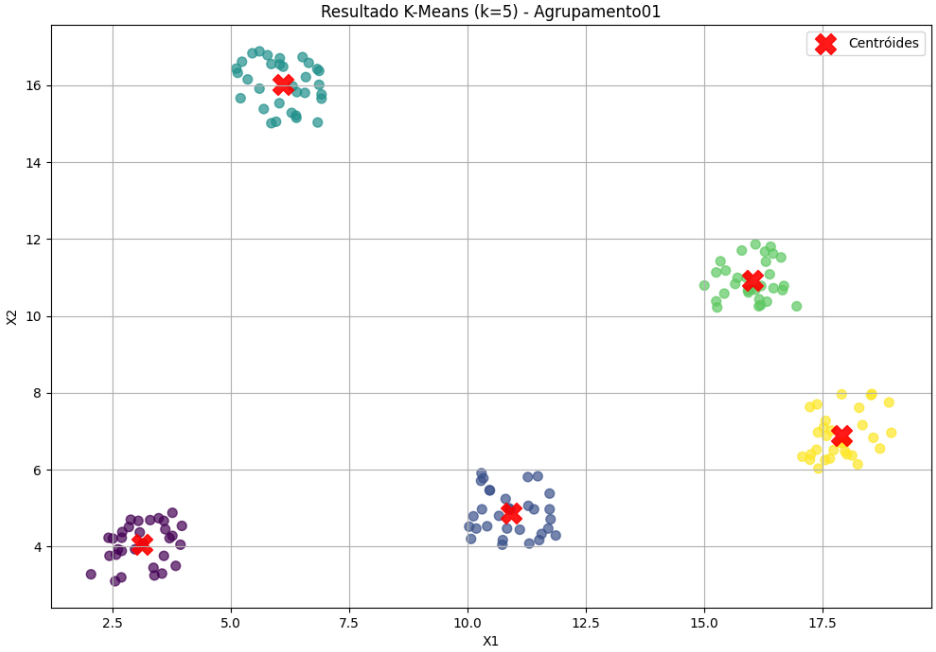
Metodo do Cotovelo (Elbow Method)



Centros Encontrados (K-Means k=5)

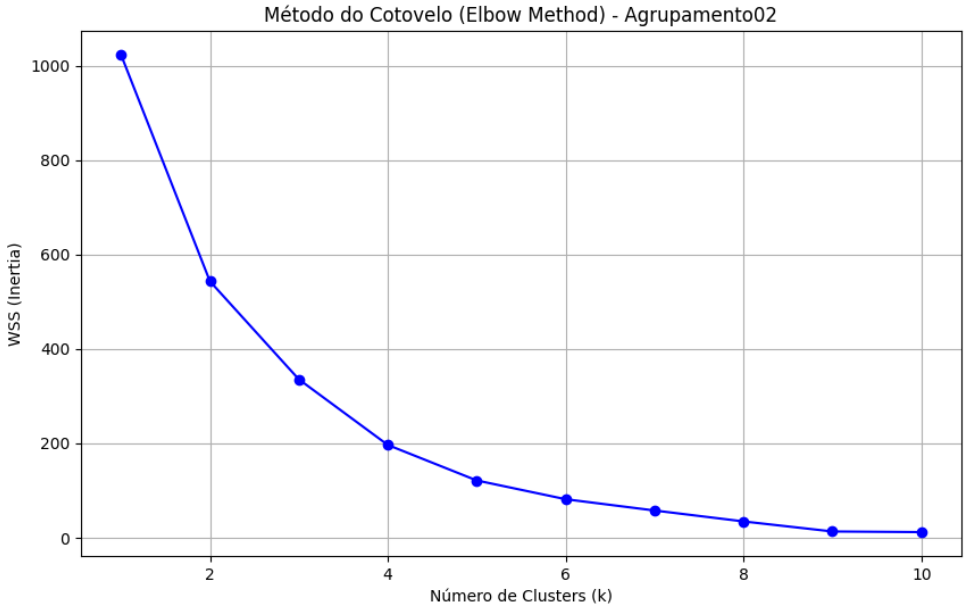
	X1	X2
Centro 1	3.124839	4.065806
Centro 2	10.917742	4.871290
Centro 3	6.109063	16.025000
Centro 4	16.016452	10.914516
Centro 5	17.895484	6.884194

Grafico de Resultado (K-Means k=5)


Resultado K-Means (k=5) - Agrupamento01
Centróides
X1
X2
2.5
5.0
7.5
10.0
12.5
15.0
17.5
4
6
8
10
12
14
16

Problema 2: Agrupamento02.txt

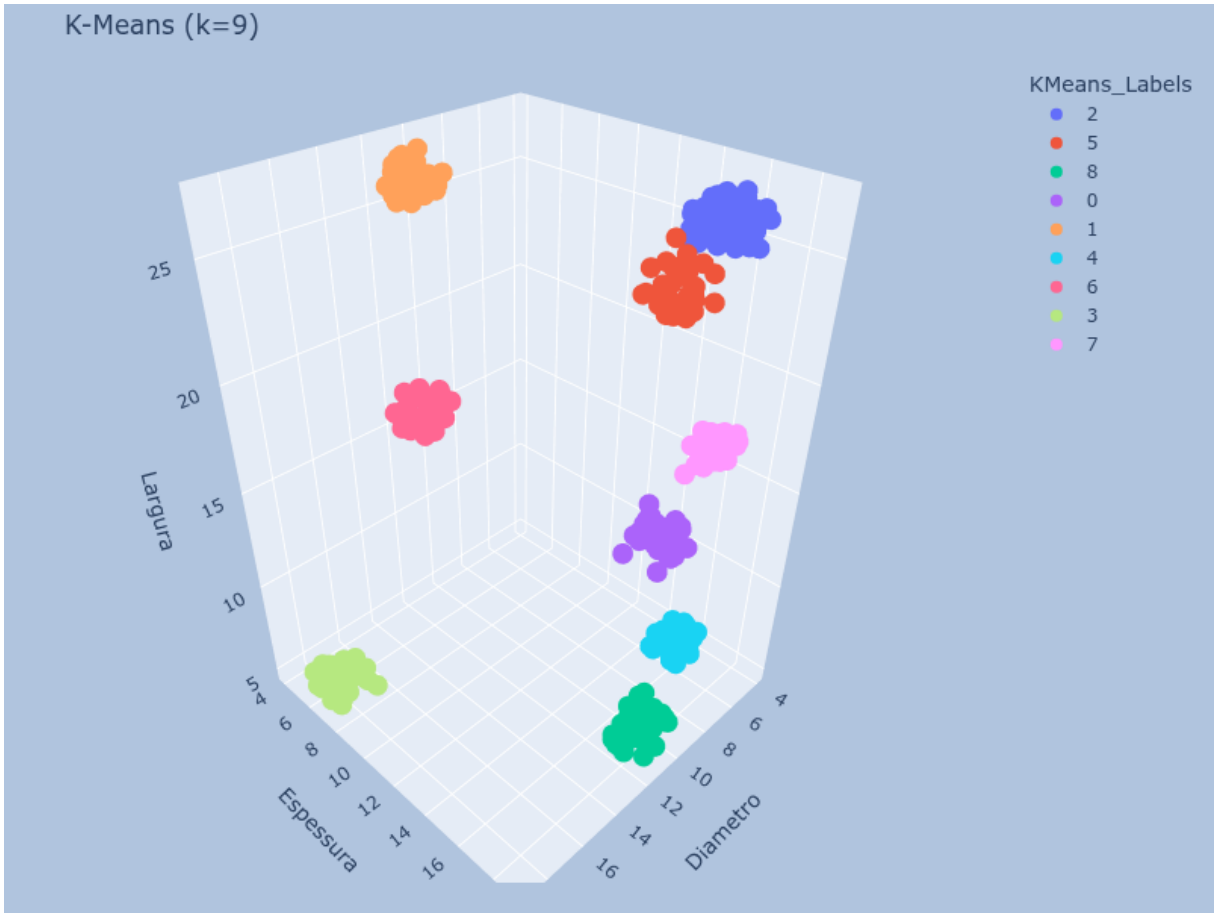
Metodo do Cotovelo (Elbow Method)



a) K-Means

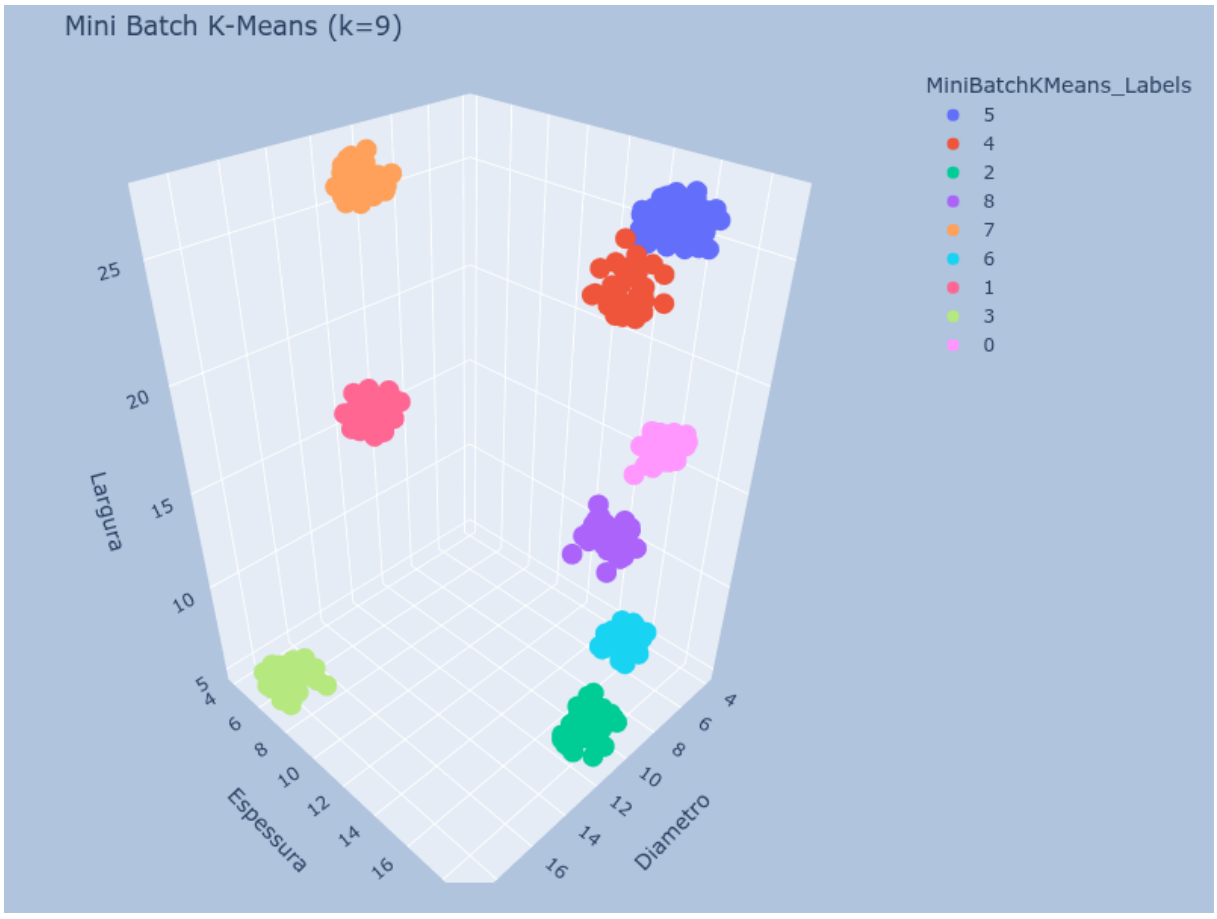
Centros Encontrados:

	Diametro	Espessura	Largura
Centro 1	10.900132	18.002487	16.096316
Centro 2	11.047335	5.048835	26.166161
Centro 3	5.754969	15.909032	26.099046
Centro 4	18.198806	6.876819	5.949503
Centro 5	5.833945	16.027929	6.011981
Centro 6	11.162219	17.927010	25.918381
Centro 7	11.047265	5.034100	15.911429
Centro 8	4.856926	15.938284	16.212552
Centro 9	11.070116	17.949994	5.923881



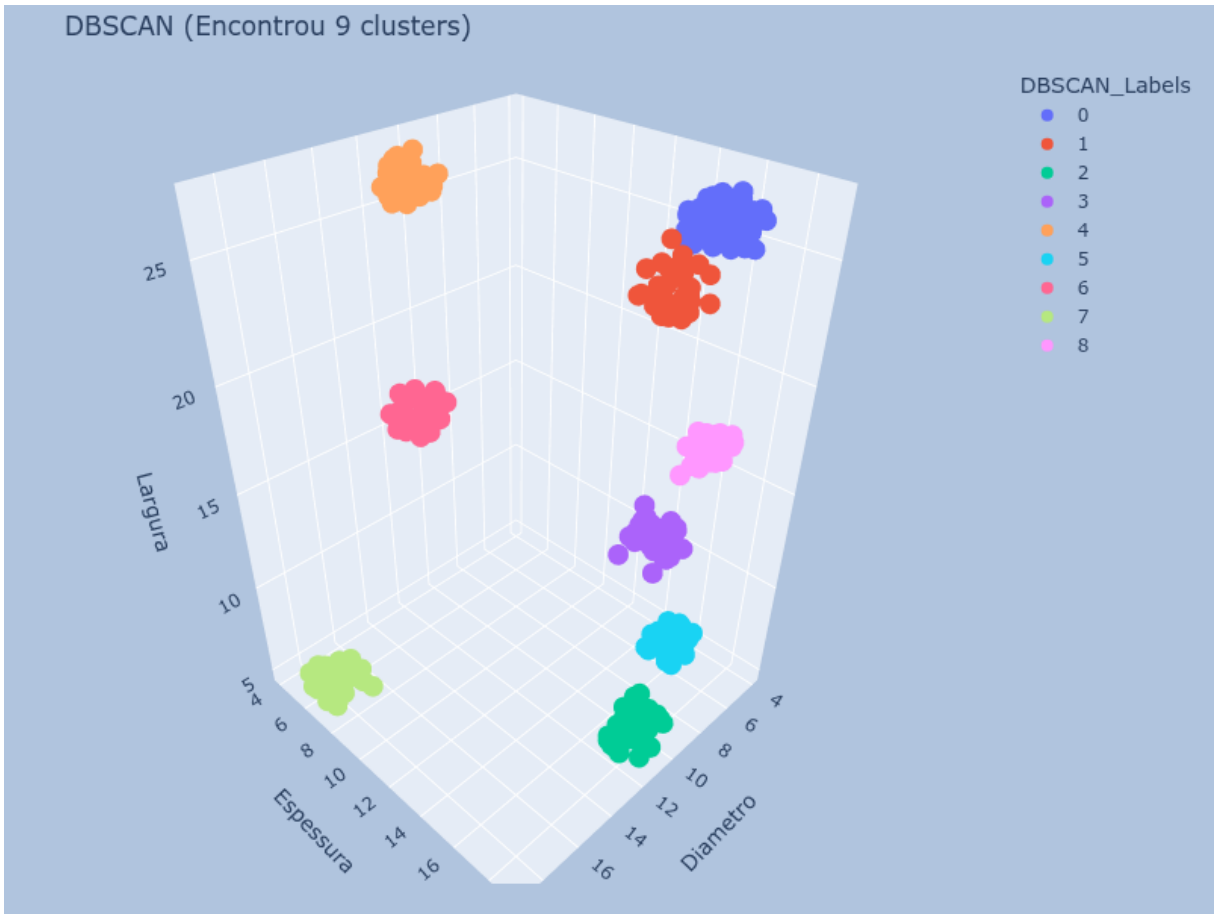
Abrir versao interativa (p2_kmeans.html)

b) Mini Batch K-Means



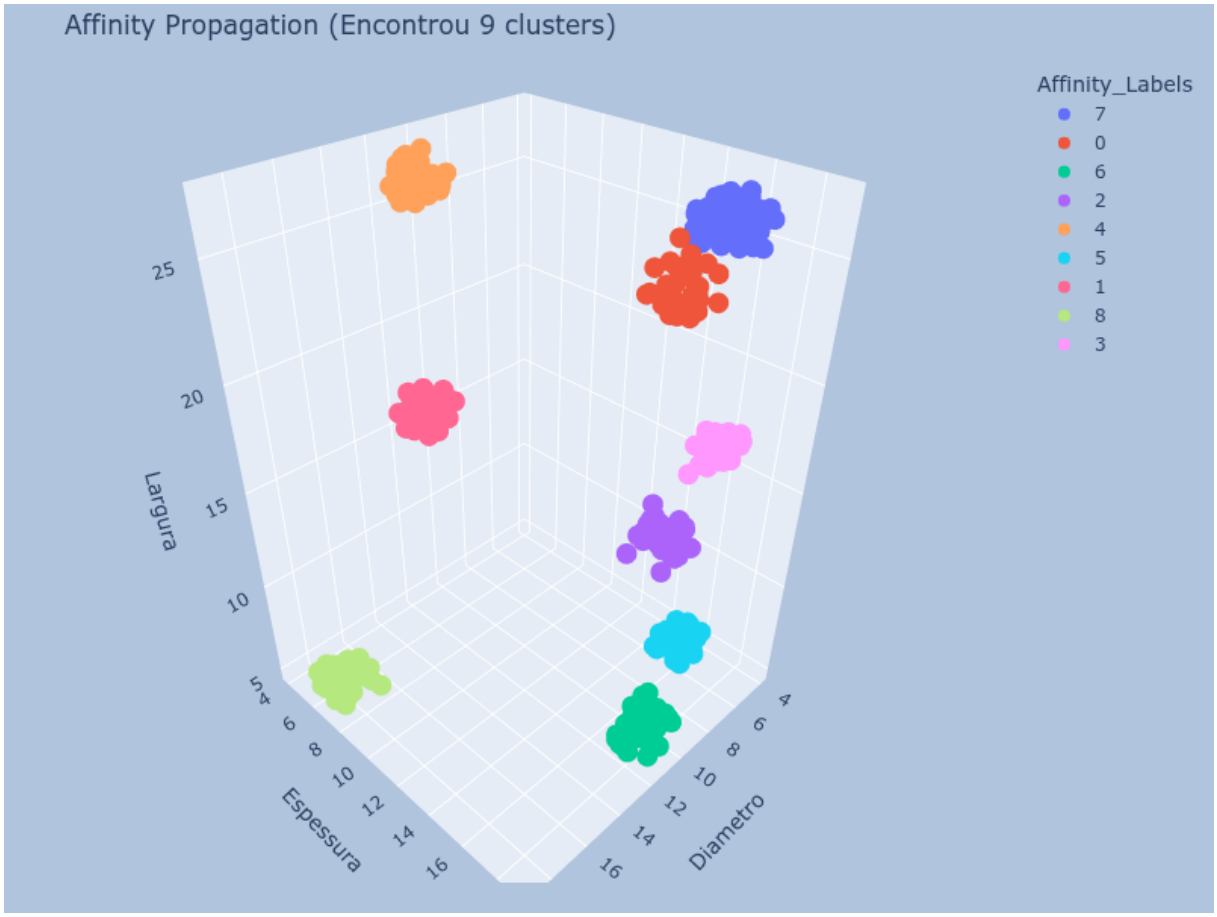
[Abrir versao interativa (p2_minibatch_kmeans.html)]

c) DBSCAN



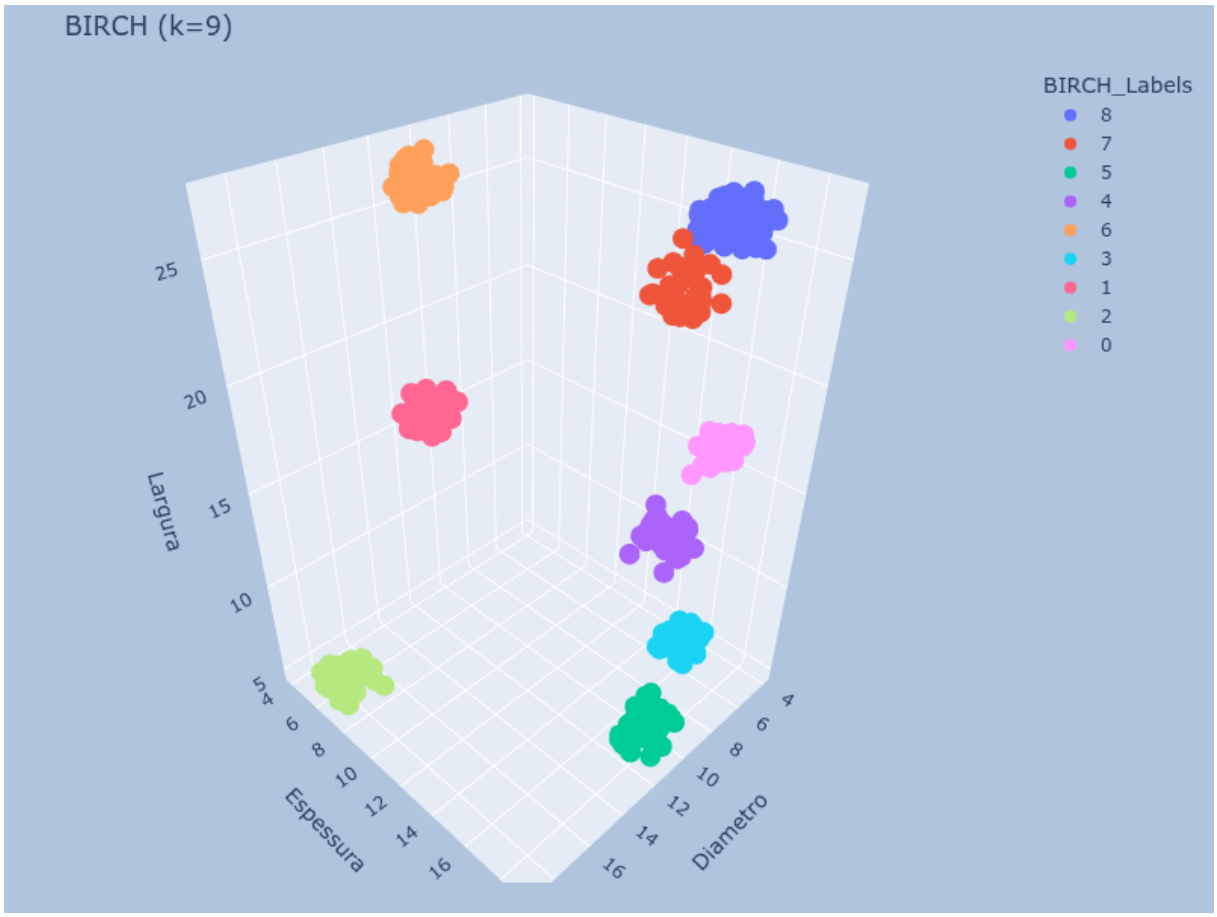
[Abrir versao interativa (p2_dbscan.html)]

d) Affinity Propagation



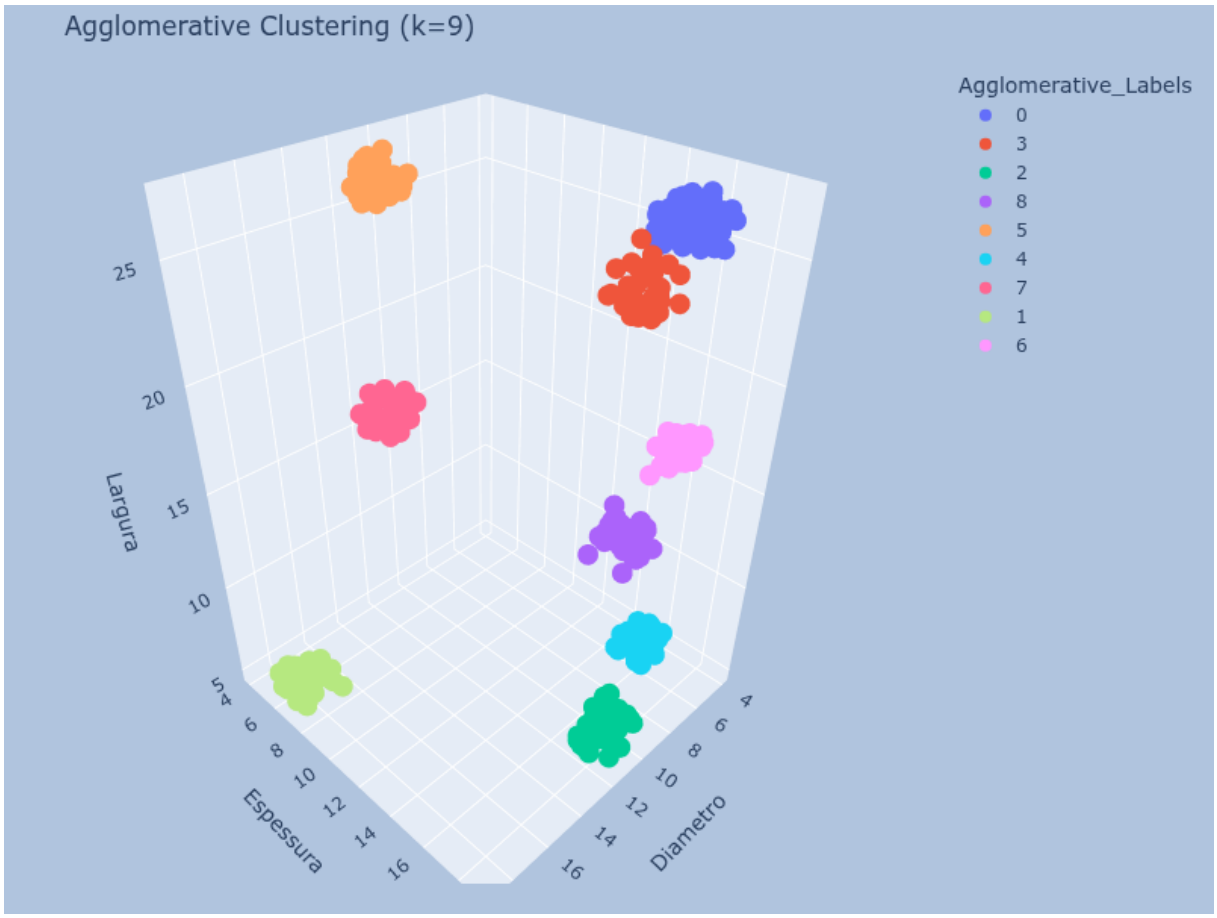
Abrir versao interativa (p2_affinity.html)

e) BIRCH



[Abrir versao interativa (p2_birch.html)]

f) Agglomerative Clustering



[Abrir versao interativa (p2_agglomerative.html)]

Anexo: Codigo Fonte (resolver_trabalho.py)

```
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.cluster import (
    KMeans,
    MiniBatchKMeans,
    DBSCAN,
    AffinityPropagation,
    Birch,
    AgglomerativeClustering,
)

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

# Ignorar avisos para uma saída mais limpa
warnings.filterwarnings("ignore")

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


def plotar_grafico_3d(df, labels_col, title, output_basename):
    """
    Função auxiliar para criar e salvar gráficos 3D interativos (HTML)
    e estáticos (PNG).
    """
    print(f"  Gerando gráfico 3D: {title}...")
    fig = px.scatter_3d(
        df, x="Diametro", y="Espessura", z="Largura", color=labels_col, title=title
    )

    # Ajusta o layout como no exemplo
    fig.update_layout(
        margin=dict(l=20, r=20, t=40, b=20), paper_bgcolor="LightSteelBlue"
    )

    # Salva como HTML interativo
    html_file_path = f"{output_basename}.html"
    fig.write_html(html_file_path)

    # Salva como PNG estático (requer 'kaleido' instalado)
    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 resolver_problema_1(input_dir, output_dir):
    """
```

```
Resolve a Questão 1: Agrupamento01.txt com K-Means.
"""
print("Iniciando Problema 1 (Agrupamento01.txt)...")

file_path = os.path.join(input_dir, "Agrupamento01.txt")
try:
    dados = pd.read_csv(file_path)
except FileNotFoundError:
    print(f"ERRO: Arquivo não encontrado em {file_path}")
    return

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

print("  Calculando WSS para o Método do Cotovelo...")
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)
    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)")
plt.title("Método do Cotovelo (Elbow Method) - Agrupamento01")
plt.grid(True)
cotovelo_path = os.path.join(output_dir, "p1_grafico_cotovelo.png")
plt.savefig(cotovelo_path)
plt.close()
print(f"  Gráfico do Cotovelo salvo em: {cotovelo_path}")

k_ideal = 5

print(f"  Executando K-Means com k={k_ideal}...")
kmeans = KMeans(n_clusters=k_ideal, random_state=42, n_init=10)
y_km = kmeans.fit_predict(X)
centers = kmeans.cluster_centers_

# Salvar centros em TXT para o PDF
centros_df = pd.DataFrame(
    centers, columns=X.columns, index=[f"Centro {i+1}" for i in range(k_ideal)]
)
centros_path = os.path.join(output_dir, "p1_centros.txt")
with open(centros_path, "w", encoding="utf-8") as f:
    f.write(centros_df.to_string(line_width=80))
print(f"  Centros salvos em: {centros_path}")

plt.figure(figsize=(12, 8))
plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=y_km, s=50, cmap="viridis", alpha=0.7)
plt.scatter(
    centers[:, 0],
    centers[:, 1],
    c="red",
    s=250,
    marker="X",
    alpha=0.9,
    label="Centróides",
)
```

```
    plt.title(f"Resultado K-Means (k={k_ideal}) - Agrupamento01")
    plt.xlabel(X.columns[0])
    plt.ylabel(X.columns[1])
    plt.legend()
    plt.grid(True)
    resultado_path = os.path.join(output_dir, "p1_kmeans_resultado.png")
    plt.savefig(resultado_path)
    plt.close()
    print(f"  Gráfico do resultado K-Means salvo em: {resultado_path}")
    print("Problema 1 concluído.\n")


def resolver_problema_2(input_dir, output_dir):
    """
    Resolve a Questão 2: Agrupamento02.txt com 6 algoritmos diferentes.
    """
    print("Iniciando Problema 2 (Agrupamento02.txt)...")

    file_path = os.path.join(input_dir, "Agrupamento02.txt")
    try:
        df = pd.read_csv(file_path)
    except FileNotFoundError:
        print(f"ERRO: Arquivo não encontrado em {file_path}")
        return

    df_plot = df.iloc[:, 0:3].copy()

    print("  Normalizando dados com StandardScaler...")
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(df_plot)

    print("  Calculando WSS para o Método do Cotovelo (Agrupamento02)...")
    wss_p2 = []
    K_range_p2 = range(1, 11)

    for k in K_range_p2:
        kmeans_elbow = KMeans(n_clusters=k, random_state=42, n_init=10)
        kmeans_elbow.fit(X_scaled)
        wss_p2.append(kmeans_elbow.inertia_)

    plt.figure(figsize=(10, 6))
    plt.plot(K_range_p2, wss_p2, "bo-")
    plt.xlabel("Número de Clusters (k)")
    plt.ylabel("WSS (Inertia)")
    plt.title("Método do Cotovelo (Elbow Method) - Agrupamento02")
    plt.grid(True)
    cotovelo_path_p2 = os.path.join(output_dir, "p2_grafico_cotovelo.png")
    plt.savefig(cotovelo_path_p2)
    plt.close()
    print(f"  Gráfico do Cotovelo salvo em: {cotovelo_path_p2}")

    k_ideal_p2 = 9  # [cite: 50, 758]

    # --- a) K-Means ---
    print("\nExecutando (a) K-Means...")
    kmeans = KMeans(n_clusters=k_ideal_p2, random_state=0, n_init=10)  # [cite: 50, 758]
    labels_km = kmeans.fit_predict(X_scaled)  # [cite: 51]
    df_plot["KMeans_Labels"] = labels_km.astype(str)  # [cite: 54]

    # Salvar centros (K-Means) em TXT
```

```
centers_km_scaled = kmeans.cluster_centers_
centers_km_original = scaler.inverse_transform(centers_km_scaled)
centros_km_df = pd.DataFrame(
    centers_km_original,
    columns=df_plot.columns[:3],
    index=[f"Centro {i+1}" for i in range(k_ideal_p2)],
)
centros_km_path = os.path.join(output_dir, "p2_centros_kmeans.txt")
# Força a formatação com largura máxima de 100 caracteres para o PDF
with open(centros_km_path, "w", encoding="utf-8") as f:
    f.write(centros_km_df.to_string(line_width=80))
print(f"  Centros K-Means salvos em: {centros_km_path}")

plotar_grafico_3d(
    df_plot,
    "KMeans_Labels",
    f"K-Means (k={k_ideal_p2})",
    os.path.join(output_dir, "p2_kmeans"),
)

# --- b) Mini Batch K-Means ---
print("\nExecutando (b) Mini Batch K-Means...")
mbk = MiniBatchKMeans(n_clusters=k_ideal_p2, random_state=42, n_init=10)
labels_mbk = mbk.fit_predict(X_scaled)
df_plot["MiniBatchKMeans_Labels"] = labels_mbk.astype(str)
plotar_grafico_3d(
    df_plot,
    "MiniBatchKMeans_Labels",
    f"Mini Batch K-Means (k={k_ideal_p2})",
    os.path.join(output_dir, "p2_minibatch_kmeans"),
)

# --- c) DBSCAN ---
print("\nExecutando (c) DBSCAN...")
dbscan = DBSCAN(eps=0.5, min_samples=10)  # Valores ajustados
labels_db = dbscan.fit_predict(X_scaled)
df_plot["DBSCAN_Labels"] = labels_db.astype(str)
n_clusters_db = len(set(labels_db)) - (1 if -1 in labels_db else 0)
print(f"  DBSCAN encontrou {n_clusters_db} clusters (Rótulo '-1' é ruído)")
plotar_grafico_3d(
    df_plot,
    "DBSCAN_Labels",
    f"DBSCAN (Encontrou {n_clusters_db} clusters)",
    os.path.join(output_dir, "p2_dbscan"),
)

# --- d) Affinity Propagation ---
print("\nExecutando (d) Affinity Propagation...")
af = AffinityPropagation(random_state=42, damping=0.9)
labels_af = af.fit_predict(X_scaled)
df_plot["Affinity_Labels"] = labels_af.astype(str)
n_clusters_af = len(af.cluster_centers_indices_)
print(f"  Affinity Propagation encontrou {n_clusters_af} clusters")
plotar_grafico_3d(
    df_plot,
    "Affinity_Labels",
    f"Affinity Propagation (Encontrou {n_clusters_af} clusters)",
    os.path.join(output_dir, "p2_affinity"),
)
```

```
    # --- e) BIRCH ---
    print("\nExecutando (e) BIRCH...")
    birch = Birch(n_clusters=k_ideal_p2)
    labels_birch = birch.fit_predict(X_scaled)
    df_plot["BIRCH_Labels"] = labels_birch.astype(str)
    plotar_grafico_3d(
        df_plot,
        "BIRCH_Labels",
        f"BIRCH (k={k_ideal_p2})",
        os.path.join(output_dir, "p2_birch"),
    )

    # --- f) Agglomerative Clustering ---
    print("\nExecutando (f) Agglomerative Clustering...")
    agg = AgglomerativeClustering(n_clusters=k_ideal_p2)
    labels_agg = agg.fit_predict(X_scaled)
    df_plot["Agglomerative_Labels"] = labels_agg.astype(str)
    plotar_grafico_3d(
        df_plot,
        "Agglomerative_Labels",
        f"Agglomerative Clustering (k={k_ideal_p2})",
        os.path.join(output_dir, "p2_agglomerative"),
    )

    print("\nProblema 2 concluído.")


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

    print(f"Iniciando trabalho prático...")
    print(f"Arquivos de entrada serão lidos de: {os.path.abspath(INPUT_DIR)}")
    print(f"Gráficos e resultados serão salvos em: {os.path.abspath(OUTPUT_DIR)}\n")

    resolver_problema_1(INPUT_DIR, OUTPUT_DIR)
    resolver_problema_2(INPUT_DIR, OUTPUT_DIR)

    print("\n--- Processo Concluído ---")
    print(
        "Todos os gráficos (PNG e HTML) e centros (TXT) foram salvos no diretório './result/'."
    )
```