Segmentação.py

Função: agrupar clientes em grupos específicos, utilizando coisas em comum entre eles: como hábitos de compras ou comportamentos de compra. Ele é útil no projeto porque ajuda os clientes da Cannoli a direcionar campanhas específicas para perfis específicos de clientes como, por exemplo, oferecer benefícios para clientes fiéis ou algo para reter clientes que não estão fidelizados.

Importações

```
# IA_e_ML/segmentacao.py

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette_score

import joblib

import matplotlib.pyplot as plt
```

Carregue seus dados

```
orders = pd.read_csv("data/orders.csv", parse_dates=["date"])

customers = pd.read_csv("data/customers.csv",
parse_dates=["signup_date","last_active_date"])
```

Construção de métrica RFM

```
snapshot_date = orders['date'].max() + pd.Timedelta(days=1)

rfm = orders.groupby('customer_id').agg({

   'date': lambda x: (snapshot_date - x.max()).days,

   'order_id': 'nunique',

   'amount': 'sum'

}).rename(columns={'date':'recency','order_id':'frequency','amount':'monetary'}).reset_index()
```

Merge com dados demográficos

```
df = rfm.merge(customers, on='customer_id', how='left')
```

Tratamento nulos simples

```
df['monetary'] = df['monetary'].fillna(0)

df[['recency','frequency']] =
df[['recency','frequency']].fillna(df[['recency','frequency']].median())
```

features para cluster

```
X = df[['recency','frequency','monetary']].copy()

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)
```

opcional: PCA para visualizar

```
pca = PCA(n_components=2, random_state=42)

X_pca = pca.fit_transform(X_scaled)
```

escolher K via silhouette

```
best_k, best_score = 2, -1

for k in range(2,8):
    km = KMeans(n_clusters=k, random_state=42)
    labels = km.fit_predict(X_scaled)
    score = silhouette_score(X_scaled, labels)
    if score > best_score:
        best_score = score
        best_k = k


kmeans = KMeans(n_clusters=best_k, random_state=42)

df['cluster'] = kmeans.fit_predict(X_scaled)
```

salvar objetos

```
joblib.dump(scaler, "IA_e_ML/segmentacao_scaler.joblib")
```

```
joblib.dump(pca, "IA_e_ML/segmentacao_pca.joblib")
joblib.dump(kmeans, "IA_e_ML/kmeans_model.joblib")
df.to_csv("IA_e_ML/segmentos_clientes.csv", index=False)
```

plot para relatório

```
plt.figure(figsize=(6,4))
plt.scatter(X_pca[:,0], X_pca[:,1], c=df['cluster'], cmap='viridis', s=15)
plt.title(f"KMeans clusters (k={best_k})")
plt.xlabel("PCA1"); plt.ylabel("PCA2")
plt.tight_layout()
plt.savefig("IA_e_ML/cluster_pca.png", dpi=150)
```