Trabalho 2: Análise e Clusterização de Chegadas de Turistas Internacionais no Brasil

Bruna Matias de Lima
Departamento de Computação
Universidade Federal de São Carlos
São Carlos, Brasil
RA:820582

Larissa Dias da Silva
Departamento de Computação
Universidade Federal de São Carlos
São Carlos, Brasil
RA: 800204

Professor: Alan Demétrius Baria Valejo
Disciplina: Aprendizado de Máquina I

I. Introdução

Este relatório tem como objetivo analisar o conjunto de dados de chegadas de turistas internacionais ao Brasil em 2024, explorando padrões e características desses visitantes. Serão aplicadas técnicas de pré-processamento, análise exploratória de dados (EDA) e algoritmos de agrupamento (clustering) para identificar segmentos distintos de perfis de chegada. A motivação para este trabalho consiste na importância estratégica do turismo para a economia, e como o entendimento dos dados de chegada pode impactar decisões para esse setor.

Neste trabalho, utilizamos o conjunto de dados "Estimativas de Chegadas de Turistas Internacionais ao Brasil de 2024", disponível em https://dados.gov.br/dados/conjuntos-dados/estimativas-de-chegadas-de-turistas-internacionais-ao-brasil.

O projeto foi desenvolvido no Google Colab utilizando a linguagem Python, com o uso das bibliotecas `Pandas`, `Matplotlib`, `Seaborn`, `Plotly Express` e `Scikit-learn`. O notebook desenvolvido pode ser visto em: https://colab.research.google.com/drive/1Q55e1OlQwZ0r_YS__-fCfedYKcdxcTLW?usp=sharing.

II. Metodologia

A. Bases de Dados

O conjunto de dados principal utilizado é o "Estimativas de Chegadas de Turistas Internacionais ao Brasil de 2024". Este dataset contém informações relevantes sobre o fluxo de turistas, incluindo:

- `Continente`: Continente de origem do turista.
- `cod continente`: Código do continente.
- `País`: País de origem do turista.
- `cod pais`: Código do país.
- `UF`: Unidade da Federação de chegada no Brasil.
- `cod uf`: Código da UF.
- `Via`: Meio de transporte utilizado (Aérea, Terrestre, Fluvial, Marítima).
- `cod via`: Código da via.
- `ano`: Ano da chegada (2024).
- `Mês`: Mês da chegada.
- `cod mes`: Código do mês.
- `Chegadas`: Número de chegadas de turistas.

B. Ferramentas

Para a implementação e análise, foram utilizadas as seguintes bibliotecas Python:

- `Pandas`: Manipulação e análise de dados.
- `Matplotlib`: Geração de gráficos estáticos.
- `Seaborn`: Visualização estatística de dados.
- `Plotly Express`: Criação de gráficos interativos.
- `Scikit-learn`: Ferramentas para machine learning, incluindo:
  - `StandardScaler`: Padronização de dados.
  - `MinMaxScaler`: Normalização de dados.
  - `KMeans`: Algoritmo de agrupamento (clustering).
  - `PCA` (Principal Component Analysis): Redução de dimensionalidade.
  - `DBSCAN` (Density-Based Spatial Clustering of Applications with Noise): Algoritmo de agrupamento baseado em densidade.
  - `NearestNeighbors`: Para auxiliar na determinação de parâmetros do DBSCAN.

C. Problema Escolhido

O problema abordado é a identificação de grupos (clusters) de Unidades Federativas (UFs) e de padrões nas chegadas de turistas internacionais, baseando-se em características como continente de origem, via de entrada, e sazonalidade. Isso permite entender melhor os diferentes perfis de chegada e pode ser útil para o planejamento turístico.

D. Critérios de Avaliação

Para a avaliação dos algoritmos de clustering, foram utilizados os seguintes critérios:

- **Método do Cotovelo (Elbow Method)**: Utilizado para determinar o número ótimo de clusters (`k`) para o algoritmo K-Means.
- **Gráfico de K-Distância**: Utilizado para auxiliar na escolha do parâmetro `eps` para o algoritmo DBSCAN, identificando o "cotovelo" na curva de distâncias.

E. Algoritmos Selecionados

Foram selecionados dois algoritmos de clustering para este trabalho:

- **K-Means**: Um algoritmo de clustering não supervisionado que particiona os dados em `k` grupos, onde cada ponto de dados pertence ao cluster cujo centro (centróide) é o mais próximo.
- **DBSCAN**: Um algoritmo de agrupamento baseado em densidade que descobre clusters de formato arbitrário em um espaço de dados. Ele identifica "pontos de ruído" como outliers.

III. Análise dos Resultados

A. Pré-processamento

1) Conhecendo os Dados: Inicialmente, realizamos uma inspeção dos dados para entender sua estrutura, tipos de variáveis e algumas linhas de amostra. O dataset consiste em 22856 entradas e 12 colunas, como 'Continente', 'País', 'UF', 'Via', 'ano', 'Mês', 'Chegadas', entre outras. Os tipos de dados foram verificados, revelando que a maioria das colunas é do tipo 'object' (categórica) ou 'int64' (numérica).

2) Limpeza dos Dados: **Verificação de Valores Nulos e Duplicados:** Verificamos a presença de valores nulos em todas as colunas do dataset, e foi constatado que não há valores ausentes.

```
data.isnull().sum()
```

Listing 1. Verificação de Valores Nulos

Para identificar e tratar linhas duplicadas, contamos o número de linhas completamente duplicadas. Foram encontradas 1640 linhas duplicadas. Para lidar com essa questão, agrupamos as instâncias com dados iguais e somamos o atributo 'Chegadas', garantindo que não houvesse mais duplicatas.

```
data.duplicated().sum()

group_cols = [col for col in data.columns
    if col != 'Chegadas']
data = data.groupby(group_cols, as_index=
    False)['Chegadas'].sum()

data.duplicated().sum()
```

Listing 2. Tratamento de Linhas Duplicadas

Busca de Ruídos: Para identificar a presença de outliers (ruídos) na variável 'Chegadas', utilizamos um boxplot.

No cenário de turismo, outliers já são esperados devido à alta sazonalidade em determinadas épocas do ano, representando oportunidades de lucro para o setor. Por essa razão, optamos por manter esses dados.

B. Análise Exploratória dos Dados

1) Criação de Novas Variáveis para Explorar a Sazonalidade: Para capturar padrões sazonais no dataset, criamos novas variáveis de tempo a partir das colunas 'ano' e 'cod mes'. As novas variáveis incluem 'Data' (formato datetime), 'mes', 'trimestre', 'semestre' e 'estacao'.

```
data['ano'] = data['ano'].astype(str)
data['cod mes'] = data['cod mes'].astype(
    str).str.zfill(2)

data['Data'] = pd.to_datetime(data['ano']
    + '-' + data['cod mes'] + '-01', format
    ='%Y-%m-%d')

# Criar vari veis sazonais
data['mes'] = data['Data'].dt.month
data['trimestre'] = data['Data'].dt.
    quarter
data['semestre'] = data['mes'].apply(
    lambda x: 1 if x <= 6 else 2)
data['ano_mes'] = data['Data'].dt.
    to_period('M').astype(str)

# Vari vel de esta  o do ano
def estacao(m):
    if m in [12, 1, 2]:
        return 'ver o'
    elif m in [3, 4, 5]:
        return 'outono'
    elif m in [6, 7, 8]:
        return 'inverno'
    else:
        return 'primavera'

data['estacao'] = data['mes'].apply(
    estacao)
```

Listing 3. Criação de Variáveis Sazonais

As estatísticas descritivas das variáveis numéricas foram analisadas, mostrando a distribuição e a centralidade dos dados.

2) Gráficos: **Chegadas por Localidades:** Analisamos o total de chegadas por continente para entender as principais regiões de origem dos turistas.

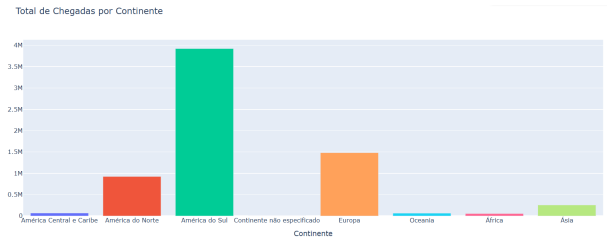

Fig. 1. Total de chegadas por continente.

Em seguida, investigamos os 4 principais países em número de chegadas para identificar as fontes mais importantes de turistas.

Por fim, exploramos o total de chegadas por Unidade da Federação (UF) para verificar quais estados recebem mais turistas.

Formas de Chegadas: A evolução das chegadas por via de entrada (Aérea, Terrestre, Fluvial, Marítima) ao longo do tempo foi analisada para identificar tendências e sazonalidades.

Também analisamos os 10 estados com mais chegadas por via de entrada para entender as preferências de transporte em cada região.

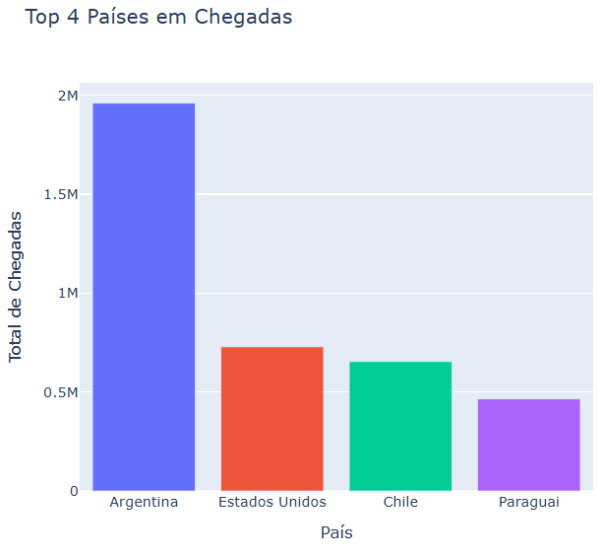

Fig. 2. 4 países com maiores chegadas

Fig. 3. Total de chegadas por via de entrada

Sazonalidade das Chegadas: A evolução mensal das chegadas foi examinada para identificar picos e vales ao longo do ano.

O total de chegadas por trimestre foi visualizado para uma perspectiva de sazonalidade em períodos maiores.

E o total de chegadas por estação do ano também foi avaliado.

C. Seleção de Variáveis

Para entender as relações entre as variáveis numéricas, calculamos e visualizamos a matriz de correlação.

```
num_cols = ['cod continente', 'cod pais',
    'cod uf', 'cod via', 'cod mes', '
    trimestre', 'semestre', 'Chegadas']
corr_matrix = data[num_cols].corr()
plt.figure(figsize=(10, 8))
```

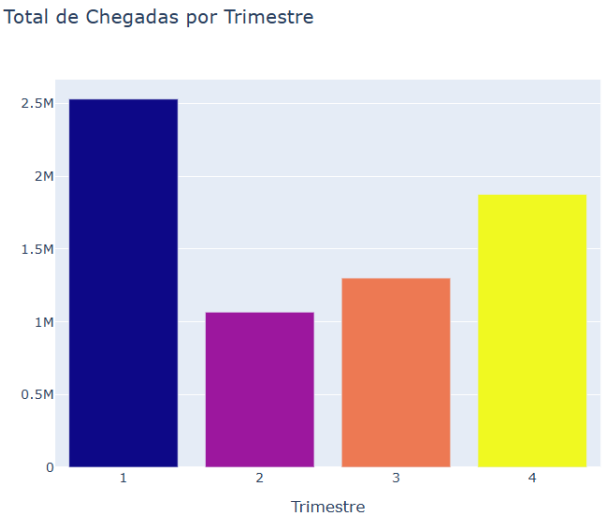

Fig. 4. Total de chegadas por trimestre

```
sns.heatmap(corr_matrix, annot=True, cmap=
    'Blues', fmt=".2f")
plt.title("Matriz de Correla  o entre
    Vari veis Num ricas")
plt.show()
```

Listing 4. Matriz de Correlação

Agrupamos os dados por 'UF' e adicionamos a via e a estação mais comum, além do continente. Isso foi feito para preparar o dataset para a clusterização das UFs.

```
df_agg = data.groupby('UF').agg({
    'Chegadas': 'sum',
    'Via': lambda x: x.value_counts().
        idxmax(),
    'estacao': lambda x: x.value_counts().
        idxmax(),
    'Continente': lambda x: x.value_counts
        ().idxmax()
}).reset_index()
```

Listing 5. Agregação de Dados por UF

Removemos a coluna 'semestre' por ser redundante, uma vez que 'trimestre' e 'mes' já capturam a sazonalidade.

```
data = data.drop(columns=['semestre'])
```

Listing 6. Remoção de Coluna Redundante

Codificação de Variáveis Categóricas: Identificamos as colunas categóricas e aplicamos a codificação one-hot encoding com 'pd.get_dummies' para transformá-las em formato numérico, adequado para algoritmos de machine learning.

```
cat_cols = data.select_dtypes(include='
    object').columns
data_encoded = pd.get_dummies(data,
    columns=cat_cols, drop_first=True)
data_encoded.head()
```

Listing 7. Codificação One-Hot Encoding

Normalização dos Dados: Utilizamos 'StandardScaler' para padronizar as features, garantindo que todas as variáveis numéricas tenham média zero e variância unitária, o que é crucial para o bom desempenho de algoritmos baseados em distância.

```
from sklearn.preprocessing import
    StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_encoded
    .drop('UF', axis=1))
```

Listing 8. Normalização com StandardScaler

D. Aplicação do K-Means

Para determinar o número ótimo de clusters (k) para o K-Means, empregamos o Método do Cotovelo, que analisa a soma dos quadrados das distâncias dos pontos aos seus respectivos centróides (WCSS) para diferentes valores de k.

```
from sklearn.cluster import KMeans

wcss = []
K = range(1, 10)
for k in K:
    kmeans = KMeans(n_clusters=k,
        random_state=42, n_init='auto')
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

plt.plot(K, wcss, 'bo-')
plt.xlabel('N mero de Clusters (k)')
plt.ylabel('WCSS')
plt.title('M todo do Cotovelo')
plt.show()
```

Listing 9. Método do Cotovelo para K-Means

Com base no gráfico do Método do Cotovelo, o valor de k=4 foi selecionado como o número ideal de clusters. Em seguida, aplicamos o algoritmo K-Means com este valor de k e adicionamos os rótulos de cluster ao DataFrame.

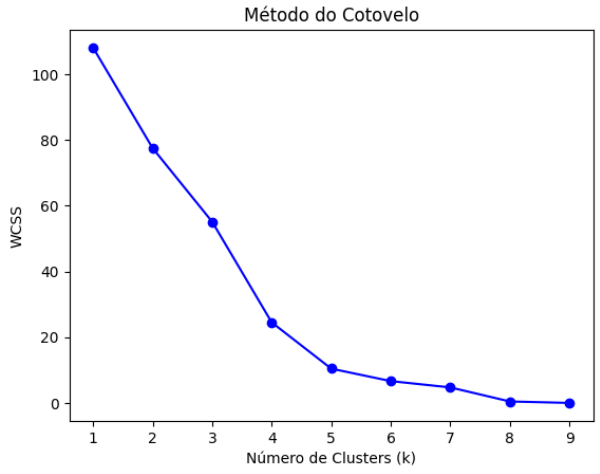

Fig. 5. Método do cotovelo

```
kmeans = KMeans(n_clusters=4, random_state
    =42, n_init='auto')
df_encoded['cluster'] = kmeans.fit_predict
    (X_scaled)
```

Listing 10. Aplicação do K-Means

Os resultados dos clusters, com as respectivas UFs, foram exibidos, mostrando a distribuição das unidades federativas entre os 4 grupos identificados.

Para visualizar os clusters e suas relações, utilizamos o PCA para reduzir a dimensionalidade dos dados para 3D e plotar um gráfico de dispersão 3D, colorindo os pontos pelos clusters.

```
from sklearn.decomposition import PCA
import plotly.express as px

pca_3d = PCA(n_components=3)
X_pca_3d = pca_3d.fit_transform(X_scaled)

df_pca_3d = pd.DataFrame({
    'UF': df_encoded['UF'],
    'Cluster': df_encoded['cluster'],
    'PCA1': X_pca_3d[:, 0],
    'PCA2': X_pca_3d[:, 1],
    'PCA3': X_pca_3d[:, 2]
})

fig = px.scatter_3d(
    df_pca_3d,
    x='PCA1',
    y='PCA2',
    z='PCA3',
    color='Cluster',
    hover_name='UF',
    title='Clusteriza o das UFs em 3D (
        K-Means + PCA)',
    color_continuous_scale='Viridis'
)

fig.update_layout(scene=dict(
    xaxis_title='PCA 1',
    yaxis_title='PCA 2',
    zaxis_title='PCA 3'
))
fig.show()
```

Listing 11. Visualização 3D dos Clusters (K-Means + PCA)

Analisamos a distribuição da via de entrada por cluster, para entender a predominância de cada modal em cada grupo de UFs.

```
import seaborn as sns
import matplotlib.pyplot as plt

df_clusters = df_encoded.join(df_agg[['Via
    ', 'estacao', 'Continente']])

plt.figure(figsize=(10, 5))
sns.countplot(data=df_clusters, x='Via',
    hue='cluster', palette='Set2')
plt.title('Distribui o da Via de
    Entrada por Cluster')
plt.xlabel('Via de Entrada')
plt.ylabel('N mero de UFs')
```

```
plt.show()
```

Listing 12. Distribuição da Via de Entrada por Cluster

As UFs foram agrupadas em 4 clusters, com as seguintes características observadas:

- **Cluster 0**: Difere do Cluster 3 pelo volume de chegadas, forma de chegada e alta sazonalidade no verão, por ter mais estados litorâneos (6 de 11).
- **Cluster 1**: Não possui fronteira terrestre (exceto Pará, com poucos turistas por essa via), e as chegadas são exclusivamente aéreas.
- **Cluster 2**: Amapá, com características específicas.
- **Cluster 3**: São Paulo, que recebe a maior quantidade de turistas, exclusivamente por via aérea.

Essa divisão pode refletir fatores geográficos, logísticos e econômicos, sugerindo um uso estratégico para o planejamento turístico regional.

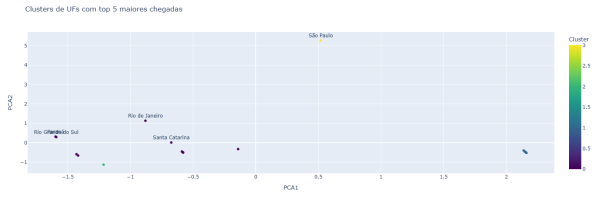

Fig. 6. Agrupamento K-means

E. Análise com DBSCAN

Para aplicar o DBSCAN, primeiro, removemos a coluna 'Data', pois não é uma feature numérica adequada para o algoritmo. Em seguida, normalizamos os dados restantes utilizando 'MinMaxScaler', que escala os valores para um intervalo entre 0 e 1.

```
from sklearn.preprocessing import
    MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.cluster import DBSCAN
from sklearn.neighbors import
    NearestNeighbors
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

if 'Data' in data_encoded.columns:
    data_final_for_dbscan = data_encoded.
        drop(columns=['Data'])
else:
    data_final_for_dbscan = data_encoded.
        copy()

scaler_dbscan = MinMaxScaler()
data_normalized_dbscan = scaler_dbscan.
    fit_transform(data_final_for_dbscan)
data_normalized_dbscan = pd.DataFrame(
    data_normalized_dbscan, columns=
    data_final_for_dbscan.columns)

n_components_pca_dbscan = 2
pca_dbscan = PCA(n_components=
    n_components_pca_dbscan)
X_pca_dbscan = pca_dbscan.fit_transform(
    data_normalized_dbscan)
```

Listing 13. Normalização de Dados para DBSCAN

Para determinar o valor ótimo de 'eps' (raio de vizinhança) para o DBSCAN, utilizamos o gráfico de k-distância. Observando o "cotovelo" na curva, onde a inclinação muda abruptamente, um valor de 'eps = 0.03' foi selecionado como ideal, com 'min_samples = 8'.

```
min_samples_val_dbscan = 8

neighbors_dbscan = NearestNeighbors(
    n_neighbors=min_samples_val_dbscan)
neighbors_fit_dbscan = neighbors_dbscan.
    fit(X_pca_dbscan)
distances_dbscan, indices_dbscan =
    neighbors_fit_dbscan.kneighbors(
    X_pca_dbscan)

distances_dbscan = np.sort(
    distances_dbscan[:,
    min_samples_val_dbscan-1], axis=0)
plt.figure(figsize=(12, 6))
plt.plot(distances_dbscan)
plt.xlabel("Pontos de Dados Ordenados por
    Dist ncia")
plt.ylabel(f"Dist ncia ao {
    min_samples_val_dbscan}- simo  Vizinho
    Mais Pr ximo (epsilon)")
plt.title("Gr fico de K-Dist ncia para
    Determinar 'eps' (para DBSCAN no PCA)")
plt.grid(True)
plt.show()
```

Listing 14. Gráfico de K-Distância para DBSCAN

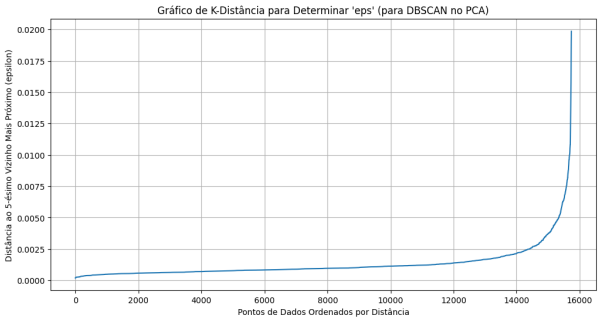

Fig. 7. Gráfico de K-Distância

Foi realizado **agrupamento DBSCAN** nos dados previamente processados com PCA (X_{pca_dbscan}), utilizando $eps = 0.03$ e $min\_samples = 7$ para a identificação dos clusters. Posteriormente, os rótulos de cluster foram adicionados ao DataFrame PCA para facilitar a visualização e, por fim, a contagem de pontos em cada cluster foi apresentada para análise da distribuição dos agrupamentos.

```
eps_val_tuned = 0.03
min_samples_val_dbscan = 7
```

```
3
4  dbscan = DBSCAN(eps=eps_val_tuned,
       min_samples=min_samples_val_dbscan)
5  clusters_dbscan = dbscan.fit_predict(
       X_pca_dbscan)
6
7  df_pca_dbscan = pd.DataFrame(X_pca_dbscan,
       columns=['PCA1', 'PCA2'])
8  df_pca_dbscan['Cluster'] = clusters_dbscan
9
10 print("--- RESULTADOS DO DBSCAN ---")
11 print("Contagem de pontos por cluster (
       ap s PCA e ajuste de 'eps'):")
12 print(df_pca_dbscan['Cluster'].
       value_counts())
```

Listing 15. Aplicação do DBSCAN com parâmetros ajustados

O gráfico subsequente ilustra os clusters gerados pelo DBSCAN. Cada cor distinta no gráfico demarca um cluster encontrado, com pontos de coloração idêntica pertencendo ao mesmo agrupamento. A cor correspondente ao cluster -1 (indicador de ruído) revela os pontos que o algoritmo não conseguiu agrupar. Caso o gráfico persista em mostrar uma elevada proporção de pontos como ruído (apenas a cor -1 sendo dominante) ou a formação de um único cluster excessivamente grande, isso denota a necessidade de recalibrar o parâmetro eps_val_tuned com base no gráfico de k-distância, ou de diminuir o valor de $min\_samples\_val\_dbscan$, ou, ainda, que a estrutura de densidade dos dados não é adequada para o DBSCAN.

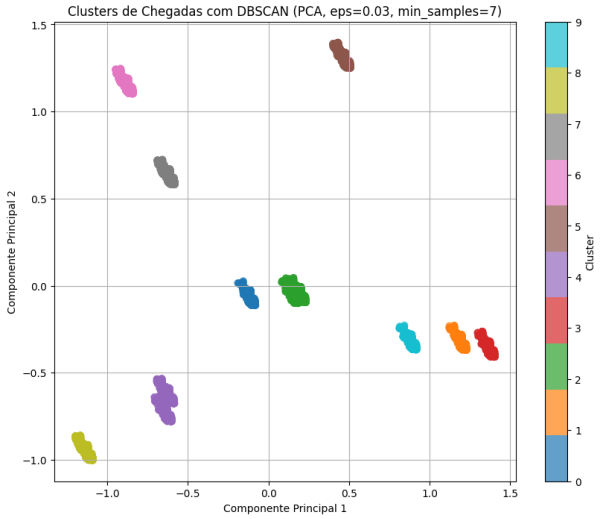

Fig. 8. Gráfico dos clusters de chegada com DBSCan

Além disso, também analisamos as quantidades de pontos em cada cluster, cada Cluster têm no mínimo 1100 pontos.

Cluster	count
4	2497
2	2320
5	1547
8	1501
6	1489
7	1443
3	1442
1	1233
9	1146
0	1121

A análise paramétrica do DBSCAN, focada na velocidade e utilizando um range de ϵ de 0.02 a 0.15 e $min\_samples$ de 7 a 11, indicou um **Melhor Silhouette Score de 0.8594** com os parâmetros $(eps, min\_samples) = (0.03, 7)$. Este resultado sugere uma boa separação e coesão entre os clusters encontrados, uma vez que o Silhouette Score está próximo de 1.

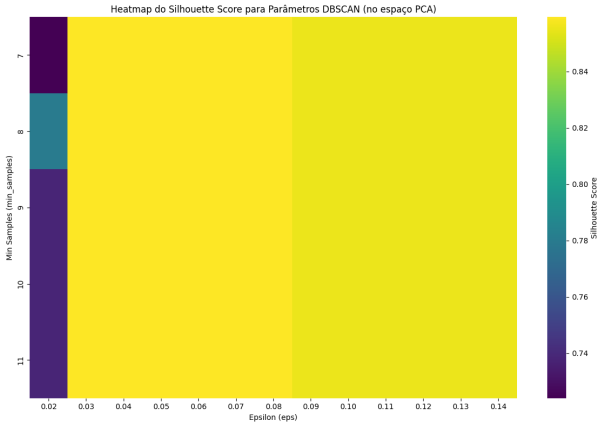

Fig. 9. Heatmap do Silhouette Score

A análise subjetiva dos clusters DBSCAN (excluindo pontos de ruído, que neste caso foram 0) revelou as seguintes características para cada grupo:

- **Cluster 0:** Este cluster possui um Total de Chegadas de 332.474 (Média por registro: 296.59). A Via de Entrada mais comum é Aérea. A Estação do Ano mais frequente para chegadas é o inverno. O Continente de origem mais comum é a Europa. Inclui chegadas de 92 países distintos, com os principais sendo: Argentina, Estados Unidos, França. Este cluster contém 1121 registros.
- **Cluster 1:** Este cluster possui um Total de Chegadas de 508.738 (Média por registro: 412.60). A Via de Entrada mais comum é Aérea. A Estação do Ano mais frequente para chegadas é a primavera. O Continente de origem mais comum é a Europa. Inclui chegadas de 92 países distintos, com os principais sendo: Alemanha, França, Argentina. Este cluster contém 1233 registros.
- **Cluster 2:** Este cluster possui um Total de Chegadas de 855.043 (Média por registro: 368.55). A Via de Entrada mais comum é Aérea. A Estação do Ano mais frequente para chegadas é o inverno. O Continente de origem mais

comum é a Europa. Inclui chegadas de 93 países distintos, com os principais sendo: Argentina, Colômbia, França. Este cluster contém 2320 registros.

- **Cluster 3:** Este cluster possui um Total de Chegadas de 560.732 (Média por registro: 388.86). A Via de Entrada mais comum é Aérea. A Estação do Ano mais frequente para chegadas é a primavera. O Continente de origem mais comum é a Europa. Inclui chegadas de 92 países distintos, com os principais sendo: Alemanha, Estados Unidos, Argentina. Este cluster contém 1442 registros.
- **Cluster 4:** Este cluster possui um Total de Chegadas de 734.239 (Média por registro: 294.05). A Via de Entrada mais comum é Aérea. A Estação do Ano mais frequente para chegadas é o outono. O Continente de origem mais comum é a Europa. Inclui chegadas de 92 países distintos, com os principais sendo: Argentina, França, Espanha. Este cluster contém 2497 registros.
- **Cluster 5:** Este cluster possui um Total de Chegadas de 806.478 (Média por registro: 521.32). A Via de Entrada mais comum é Aérea. A Estação do Ano mais frequente para chegadas é o verão. O Continente de origem mais comum é a Europa. Inclui chegadas de 93 países distintos, com os principais sendo: Argentina, Estados Unidos, Peru. Este cluster contém 1547 registros.
- **Cluster 6:** Este cluster possui um Total de Chegadas de 833.306 (Média por registro: 559.64). A Via de Entrada mais comum é Aérea. A Estação do Ano mais frequente para chegadas é o verão. O Continente de origem mais comum é a Europa. Inclui chegadas de 93 países distintos, com os principais sendo: Argentina, Alemanha, Itália. Este cluster contém 1489 registros.
- **Cluster 7:** Este cluster possui um Total de Chegadas de 956.737 (Média por registro: 663.02). A Via de Entrada mais comum é Aérea. A Estação do Ano mais frequente para chegadas é o verão. O Continente de origem mais comum é a Europa. Inclui chegadas de 92 países distintos, com os principais sendo: Alemanha, Espanha, França. Este cluster contém 1443 registros.
- **Cluster 8:** Este cluster possui um Total de Chegadas de 740.483 (Média por registro: 493.33). A Via de Entrada mais comum é Aérea. A Estação do Ano mais frequente para chegadas é o outono. O Continente de origem mais comum é a Europa. Inclui chegadas de 92 países distintos, com os principais sendo: Alemanha, Itália, França. Este cluster contém 1501 registros.
- **Cluster 9:** Este cluster possui um Total de Chegadas de 445.389 (Média por registro: 388.65). A Via de Entrada mais comum é Aérea. A Estação do Ano mais frequente para chegadas é a primavera. O Continente de origem mais comum é a Europa. Inclui chegadas de 93 países distintos, com os principais sendo: França, Peru, Argentina. Este cluster contém 1146 registros.

Todos os clusters formados são compostos majoritariamente por chegadas via aérea e têm a Europa como continente de origem mais comum. A principal distinção entre eles parece estar na estação do ano de maior ocorrência de chegadas (inverno, primavera, outono ou verão) e no volume total de chegadas. A ausência de pontos de ruído (Cluster -1) indica que o algoritmo DBSCAN conseguiu agrupar todos os pontos de dados, o que é um bom sinal para a densidade e conectividade dos dados com os parâmetros escolhidos.

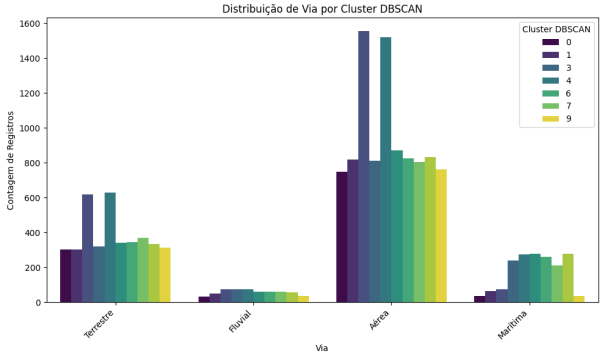

Fig. 10. Via por Cluster

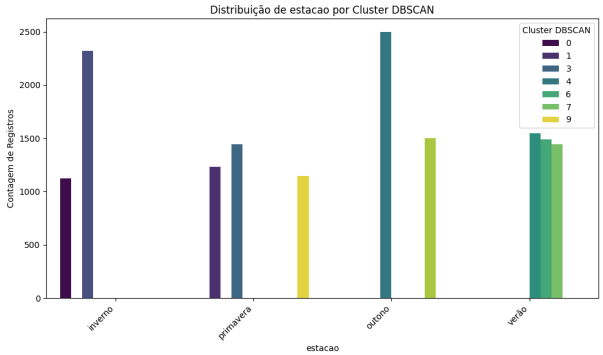

Fig. 11. Estação por Cluster

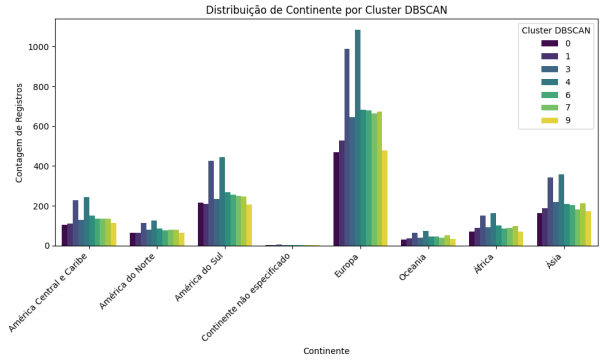

Fig. 12. Continente por Cluster

IV. Conclusões

O trabalho de agrupamento dos dados de chegadas de turistas, utilizando técnicas de pré-processamento, análise exploratória e os algoritmos K-Means e DBSCAN, revelou

insights valiosos sobre os padrões de fluxo turístico. O pré-processamento e a limpeza garantiram a qualidade dos dados, com a decisão de manter outliers devido à sua relevância para a sazonalidade no setor turístico. A análise exploratória confirmou a predominância da América do Sul e de estados como São Paulo e Rio de Janeiro nas chegadas, majoritariamente por via aérea, evidenciando a forte influência da sazonalidade.

O K-Means identificou 4 clusters principais, segmentando os estados por volume de chegadas e vias de acesso, destacando o papel singular de São Paulo. Complementarmente, o DBSCAN, com parâmetros ajustados ($\epsilon = 0.03$, $min\_samples = 7$), resultou em 10 clusters densos, sem pontos de ruído (Silhouette Score de 0.8594). Esses clusters refinam a segmentação, mostrando que, embora a via aérea e a Europa sejam predominantes, há variações significativas na sazonalidade e nos países de origem entre os grupos.

Em suma, a segmentação dos turistas em clusters distintos, seja em uma visão mais ampla pelo K-Means ou mais detalhada pelo DBSCAN, oferece uma base sólida para estratégias personalizadas de marketing e planejamento turístico. Compreender essas nuances nos padrões de chegada é fundamental para otimizar recursos e maximizar o potencial econômico do setor, permitindo ações mais direcionadas e eficazes para cada perfil de turista e região.

V. Vídeo

Vídeo: https://youtu.be/WfGv9rFTnjI

References

[1] MINISTÉRIO DO TURISMO. Estimativas de Chegadas de Turistas Internacionais ao Brasil de 2024. Dados.gov.br, [s.d.]. Disponível em: https://dados.gov.br/dados/conjuntos-dados/estimativas-de-chegadas-de-turistas-internacionais-ao-brasil. Acesso em: 16 jul. 2025.