

# High Value Customer Identification

## 1. Objective

Develop a **clustering model to identify high-value customers** in an e-commerce database, objectively defining who should be part of the **Insiders Program**. As a result, the project should:

- Generate a list of customers eligible for the program in the format:

customer_id	is_insider
1314	1

- Provide an analytical report answering the main business questions:

1. **Who are the eligible customers** to participate in the Insiders program?
2. **How many customers** will be part of the group?
3. **What are the main characteristics** of these customers?
4. **What percentage of revenue contribution** comes from the insiders?
5. **What is the revenue expectation** for this group in the upcoming months?
6. **What are the eligibility criteria** for someone to join the Insiders group?
7. **What are the conditions for someone to be removed** from the Insiders group?
8. **What guarantees that the Insiders program performs better** than the rest of the customer base?
9. **What actions can the marketing team take to increase** revenue?

## 2. Executive Summary

This project aimed to objectively and data-driven identify the **highest-value customers in an e-commerce business** and define who should compose the **Insiders Program**, enabling smarter strategies for retention, loyalty, and revenue growth.

Through the application of a **GMM (Euclidean) clustering model**, with excellent segmentation quality demonstrated by a **Silhouette Score of 0.7048**, it was possible to divide the customer base into **6 clusters with clearly distinct behavioral and financial profiles**. Each group presented specific patterns of purchasing frequency, engagement, operational risk, and revenue contribution, allowing for a deep strategic understanding of the customer base.

Among the generated clusters, **Cluster 1** clearly stood out as the group with the greatest strategic value. This group consists of:

- **419 customers (14.8% of the base)**
- Responsible for approximately **62.3% of total revenue**
- Highest gross revenue, number of orders, and purchase volume
- **High purchase frequency and low recency**
- **Longest customer lifetime**
- **Low return rate**

- **Highest monetary\_per\_day** in the base

These customers purchase more frequently, for a longer period of time, spend more, and generate lower operational risk, naturally becoming the business **Insiders**. Revenue projections based on current behavior indicate a potential of **over R\$ 1 million in 30 days** and approximately **R\$ 3 million in 90 days**, demonstrating their relevance not only historically, but also in the future.

Beyond identifying the VIP group, the project delivered:

- **Clear eligibility criteria** to join the Insider Program (high value, high recurrence, consistent engagement, and low risk)
- **Objective removal criteria**, ensuring quality control of the program (decreasing frequency, increasing recency, revenue decline, or rising return rates)
- Analytical evidence that the Insider group is **significantly superior to the rest of the base**
- A strategic action plan for each cluster, balancing:
  - VIP retention
  - Acceleration of promising customers
  - Recovery of customers at churn risk
  - Risk control for problematic customers

In summary, this project delivers not only an analytical model, but a **business decision tool**, capable of:

- Prioritizing the customers that truly matter
- Directing marketing investments more efficiently
- Increasing revenue and retention
- Reducing waste and risk
- Supporting strategic decisions with statistical evidence

The final result proves that the application of **Data Science and Machine Learning** is essential to transform customer management, making the loyalty program smarter, more profitable, and truly value-driven.

## 3. About the Data

The dataset comes from an international e-commerce platform and contains real purchase records made by customers over time. Each row represents a transaction and includes information such as invoice number, purchased product, quantity, price, purchase date, customer ID, and country.

With this data, it is possible to understand customer behavior: how much they buy, how often they return, how long they remain active, and how much they contribute to revenue. This enables customer value analysis, segmentation, and the development of smarter business strategies.

## 4. Getting Started

### 4.1 Importing Libraries

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from sklearn.metrics import silhouette_score, silhouette_samples
```

```
import umap.umap_ as umap
from sklearn.preprocessing import MinMaxScaler
import ydata_profiling
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.ensemble import RandomForestRegressor
from sklearn.mixture import GaussianMixture
```

## 4.2 Style Import

```
In [2]: plt.style.use('../styles/personalestilo.mplstyle')
```

Bad key axes.color\_cycle in file ../styles/personalestilo.mplstyle, line 9 ('axes.color\_cycle: df691b, 5cb85c, 5bc0de, f0ad4e, d9534f, 4e5d6c')  
You probably need to get an updated matplotlibrc file from  
<https://github.com/matplotlib/matplotlib/blob/v3.10.0/lib/matplotlib/mpl-data/matplotlibrc>  
or from the matplotlib source distribution

## 4.3 Additional Functions

```
In [ ]: def snake_case(lst):
    def convert(s):
        s = s.replace(' ', '_')
        new_s = ""
        for i, c in enumerate(s):
            if c.isupper():
                if i > 0 and (s[i-1].islower() or (i+1 < len(s) and s[i+1].islower())):
                    new_s += "_"
                new_s += c.lower()
            else:
                new_s += c
        return new_s
    return [convert(s) for s in lst]

def get_cluster_profile(df, cluster_col='cluster', id_col='customer_id'):
    total_customers = df[id_col].nunique()
    feature_cols = [col for col in df.columns if col not in [id_col, cluster_col]]
    agg_dict = {
        id_col: 'nunique'
    }
    for col in feature_cols:
        if pd.api.types.is_numeric_dtype(df[col]):
            agg_dict[col] = 'mean'
    cluster_profile = df.groupby(cluster_col).agg(agg_dict).reset_index()
    cluster_profile = cluster_profile.rename(columns={id_col: 'num_customers'})
    cluster_profile['perc_customers'] = (cluster_profile['num_customers'] / total_customers * 100)
    display(cluster_profile)
    return cluster_profile
```

## 4.4 Loading the Data

```
In [ ]: df_raw = pd.read_csv('../data/ecommerce.csv', encoding='latin1').drop(columns=["Unnamed: 8"])
```

## 5. Data Description

```
In [5]: df1 = df_raw.copy()
```

```
In [6]: df1.head()
```

Out[6]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	29-Nov-16	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	29-Nov-16	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	29-Nov-16	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	29-Nov-16	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	29-Nov-16	3.39	17850.0	United Kingdom

## 5.1 Renaming Columns

In [7]: `df1.columns = snake_case(df1.columns)`

## 5.2 Data Dimensions

In [8]: `print(f'Dimensões do DataFrame: {df1.shape[0]} linhas, {df1.shape[1]} colunas')`

Dimensões do DataFrame: 541909 linhas, 8 colunas

## 5.3 Data Types

In [9]: `df1.dtypes`

Out[9]:

```

invoice_no      object
stock_code      object
description      object
quantity        int64
invoice_date     object
unit_price      float64
customer_id     float64
country         object
dtype: object

```

In [10]: `df1['invoice_date'] = pd.to_datetime(df1['invoice_date'], format='%d-%b-%y')`  
`df1['customer_id'] = df1['customer_id'].astype('Int64')`

## 5.4 Checking and Filling NaN Values

In [11]: `print('Valores ausentes por coluna:')`  
`print(df1.isna().sum())`

```

Valores ausentes por coluna:
invoice_no      0
stock_code      0
description    1454
quantity        0
invoice_date     0
unit_price      0
customer_id    135080
country         0
dtype: int64

```

```
In [12]: mask_nan_cust = df1['customer_id'].isna()

invoice_groups = df1.loc[mask_nan_cust, 'invoice_no'].unique()

invoice_to_custid = {inv: cid for cid, inv in enumerate(invoice_groups, start=19000)}

df1.loc[mask_nan_cust, 'customer_id'] = df1.loc[mask_nan_cust, 'invoice_no'].map(invoice_to_custid)

df1['customer_id'] = df1['customer_id'].astype('Int64')

df1 = df1[df1['customer_id'] != 16446]

df1 = df1.drop('description', axis=1)

df1 = df1[~df1['country'].isin(['European Community', 'Unspecified'])]

df1 = df1[df1['unit_price'] >= 0.04].copy()

df1 = df1[~df1['stock_code'].str.fullmatch(r'[A-Za-z]+')]

df1_returns = df1[df1['quantity'] < 0].copy()
df1_purchase = df1[df1['quantity'] > 0].copy()
```

```
In [13]: print('Valores ausentes por columna:')
print(df1.isna().sum())
```

```
Valores ausentes por columna:
invoice_no      0
stock_code      0
quantity        0
invoice_date    0
unit_price      0
customer_id     0
country         0
dtype: int64
```

## 5.5 Descriptive Statistics

```
In [14]: df_numeric = df1.select_dtypes(include=['int64', 'float64', 'Int64'])
df_categoric = df1.select_dtypes(include=['object'])
```

### 5.5.1 Numerical Attributes

```
In [15]: numerical_stats = pd.DataFrame({
    'mean': df_numeric.mean(),
    'median': df_numeric.median(),
    'std': df_numeric.std(),
    'min': df_numeric.min(),
    'max': df_numeric.max(),
    'range': df_numeric.max() - df_numeric.min(),
    'skew': df_numeric.skew(),
    'kurtosis': df_numeric.kurt()
})

display(numerical_stats)
```

	mean	median	std	min	max	range	skew	kurtosis
<b>quantity</b>	9.905463	3.0	148.429964	-74215.0	74215.0	148430.0	0.317835	233158.512277
<b>unit_price</b>	3.311659	2.08	5.344376	0.04	1050.15	1050.11	52.769587	6973.649182
<b>customer_id</b>	16672.382766	16241.0	2897.685605	12346.0	22709.0	10363.0	0.50036	-0.777262

## 6. Feature Engineering

```
In [16]: df2 = df1.copy()
```

## 6.1 Variable Derivation

```
In [17]: df_ref = df2[['customer_id']].drop_duplicates().reset_index(drop=True)
```

### 6.1.1 Gross Revenue

```
In [ ]: gross_revenue = (  
    df1_purchase.groupby('customer_id')  
        .apply(lambda x: (x['quantity'] * x['unit_price']).sum())  
        .reset_index(name='gross_revenue')  
    )  
  
df_ref = df_ref.merge(gross_revenue, on='customer_id', how='left')
```

C:\Users\Patryck\AppData\Local\Temp\ipykernel\_5492\4049961655.py:5: FutureWarning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass `include\_groups=False` to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.  
.apply(lambda x: (x['quantity'] \* x['unit\_price']).sum())

### 6.1.2 Recency

```
In [19]: max_date = df1_purchase['invoice_date'].max()  
recency = (  
    df1_purchase.groupby('customer_id')['invoice_date']  
        .max()  
        .reset_index()  
    )  
recency['recency'] = (max_date - recency['invoice_date']).dt.days  
recency = recency[['customer_id', 'recency']]  
df_ref = df_ref.merge(recency, on='customer_id', how='left')
```

### 6.1.3 Quantity of Products Purchased

```
In [ ]: qtd_produtos = (  
    df1_purchase.groupby('customer_id')  
        .apply(lambda x: x['quantity'].sum())  
        .reset_index(name='qtd_produtos')  
    )  
  
df_ref = df_ref.merge(  
    qtd_produtos[['customer_id', 'qtd_produtos']],  
    on='customer_id',  
    how='left'  
    )
```

C:\Users\Patryck\AppData\Local\Temp\ipykernel\_5492\499747772.py:4: FutureWarning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass `include\_groups=False` to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.  
.apply(lambda x: x['quantity'].sum())

### 6.1.4 Quantity of Purchases

```
In [21]: quantidade_compras = (  
    df1_purchase.groupby('customer_id')['invoice_no']  
        .nunique()  
        .reset_index(name='orders_count')  
    )  
df_ref = df_ref.merge(quantidade_compras, on='customer_id', how='left')
```

### 6.1.5 Frequency

```
In [22]: frequency = ( df1_purchase .sort_values(['customer_id', 'invoice_date']) .drop_duplicates(subset=['customer_id', 'invoice_date'])
df_ref = df_ref.merge(frequency, on='customer_id', how='left')
```

### 6.1.6 Number of Returns

```
In [23]: returns = (
    df1_returns
        .groupby('customer_id')['quantity']
        .sum()
        .abs()
        .reset_index(name='returns')
    )

df_ref = df_ref.merge(returns, on='customer_id', how='left')
df_ref['returns'] = df_ref['returns'].fillna(0)

df_ref['returns_rate'] = df_ref['returns'] / df_ref['qtd_produtos']
```

### 6.1.7 Monetary Per Day

```
In [ ]: lifetime = (
    df1_purchase
        .sort_values(['customer_id', 'invoice_date'])
        .groupby('customer_id')['invoice_date']
        .agg(['min', 'max'])
        .reset_index()
    )

lifetime['lifetime_days'] = (lifetime['max'] - lifetime['min']).dt.days + 1

df_ref = df_ref.merge(lifetime[['customer_id', 'lifetime_days']], on='customer_id', how='left')

df_ref['lifetime_days'] = df_ref['lifetime_days'].replace([np.inf, -np.inf], 0).fillna(0)

df_ref['monetary_per_day'] = (
    df_ref['gross_revenue'] / df_ref['lifetime_days']
)

df_ref['monetary_per_day'] = df_ref['monetary_per_day'].replace([np.inf, -np.inf], 0).fillna(0)

In [25]: df_ref = df_ref.dropna()
```

## 7. Exploratory Data Analysis

```
In [26]: df3 = df_ref.copy()
```

### 7.1 Univariate Analysis

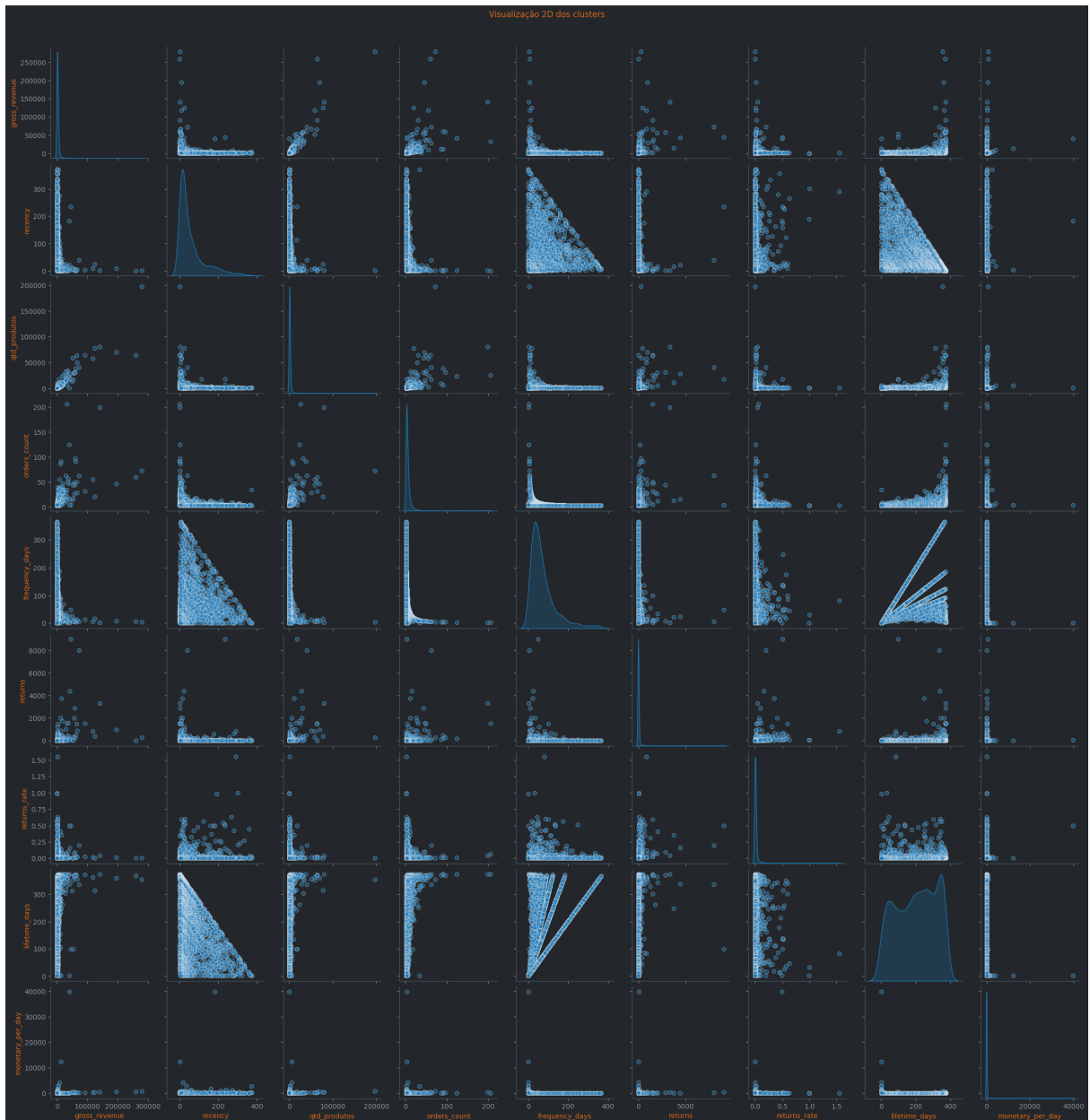
```
In [ ]: profile = ydata_profiling.ProfileReport(df3, title='Pandas Profiling Report', explorative=True)
profile.to_file('../pdf/profile_report.html')
```

```
Summarize dataset:  0%|          | 0/5 [00:00<?, ?it/s]
100%|██████████| 10/10 [00:00<00:00, 212.76it/s]
Generate report structure:  0%|          | 0/1 [00:00<?, ?it/s]
Render HTML:  0%|          | 0/1 [00:00<?, ?it/s]
Export report to file:  0%|          | 0/1 [00:00<?, ?it/s]
```

### 7.2 Bivariate Analysis

```
In [28]: df_viz = df3.drop('customer_id', axis=1)
```

```
sns.pairplot(df_viz, diag_kind='kde', plot_kws={'alpha':0.5})
plt.suptitle('Visualização 2D dos clusters', y=1.02)
plt.show()
```



## 7.3 Space Study

```
In [29]: df3v = df3.drop('customer_id', axis=1)
```

```
In [30]: df3v.columns
```

```
Out[30]: Index(['gross_revenue', 'recency', 'qtd_produtos', 'orders_count',
               'frequency_days', 'returns', 'returns_rate', 'lifetime_days',
               'monetary_per_day'],
              dtype='object')
```

```
In [31]: features_to_scale = [
          'gross_revenue', 'recency', 'qtd_produtos',
          'orders_count', 'frequency_days', 'returns', 'returns_rate', 'lifetime_days', 'monetary_per
        ]

scaler = MinMaxScaler()
df3v_scaled = df3v.copy()
df3v_scaled[features_to_scale] = scaler.fit_transform(df3v[features_to_scale])
```

### 7.3.1 PCA

```
In [ ]: n_components = df3v.shape[1]
pca = PCA(n_components=n_components)
principal_components = pca.fit_transform(df3v_scaled)

pc_columns = [f'PC{i+1}' for i in range(n_components)]
df_pca = pd.DataFrame(data=principal_components, columns=pc_columns)

fig, axes = plt.subplots(1, 2, figsize=(16, 6))

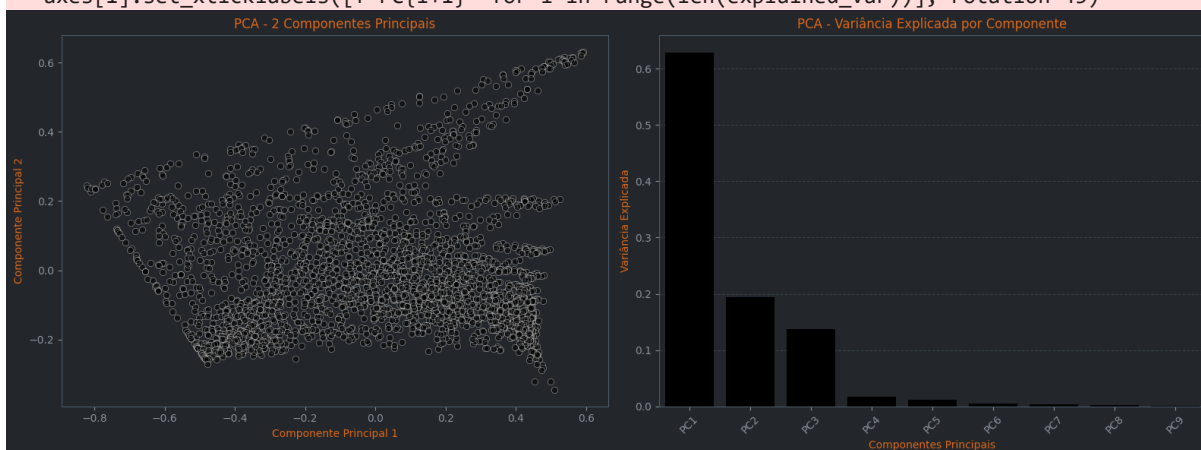
sns.scatterplot(x='PC1', y='PC2', data=df_pca, alpha=0.5, color='black', ax=axes[0])
axes[0].set_title('PCA - 2 Componentes Principais')
axes[0].set_xlabel('Componente Principal 1')
axes[0].set_ylabel('Componente Principal 2')

explained_var = pca.explained_variance_ratio_
sns.barplot(x=[f'PC{i+1}' for i in range(len(explained_var))], y=explained_var, color='black',
axes[1].set_ylabel('Variância Explicada')
axes[1].set_xlabel('Componentes Principais')
axes[1].set_title('PCA - Variância Explicada por Componente')
axes[1].set_xticklabels([f'PC{i+1}' for i in range(len(explained_var))], rotation=45)
axes[1].grid(axis='y', linestyle='--', alpha=0.5)

plt.tight_layout()
plt.show()
```

C:\Users\Patryck\AppData\Local\Temp\ipykernel\_5492\421556761.py:25: UserWarning: set\_xticklabels() should only be used with a fixed number of ticks, i.e. after set\_ticks() or using a FixedLocator.

```
axes[1].set_xticklabels([f'PC{i+1}' for i in range(len(explained_var))], rotation=45)
```



### 7.3.2 UMAP

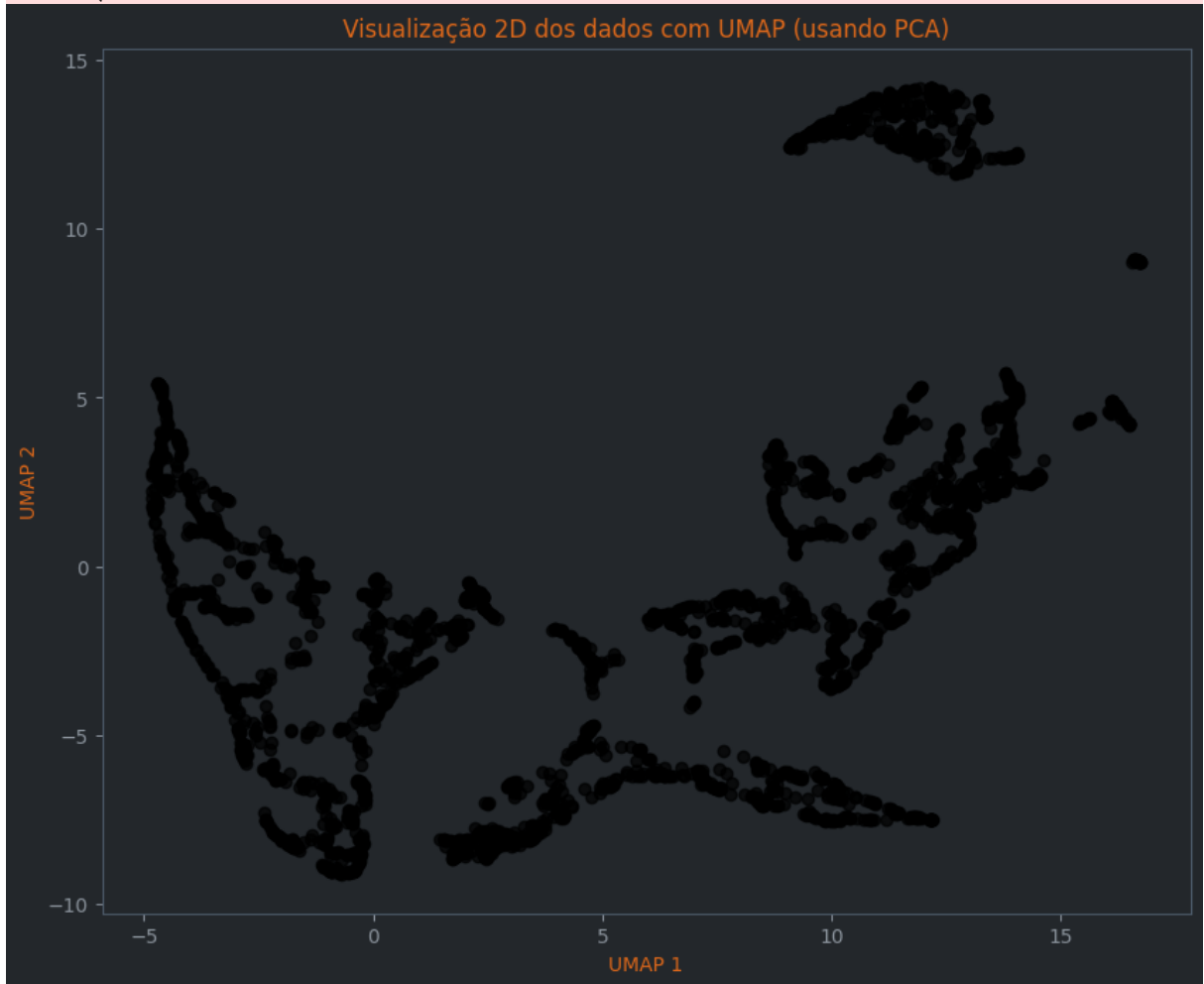
```
In [33]: reducer = umap.UMAP(random_state=42)

umap_features = df_pca.copy()
embedding = reducer.fit_transform(umap_features)

plt.figure(figsize=(10, 8))
plt.scatter(
    embedding[:, 0],
    embedding[:, 1],
    color='black',
    alpha=0.7
)
plt.title('Visualização 2D dos dados com UMAP (usando PCA)')
plt.xlabel('UMAP 1')
plt.ylabel('UMAP 2')
plt.show()
```

```
c:\Users\Patryck\AppData\Local\Programs\Python\Python311\Lib\site-packages\umap\umap_.py:1952:
UserWarning: n_jobs value 1 overridden to 1 by setting random_state. Use no seed for parallelis
m.
```

```
warn(
```

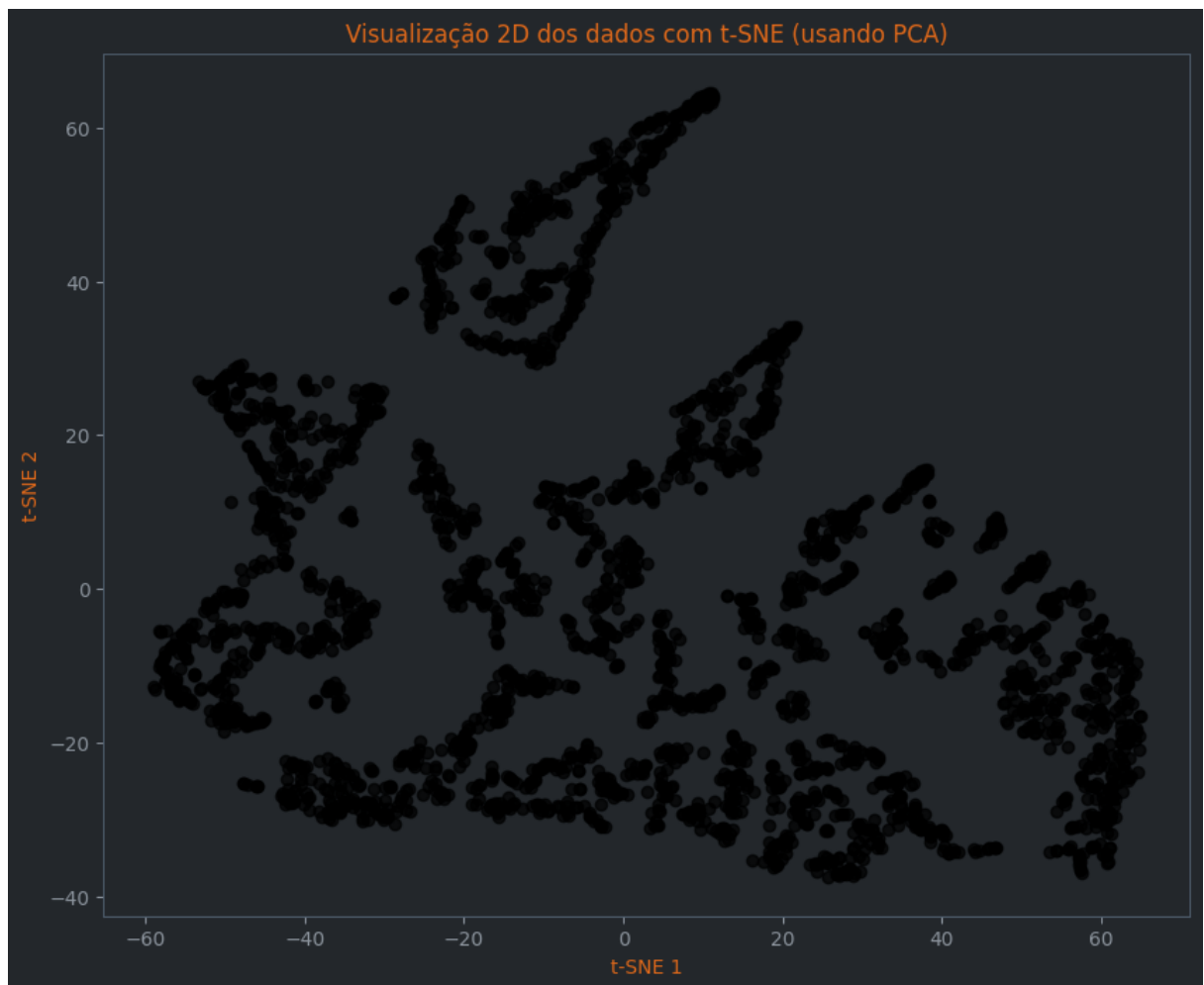


### 7.3.3 T-SNE

```
In [34]: reducer = TSNE(random_state=42, n_components=2)

tsne_features = df_pca.copy()
embedding = reducer.fit_transform(tsne_features)

plt.figure(figsize=(10, 8))
plt.scatter(
    embedding[:, 0],
    embedding[:, 1],
    color='black',
    alpha=0.7
)
plt.title('Visualização 2D dos dados com t-SNE (usando PCA)')
plt.xlabel('t-SNE 1')
plt.ylabel('t-SNE 2')
plt.show()
```



### 7.3.4 Tree-Based Embedding

```
In [35]: X = df3v.drop(columns=['gross_revenue'])
y = df3v['gross_revenue']

rf = RandomForestRegressor(n_estimators=300, random_state=42)
rf.fit(X, y)

leaf_indices = rf.apply(X)

embedding_tree = pd.DataFrame(
    leaf_indices,
    index=df3v.index,
    columns=[f'leaf_{i}' for i in range(leaf_indices.shape[1])]
)

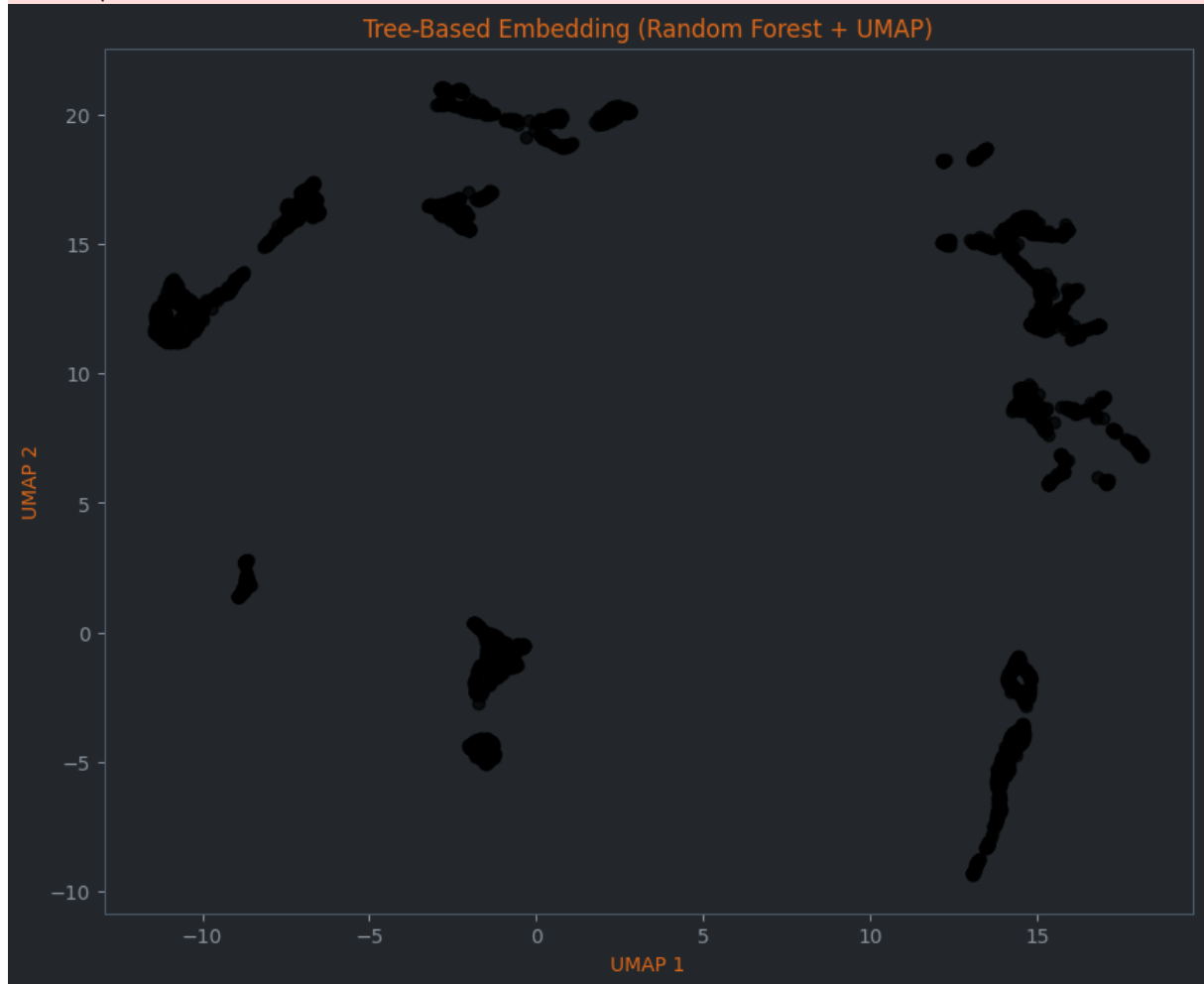
reducer = umap.UMAP(n_components=2, random_state=42)
tree_2d_umap_array = reducer.fit_transform(embedding_tree)

tree_2d_umap = pd.DataFrame(
    tree_2d_umap_array,
    index=df3v.index,
    columns=['UMAP_1', 'UMAP_2']
)

fig, ax = plt.subplots(figsize=(10,8))
scatter = ax.scatter(tree_2d_umap['UMAP_1'], tree_2d_umap['UMAP_2'], alpha=0.7, color='black')
ax.set_title('Tree-Based Embedding (Random Forest + UMAP)')
ax.set_xlabel('UMAP 1')
ax.set_ylabel('UMAP 2')
plt.show()
plt.close(fig)
```

```
c:\Users\Patryck\AppData\Local\Programs\Python\Python311\Lib\site-packages\umap\umap_.py:1952:
UserWarning: n_jobs value 1 overridden to 1 by setting random_state. Use no seed for parallelis
m.
```

```
warn(
```



## 8. Fine Tuning

```
In [36]: df4 = tree_2d_umap.copy()
```

```
In [37]: X = df4.copy()
```

### 8.1 K-MEANS

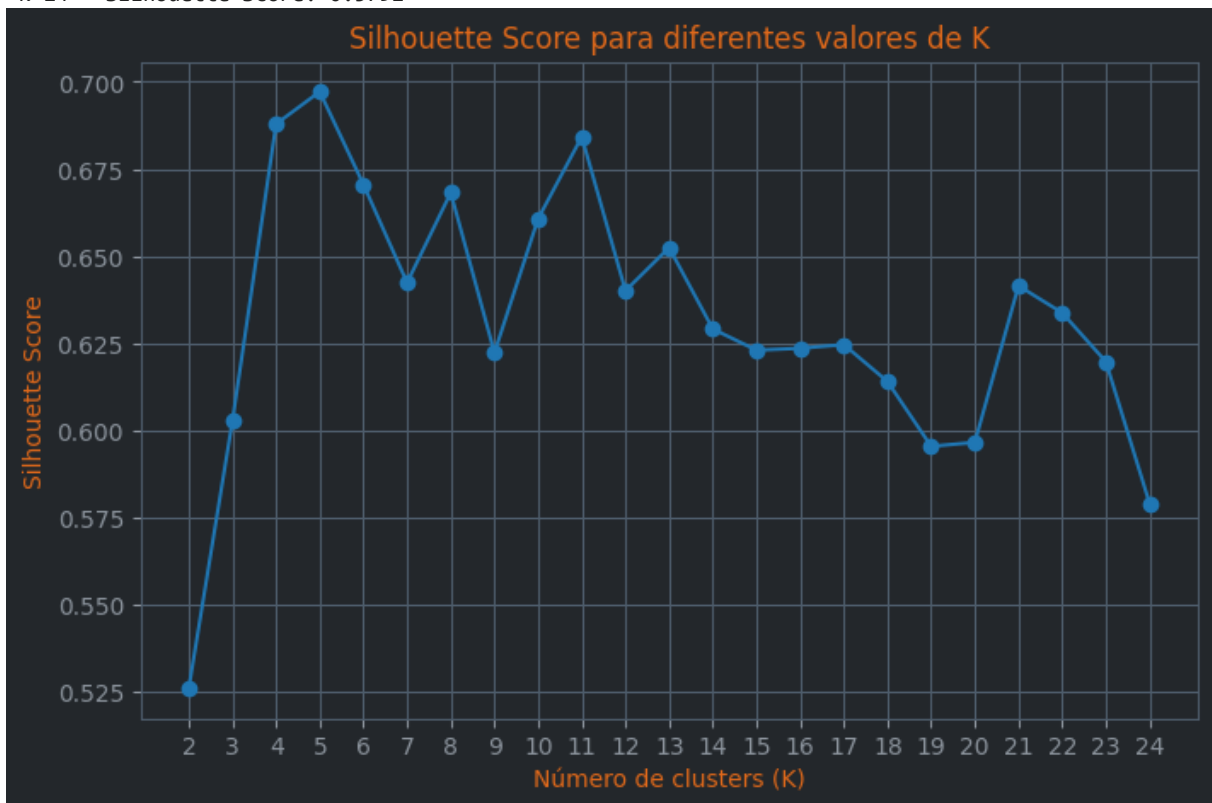
```
In [ ]: range_n_clusters = range(2, 25)
kmeans_silhouette_scores = []

for n_clusters in range_n_clusters:
    kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=10, max_iter=300, init='rand
    labels = kmeans.fit_predict(X)
    sil_score = silhouette_score(X, labels)
    kmeans_silhouette_scores.append(sil_score)
    print(f"K={n_clusters} - Silhouette Score: {sil_score:.4f}")

plt.figure(figsize=(8,5))
plt.plot(range_n_clusters, kmeans_silhouette_scores, marker='o')
plt.title('Silhouette Score para diferentes valores de K')
plt.xlabel('Número de clusters (K)')
plt.ylabel('Silhouette Score')
plt.xticks(range_n_clusters)
plt.grid(True)
plt.show()
```

```
best_k = range_n_clusters[kmeans_silhouette_scores.index(max(kmeans_silhouette_scores))]
print(f"\nMelhor K de acordo com o Silhouette Score: {best_k} (Score: {max(kmeans_silhouette_sc
```

```
K=2 - Silhouette Score: 0.5260
K=3 - Silhouette Score: 0.6027
K=4 - Silhouette Score: 0.6879
K=5 - Silhouette Score: 0.6972
K=6 - Silhouette Score: 0.6706
K=7 - Silhouette Score: 0.6424
K=8 - Silhouette Score: 0.6682
K=9 - Silhouette Score: 0.6223
K=10 - Silhouette Score: 0.6607
K=11 - Silhouette Score: 0.6842
K=12 - Silhouette Score: 0.6401
K=13 - Silhouette Score: 0.6525
K=14 - Silhouette Score: 0.6291
K=15 - Silhouette Score: 0.6231
K=16 - Silhouette Score: 0.6236
K=17 - Silhouette Score: 0.6247
K=18 - Silhouette Score: 0.6141
K=19 - Silhouette Score: 0.5955
K=20 - Silhouette Score: 0.5966
K=21 - Silhouette Score: 0.6415
K=22 - Silhouette Score: 0.6338
K=23 - Silhouette Score: 0.6198
K=24 - Silhouette Score: 0.5792
```



Melhor K de acordo com o Silhouette Score: 5 (Score: 0.6972)

## 8.2 GMM

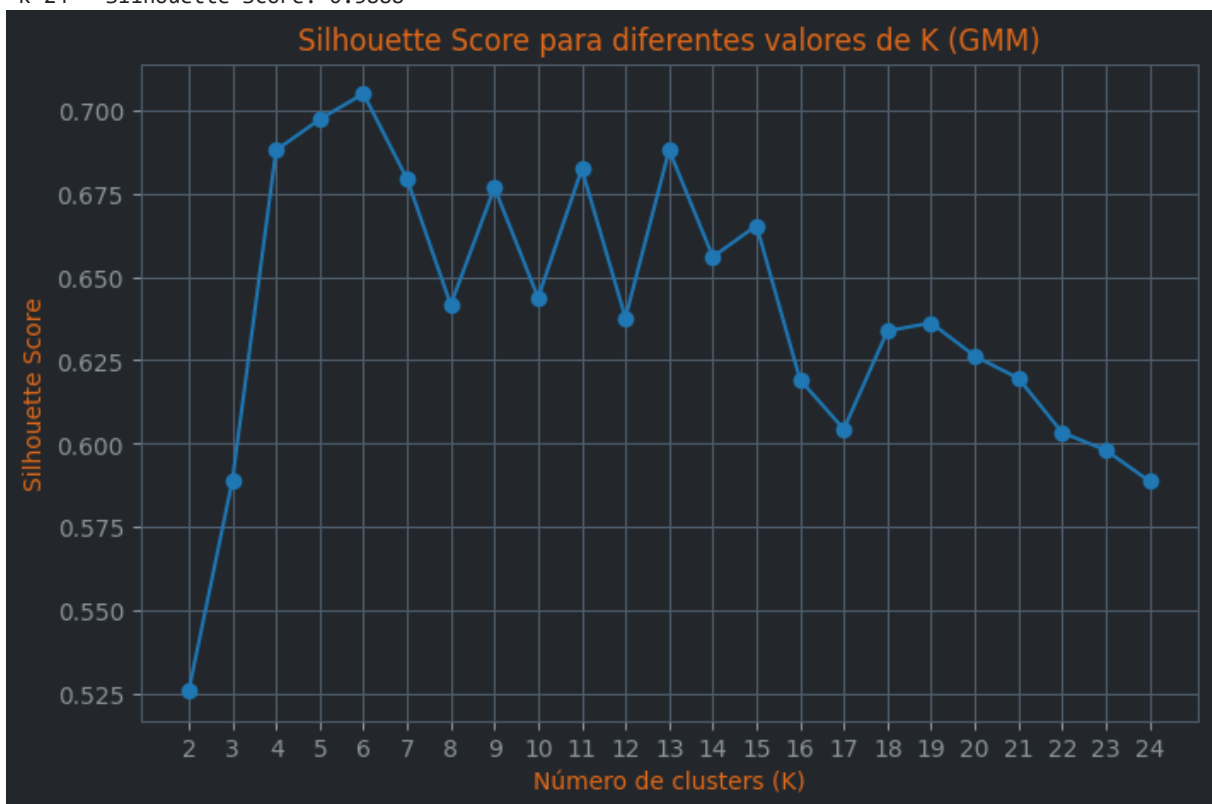
```
In [ ]: range_n_clusters = range(2, 25)
gmm_silhouette_scores = []

for n_clusters in range_n_clusters:
    gmm = GaussianMixture(n_components=n_clusters, random_state=42, n_init=10, max_iter=300)
    gmm.fit(X)
    labels = gmm.predict(X)
    sil_score = silhouette_score(X, labels)
    gmm_silhouette_scores.append(sil_score)
    print(f"K={n_clusters} - Silhouette Score: {sil_score:.4f}")
```

```
plt.figure(figsize=(8,5))
plt.plot(range_n_clusters, gmm_silhouette_scores, marker='o')
plt.title('Silhouette Score para diferentes valores de K (GMM)')
plt.xlabel('Número de clusters (K)')
plt.ylabel('Silhouette Score')
plt.xticks(range_n_clusters)
plt.grid(True)
plt.show()

best_k = range_n_clusters[gmm_silhouette_scores.index(max(gmm_silhouette_scores))]
print(f"\nMelhor K de acordo com o Silhouette Score (GMM): {best_k} (Score: {max(gmm_silhouette_scores)})")
```

K=2 - Silhouette Score: 0.5260  
K=3 - Silhouette Score: 0.5891  
K=4 - Silhouette Score: 0.6879  
K=5 - Silhouette Score: 0.6972  
K=6 - Silhouette Score: 0.7048  
K=7 - Silhouette Score: 0.6793  
K=8 - Silhouette Score: 0.6416  
K=9 - Silhouette Score: 0.6767  
K=10 - Silhouette Score: 0.6438  
K=11 - Silhouette Score: 0.6823  
K=12 - Silhouette Score: 0.6377  
K=13 - Silhouette Score: 0.6880  
K=14 - Silhouette Score: 0.6560  
K=15 - Silhouette Score: 0.6654  
K=16 - Silhouette Score: 0.6192  
K=17 - Silhouette Score: 0.6044  
K=18 - Silhouette Score: 0.6338  
K=19 - Silhouette Score: 0.6362  
K=20 - Silhouette Score: 0.6263  
K=21 - Silhouette Score: 0.6196  
K=22 - Silhouette Score: 0.6034  
K=23 - Silhouette Score: 0.5981  
K=24 - Silhouette Score: 0.5888



Melhor K de acordo com o Silhouette Score (GMM): 6 (Score: 0.7048)

## 8.3 Hierarchical Clustering

```
In [ ]: range_n_clusters = range(2, 25)
        hclust_silhouette_scores = []
```

```

for n_clusters in range_n_clusters:
    hclust = AgglomerativeClustering(n_clusters=n_clusters, linkage='ward', metric='euclidean')
    labels = hclust.fit_predict(X)
    sil_score = silhouette_score(X, labels, metric='euclidean')
    hclust_silhouette_scores.append(sil_score)
    print(f"Hierarchical clustering K={n_clusters} - Silhouette Score: {sil_score:.4f}")

plt.figure(figsize=(8,5))
plt.plot(range_n_clusters, hclust_silhouette_scores, marker='o')
plt.title('Silhouette Score para diferentes valores de K (Hierarchical)')
plt.xlabel('Número de clusters (K)')
plt.ylabel('Silhouette Score')
plt.xticks(range_n_clusters)
plt.grid(True)
plt.show()

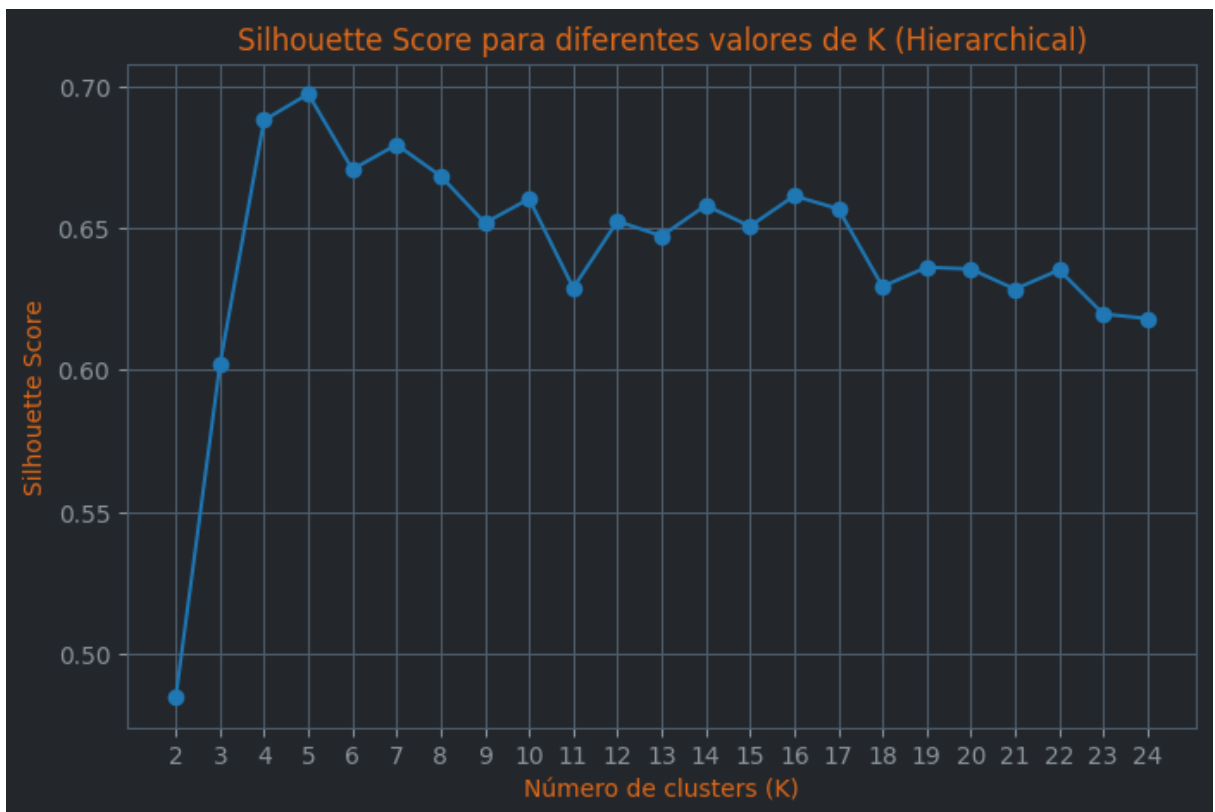
best_k_hclust = range_n_clusters[hclust_silhouette_scores.index(max(hclust_silhouette_scores))]
print(f"\nMelhor K de acordo com o Silhouette Score para o hierarchical clustering: {best_k_hclust}")

```

```

Hierarchical clustering K=2 - Silhouette Score: 0.4847
Hierarchical clustering K=3 - Silhouette Score: 0.6018
Hierarchical clustering K=4 - Silhouette Score: 0.6879
Hierarchical clustering K=5 - Silhouette Score: 0.6972
Hierarchical clustering K=6 - Silhouette Score: 0.6706
Hierarchical clustering K=7 - Silhouette Score: 0.6793
Hierarchical clustering K=8 - Silhouette Score: 0.6682
Hierarchical clustering K=9 - Silhouette Score: 0.6517
Hierarchical clustering K=10 - Silhouette Score: 0.6602
Hierarchical clustering K=11 - Silhouette Score: 0.6289
Hierarchical clustering K=12 - Silhouette Score: 0.6523
Hierarchical clustering K=13 - Silhouette Score: 0.6471
Hierarchical clustering K=14 - Silhouette Score: 0.6579
Hierarchical clustering K=15 - Silhouette Score: 0.6505
Hierarchical clustering K=16 - Silhouette Score: 0.6612
Hierarchical clustering K=17 - Silhouette Score: 0.6567
Hierarchical clustering K=18 - Silhouette Score: 0.6293
Hierarchical clustering K=19 - Silhouette Score: 0.6361
Hierarchical clustering K=20 - Silhouette Score: 0.6355
Hierarchical clustering K=21 - Silhouette Score: 0.6284
Hierarchical clustering K=22 - Silhouette Score: 0.6352
Hierarchical clustering K=23 - Silhouette Score: 0.6197
Hierarchical clustering K=24 - Silhouette Score: 0.6181

```



Melhor K de acordo com o Silhouette Score para o hierarchical clustering: 5 (Score: 0.6972)

## 8.4 Results

```
In [ ]: ks = list(range(2, 25))
scores_dict = {
    'KMeans': kmeans_silhouette_scores,
    'GMM': gmm_silhouette_scores,
    'Agglomerative': hclust_silhouette_scores,
}

scores_df = pd.DataFrame(
    [scores_dict[algo] for algo in scores_dict],
    columns=[f'K={k}' for k in ks],
    index=['KMeans', 'GMM', 'Agglomerative']
)

def highlight_best(s):
    is_max = s == s.max()
    return ['background-color: lightgreen; font-weight: bold' if v else '' for v in is_max]

display(scores_df.style.apply(highlight_best, axis=1))
```

	K=2	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10
KMeans	0.526025	0.602701	0.687906	<b>0.697174</b>	0.670579	0.642400	0.668217	0.622331	0.660736
GMM	0.526025	0.589087	0.687906	0.697174	<b>0.704840</b>	0.679290	0.641613	0.676717	0.643753
Agglomerative	0.484689	0.601800	0.687906	<b>0.697174</b>	0.670579	0.679290	0.668217	0.651710	0.660210

## 8.5 Silhouette Analysis

```
In [ ]: gmm_7 = GaussianMixture(n_components=7, random_state=42)
gmm_labels = gmm_7.fit_predict(X)

silhouette_avg = silhouette_score(X, gmm_labels)
sample_silhouette_values = silhouette_samples(X, gmm_labels)
```

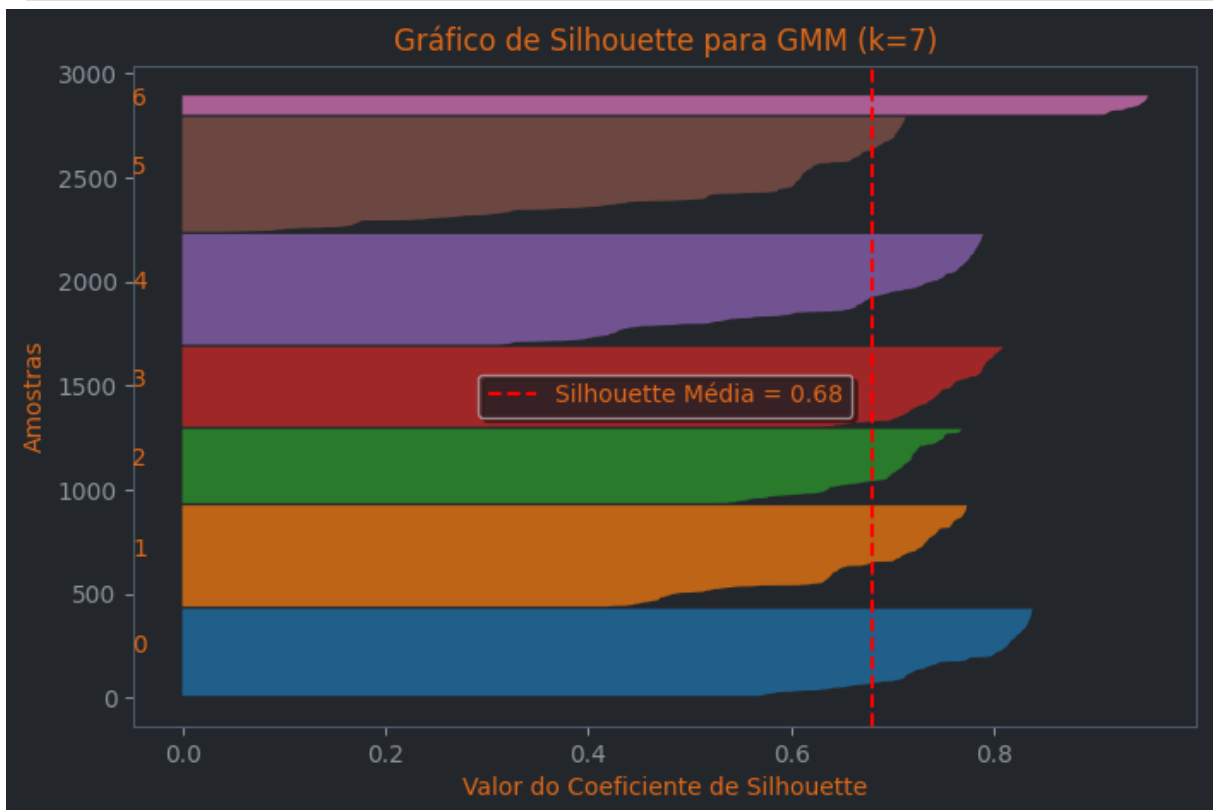
```

plt.figure(figsize=(8, 5))
y_lower = 10
for i in range(7):
    ith_cluster_silhouette_values = sample_silhouette_values[gmm_labels == i]
    ith_cluster_silhouette_values.sort()
    size_cluster_i = ith_cluster_silhouette_values.shape[0]
    y_upper = y_lower + size_cluster_i

    plt.fill_betweenx(
        np.arange(y_lower, y_upper),
        0, ith_cluster_silhouette_values,
        alpha=0.7
    )
    plt.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
    y_lower = y_upper + 10

plt.axvline(x=silhouette_avg, color="red", linestyle="--", label=f"Silhouette Média = {silhouette_avg}")
plt.xlabel("Valor do Coeficiente de Silhouette")
plt.ylabel("Amostras")
plt.title("Gráfico de Silhouette para GMM (k=7)")
plt.legend()
plt.show()

```



```

In [ ]: kmeans_5 = KMeans(n_clusters=5, random_state=42, n_init=10, max_iter=300, init='random')
kmeans_labels = kmeans_5.fit_predict(X)

silhouette_avg_kmeans = silhouette_score(X, kmeans_labels)
sample_silhouette_values_kmeans = silhouette_samples(X, kmeans_labels)

plt.figure(figsize=(8, 5))
y_lower = 10

n_clusters = 5
for i in range(n_clusters):
    ith_cluster_silhouette_values = sample_silhouette_values_kmeans[kmeans_labels == i]
    ith_cluster_silhouette_values.sort()
    size_cluster_i = ith_cluster_silhouette_values.shape[0]
    y_upper = y_lower + size_cluster_i

    plt.fill_betweenx(
        np.arange(y_lower, y_upper),

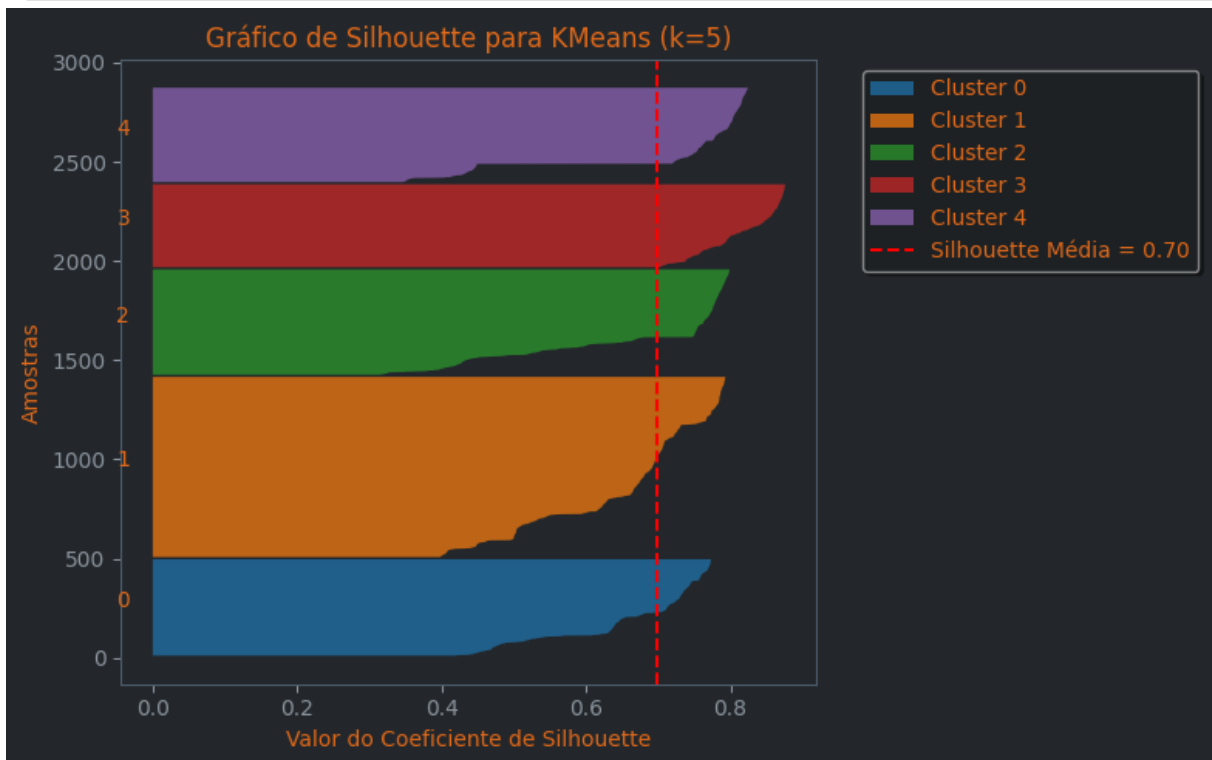
```

```

    0, ith_cluster_silhouette_values,
    alpha=0.7,
    label=f'Cluster {i}'
)
plt.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
y_lower = y_upper + 10

plt.axvline(x=silhouette_avg_kmeans, color="red", linestyle="--", label=f"Silhouette Média = {s
plt.xlabel("Valor do Coeficiente de Silhouette")
plt.ylabel("Amostras")
plt.title("Gráfico de Silhouette para KMeans (k=5)")
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()

```



```

In [ ]: agгло_5 = AgglomerativeClustering(n_clusters=5)
        agгло_labels = agгло_5.fit_predict(X)

        silhouette_avg_agglo = silhouette_score(X, agгло_labels)
        sample_silhouette_values_agglo = silhouette_samples(X, agгло_labels)

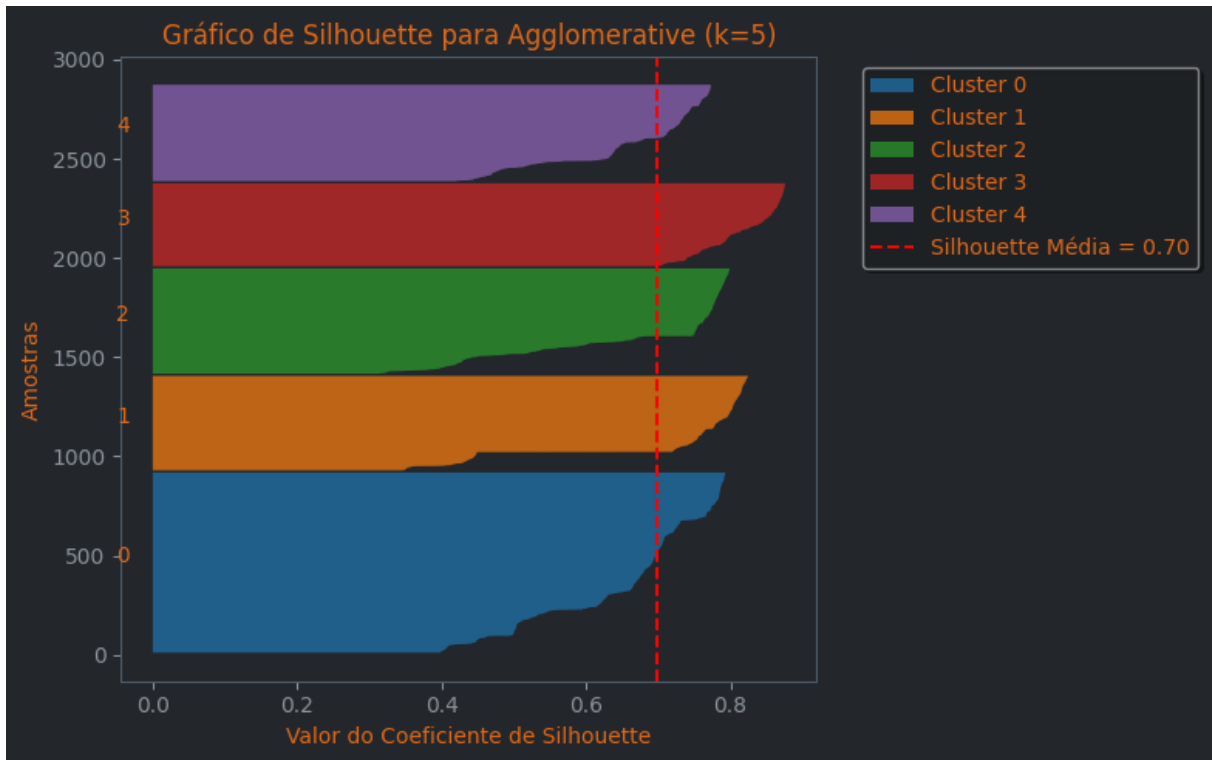
        plt.figure(figsize=(8, 5))
        y_lower = 10
        n_clusters = 5
        for i in range(n_clusters):
            ith_cluster_silhouette_values = sample_silhouette_values_agglo[agгло_labels == i]
            ith_cluster_silhouette_values.sort()
            size_cluster_i = ith_cluster_silhouette_values.shape[0]
            y_upper = y_lower + size_cluster_i

            plt.fill_betweenx(
                np.arange(y_lower, y_upper),
                0, ith_cluster_silhouette_values,
                alpha=0.7,
                label=f'Cluster {i}'
            )
            plt.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
            y_lower = y_upper + 10

        plt.axvline(x=silhouette_avg_agglo, color="red", linestyle="--", label=f"Silhouette Média = {si
        plt.xlabel("Valor do Coeficiente de Silhouette")
        plt.ylabel("Amostras")
        plt.title("Gráfico de Silhouette para Agglomerative (k=5)")

```

```
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



## 9. Machine Learning

```
In [46]: df5 = df4.copy()
df5 = pd.DataFrame(df5)
```

### 9.1 GMM

```
In [47]: gmm = GaussianMixture(n_components=6, random_state=42, n_init=10, max_iter=300)
gmm.fit(X)
```

```
df5['cluster'] = gmm.predict(X)
```

```
In [ ]: sil_score = silhouette_score(X, gmm.predict(X), metric='euclidean')
print(f"Silhouette Score do cluster (GMM - Euclidean): {sil_score:.4f}")
```

```
log_likelihood = gmm.score(X) * X.shape[0]
print(f"Log-likelihood do GMM: {log_likelihood:.4f}")
```

Silhouette Score do cluster (GMM - Euclidean): 0.7048

Log-likelihood do GMM: -13807.5059

## 10. Cluster Analysis

```
In [49]: df6 = df5.copy()
```

### 10.1 Visual Inspection

```
In [ ]: gmm_labels = gmm.predict(X)
silhouette_vals = silhouette_samples(X, gmm_labels)
```

```
n_clusters = 6
y_lower = 10
fig, ax1 = plt.subplots(figsize=(8, 6))
```

```

for i in range(n_clusters):
    ith_cluster_silhouette_values = silhouette_vals[gmm_labels == i]
    ith_cluster_silhouette_values.sort()

    size_cluster_i = ith_cluster_silhouette_values.shape[0]
    y_upper = y_lower + size_cluster_i

    color = plt.cm.nipy_spectral(float(i) / n_clusters)
    ax1.fill_betweenx(np.arange(y_lower, y_upper), 0, ith_cluster_silhouette_values, facecolor=

    ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

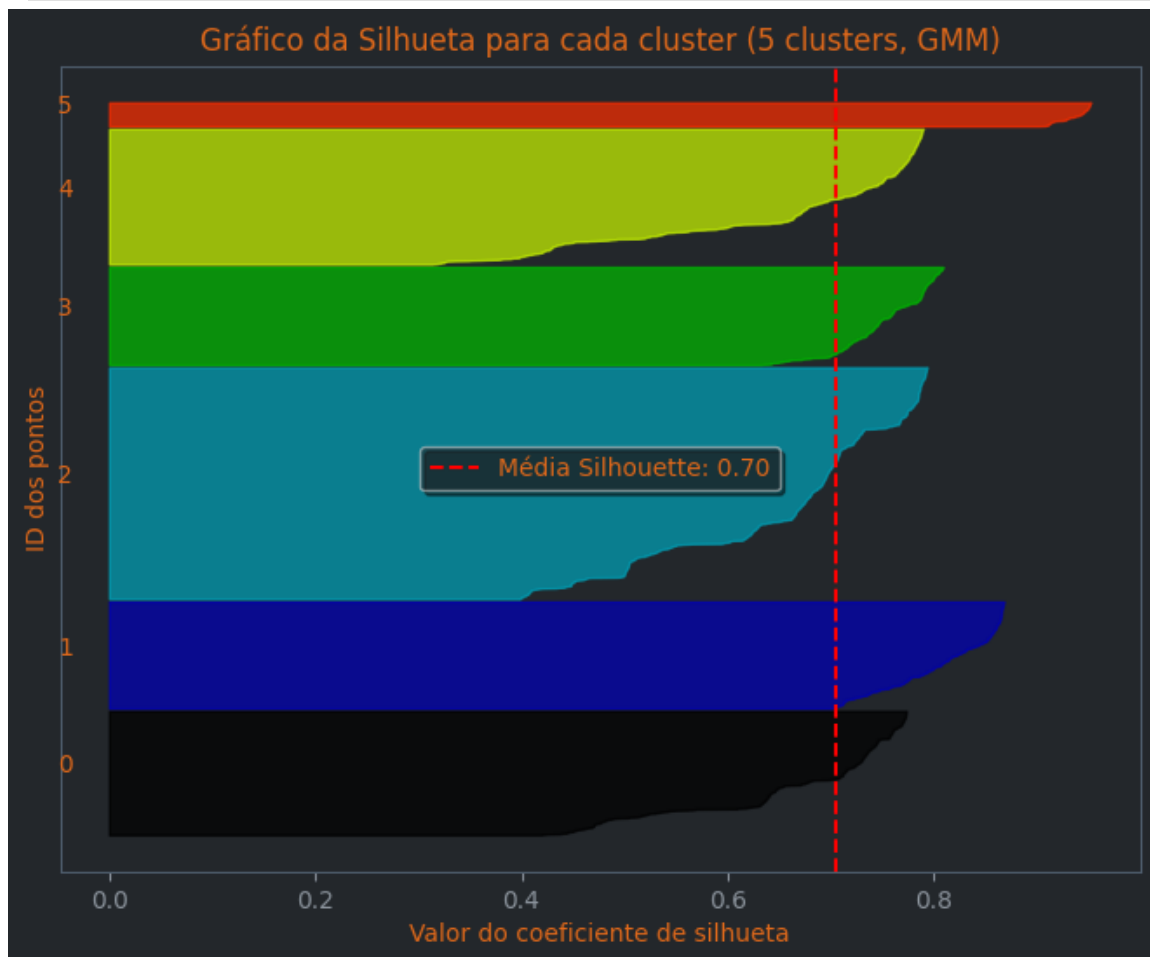
    y_lower = y_upper + 10

ax1.set_title("Gráfico da Silhueta para cada cluster (5 clusters, GMM)")
ax1.set_xlabel("Valor do coeficiente de silhueta")
ax1.set_ylabel("ID dos pontos")

silhouette_avg = silhouette_score(X, gmm_labels)
ax1.axvline(x=silhouette_avg, color="red", linestyle="--", label=f"Média Silhouette: {silhouett

ax1.set_yticks([])
ax1.legend()
plt.show()

```

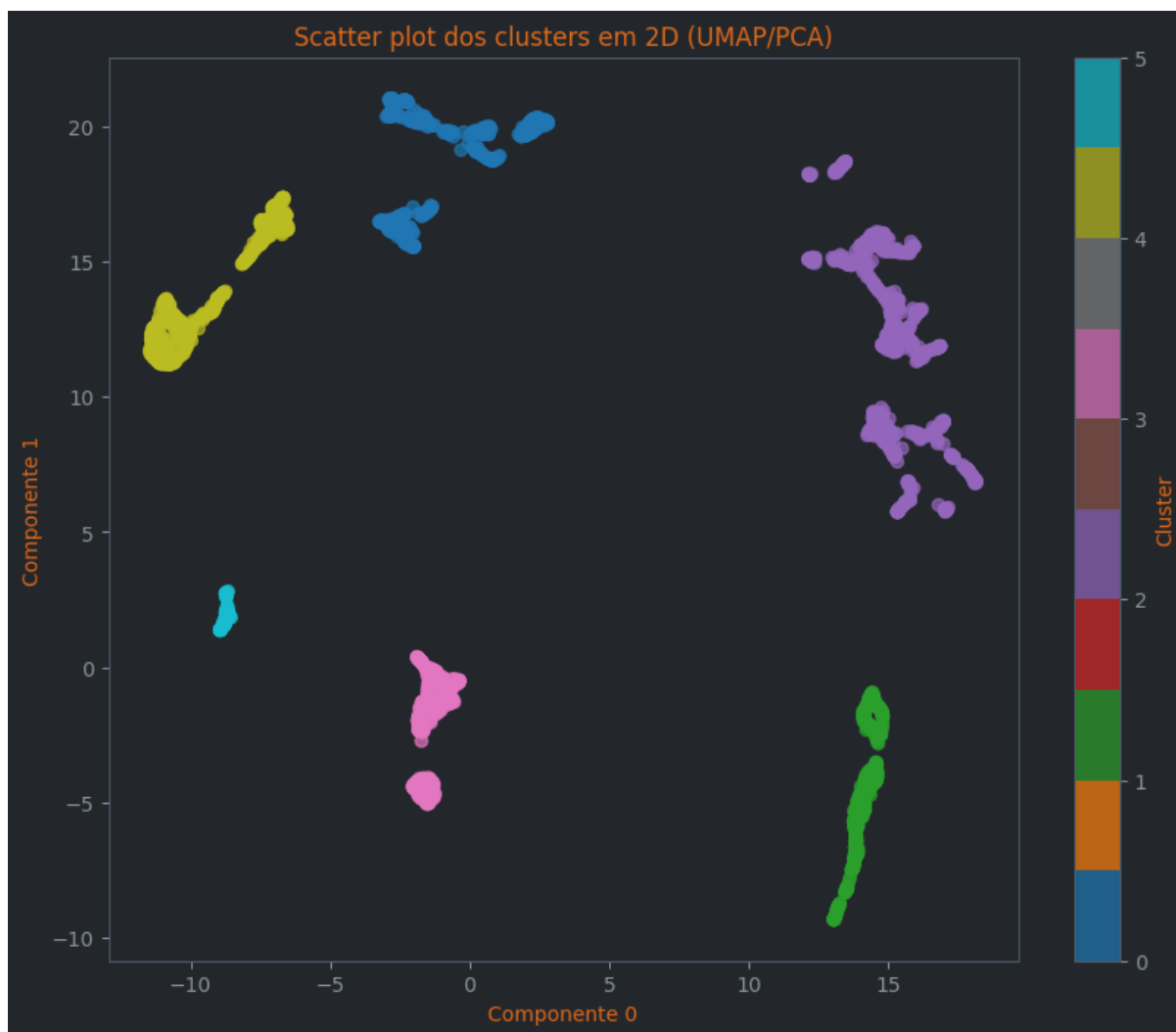


## 10.2 Visualization of Clusters

```

In [51]: plt.figure(figsize=(10,8))
scatter = plt.scatter(df6['UMAP_1'], df6['UMAP_2'], c=df6['cluster'], cmap='tab10', alpha=0.7)
plt.xlabel('Componente 0')
plt.ylabel('Componente 1')
plt.title('Scatter plot dos clusters em 2D (UMAP/PCA)')
plt.colorbar(scatter, label='Cluster')
plt.show()

```



## 10.3 Cluster Profile

```
In [52]: df3['cluster'] = df6['cluster']
cluster_profile = get_cluster_profile(df3, cluster_col='cluster', id_col='customer_id')
```

	cluster	num_customers	gross_revenue	recency	qtd_produtos	orders_count	frequency_days	recency_days
0	0	487	1052.714517	62.100616	615.078029	3.340862	80.731422	9.2
1	1	419	11843.508473	22.155131	7036.410501	16.417661	33.412178	166.0
2	2	909	2155.753619	39.850385	1266.741474	6.336634	58.374603	21.3
3	3	384	349.490833	100.705729	132.130208	2.468750	98.494829	2.4
4	4	532	661.282707	78.958647	348.048872	2.697368	96.863095	4.3
5	5	93	488.165806	82.397849	225.537634	2.677419	114.438710	5.0

### Cluster 0

#### Cluster 0 – Active Mid-Value Customers

(487 customers | 17.2% of the base)

#### Profile

- Low revenue: 1052
- Purchase relatively little
- Reasonable purchase frequency: 3.3 orders

- Relatively high recency: 62 days
- Good lifetime: 167 days
- Low return rate: 1.5%
- Medium-low monetary\_per\_day: 30.5

**Interpretation** These are customers who have already bought a reasonable amount, but did not scale in value. They are not bad, not great — they are in an **opportunity zone**.

#### **Actions**

- Stimulus campaigns: coupons / bundles
- Encourage second / third purchase
- Customer education about products
- Upgrade programs to move them to Cluster 2

**Goal:** increase ticket size + purchase frequency

#### Cluster 1

##### **Cluster 1 – Elite Customers / High Value Insiders**

**(419 customers | 14.8% of the base | highest absolute and per-day revenue)**

#### **Profile**

- **Highest gross revenue:** 11.8k on average
- **Highest purchase volume:** 7,036 products on average
- **Very high number of orders:** 16.4
- **Very recently active customers:** recency 22 days
- **Low return rate:** 1.7%
- **Highest lifetime:** 295 days
- **Very high monetary\_per\_day:** 81.9

**Interpretation** This is clearly the natural **VIP / Insider** group. They buy a lot, buy frequently, maintain a long-term relationship, and generate very high financial return with low return risk.

#### **Business Actions**

- **Exclusive offers / Insider club**
- Premium benefits: higher cashback, priority support, pre-sale access
- Retention programs: emotional recognition + real advantages
- Avoid changing price too much → **less price sensitive**

**Objective:** maintain, engage, and increase recurrence

#### Cluster 2

##### **Cluster 2 – Good Value, Consistent and Reliable Customers**

**(909 customers | 32.2% of the base | largest group)**

#### **Profile**

- Good revenue: ≈ 2.1k
- Buy quite a lot: 1,266 products
- Reasonable number of orders: 6.3
- Moderate recency: 39 days

- High lifetime: 242 days
- **Low return rate:** 1.7%
- High monetary\_per\_day: 62.6

**Interpretation** A strong, loyal, and profitable base, but not as intense as the VIPs. Stable customers, good spenders, with few return issues.

#### **Business Actions**

- Structured loyalty programs
- Incentives to increase frequency (e.g., comeback campaigns)
- Upsell to move them toward Cluster 1
- Care to prevent them from cooling down

**Objective:** accelerate their journey toward the VIP group

### Cluster 3

#### **Cluster 3 – Very Low Value Customers / High Churn Risk**

**(384 customers | 13.6%)**

#### **Profile**

- Lowest gross revenue: 349
- Very few products purchased
- Few orders
- **Very high recency:** 100 days
- Low activity
- Relatively higher return rate
- monetary\_per\_day: 22.2
- Lowest lifetime

**Interpretation** Customers practically lost. Low value, little relationship, almost inactive.

#### **Actions**

- Run campaigns only if cost is low
- Automated strategy → basic email / remarketing
- If marketing cost > predictable return → maybe not worth insisting

**Objective:** only recover those who respond cheaply

### Cluster 4

#### **Cluster 4 – Low Value Customers with Good Retention**

**(532 customers | 18.8%)**

#### **Profile**

- Low revenue: 661
- Few orders: 2.7
- High recency: 78 days
- Lifetime: 147 days
- Low return rate: 1.2%
- monetary\_per\_day: 23.7

**Interpretation** Customers who had a relationship with the brand but never evolved. They are “almost lost,” but still show some signs of life.

#### **Actions**

- Reactivation campaigns
- More aggressive offers (\$\$ matters to them)
- Simple cross-sell
- Clear and direct communication
- Customer recovery efforts

**Objective:** prevent definitive churn

#### Cluster 5

#### **Cluster 5 – Problematic Customers / High Operational Risk**

**(93 customers | 3.3%)**

#### **Profile**

- Low revenue
- Buy relatively little
- High recency
- **Highest frequency\_days** = they purchase with very long intervals
- **Highest returns\_rate** in the entire base
- Low monetary\_per\_day

**Interpretation** Small, not financially relevant and they **generate operational cost**. Likely customers who buy incorrectly, return frequently, or exploit the system.

#### **Actions**

- **Do not include in premium programs**
- Restrict benefits
- Review exchange and logistics policies
- Monitor operational risk

**Objective:** control losses

#### Resume

Cluster	Role in the Business
1	VIP. Full focus on retention and exclusivity
2	Strong and healthy base. Expand value
0	Medium potential. Stimulate growth
4	Near churn. Recover those who are worth it
3	Almost lost. Only low-cost actions
5	Problem customers. Protect operations

## 11. Conclusion

## 11.1 Business Questions

### 1. Who are the customers eligible to participate in the Insiders program?

The eligible customers are those classified in **Cluster 1**, which represents the group with the highest strategic value for the business. These are highly active customers, with a long relationship history, extremely high revenue generation, strong purchase recurrence, and low operational risk. In practical terms, they are the most loyal, most engaged, and most profitable customers in the base.

### 2. How many customers will be part of the group?

The Insiders group is composed of:

- **419 customers**
- Representing **14.84% of the total base**

In other words, it is a relatively small group, but extremely relevant financially.

### 3. What are the main characteristics of these customers?

The Insiders present the following average profile:

- **Highest average revenue:** £ 11,843 per customer
- **Highest purchased volume:** 7,036 products on average
- **Very high purchase recurrence:** 16.4 orders per customer
- **High relationship frequency** (frequency\_days = 33, average days between purchases)
- **Extremely active customers recently:** recency of only 22 days
- **Highest lifetime in the base:** 295 days
- **Low return rate:** 1.7%
- **Highest monetary\_per\_day:** 81.95

This is clearly the group with the highest value and the best commercial behavior.

### 4. What percentage of revenue contribution comes from the insiders?

```
In [ ]: faturamento_total_cluster1 = df3[df3['cluster'] == 1]['gross_revenue'].sum()

faturamento_total_base = df3['gross_revenue'].sum()
participacao_cluster1 = faturamento_total_cluster1 / faturamento_total_base

print(f"A participação do cluster de insiders no faturamento total é de {participacao_cluster1:
```

- Decimal value: **0.6229**
- In percentage: **0.6229 × 100 = 62.29%**

The customers in the Insider group represent approximately **62.3% of the company's total revenue**, despite accounting for only 14.8% of the base. This confirms that they are the most strategic and financially relevant portion of the business.

### 5. What is the expected revenue from this group for the next few months?

Since we already have the indicator **monetary\_per\_day = 81.95**, we can project expected revenue:

- Per customer per day: **£ 81.95**
- For 419 customers:

Estimates:

**30 days**  $\approx 81.95 \times 419 \times 30$   
 $\approx \text{£ } 1,030,000$  (approx.)

**3 months (90 days)**  $\approx \text{£ } 3.1$  million  
(assuming the current behavior remains)

This demonstrates that the Insider group sustains a critical portion of future revenue.

## 6. What are the conditions for a customer to be eligible for the Insider group?

The conditions can be defined aligned with the averages of Cluster 1:

A customer can join the Insider group if they demonstrate behavior similar to:

- High value generated (high gross revenue)
- High purchase recurrence (orders\_count > ~10)
- Recent engagement (low recency)
- Active frequency (recurring purchases, low frequency\_days)
- Long-term relationship (high lifetime)
- Low return rate

In other words, the customer needs to demonstrate **consistency, value and low risk**.

## 7. What are the conditions for a customer to be removed from the Insider group?

An Insider customer may be removed if they show signs of deterioration, such as:

- Significant increase in recency (goes a long time without purchasing)
- Consistent reduction in the number of orders
- Relevant drop in monetary\_per\_day
- Increase in return rate
- Clear reduction in engagement

In practice, if the customer starts behaving more like clusters 3, 4 or 5, they should no longer be considered an Insider.

## 8. What is the guarantee that the Insider program is better than the rest of the customer base?

The data objectively proves that the Insider group is superior:

Comparing Cluster 1 with the other clusters, it has:

- **the highest average revenue**
- **the highest number of purchased products**
- **the highest number of orders**
- **the longest lifetime**
- **the lowest recency**
- **the highest monetary\_per\_day**
- low return rate

In other words:

They buy more, buy more frequently, have been active for longer, generate more money and cause fewer operational issues.

This proves that the Insider program represents a **healthier, more profitable and strategically relevant** portion of the base.

## 9. What actions can the marketing team take to increase revenue?

### 1. For Insiders (Cluster 1)

- Exclusive benefits
- Priority in new releases
- Differentiated cashback
- Premium recognition
- Avoid aggressive discounts (they are not price sensitive)

### 2. For Cluster 2 (good customers)

- Campaigns to increase frequency
- Upsell to move them into the Insider group
- Loyalty programs

### 3. For Cluster 0 and 4 (warm customers)

- Moderate incentives
- Return campaigns
- Product bundles

### 4. For Cluster 3 (almost lost)

- Low-cost reactivation
- Light remarketing

### 5. For Cluster 5 (problematic customers)

- Operational control
- Do not prioritize marketing investment

## 11.2 Insiders List

```
In [56]: df_insiders = df3[['customer_id', 'cluster']].copy()
df_insiders['is_insider'] = df_insiders['cluster'].apply(lambda x: 'yes' if x == 1 else 'no')
df_insiders = df_insiders[['customer_id', 'is_insider']]
df_insiders.to_csv('../data/insiders.csv', index=False)
```

## 11.3 Conclusion

This project successfully achieved the goal of **identifying high-value customers in an e-commerce environment** and objectively defining, in a data-driven way, who should be part of the **Insiders Program**.

Using a clustering model (GMM – Euclidean), with **excellent segmentation quality**, demonstrated by a **Silhouette Score of 0.7048**, it was possible to segment the customer base into **6 clearly distinct groups**, enabling a deep understanding of purchasing behavior, financial value, risk and growth potential of each segment.

Among the generated clusters, **Cluster 1** clearly stood out as the group with the highest strategic value. It includes **419 customers (14.8% of the base)** who concentrate approximately **62.3% of the total revenue**, presenting:

- the highest individual average revenue
- the highest number of orders and purchased volume
- high purchase frequency and low recency

- the longest customer lifetime
- a low return rate
- the highest *monetary\_per\_day* in the base

In other words, these customers **buy more, buy more frequently, have been active for longer, generate more revenue, and present low operational risk**, naturally becoming the **Insiders**. In addition, based on their current behavior, it was possible to **estimate more than R\$ 1 million in revenue within 30 days** and around **R\$ 3 million in 90 days**, reinforcing their future relevance.

The model also allowed the definition of:

- **objective eligibility criteria** to enter the program (high value, recurrence, engagement, low risk)
- **clear removal criteria** (reduced frequency, increased recency, decreased revenue, increased returns)
- statistical and behavioral evidence proving that the Insider group is **significantly superior to the rest of the base**
- **specific marketing strategies** for each cluster, balancing retention, value growth, churn recovery and operational control

Therefore, this project delivers not only an analytical model but a **practical business decision tool**, capable of:

1. Intelligently segmenting the customer base
2. Prioritizing the highest value customers
3. Guiding CRM and Marketing strategies
4. Maximizing revenue and retention
5. Reducing costs and risks
6. Supporting decisions with analytical evidence