Projeto: Análise de Preços de Produtos do Mercado Livre

Vídeo:

https://drive.google.com/file/d/1AXQrnV08p0dK2hhrfNerqeqkkzObc8r4/view?usp=sharing

Sumário

1. Objetivo do Projeto

Auxiliar vendedores do Mercado Livre na precificação e segmentação (público alvo) de seus produtos. Para isso, utilizamos técnicas de ciência de dados e Machine Learnig, como a clusterização de preços (K-Means e DBSCAN), detecção de outliers e análise estatística dos dados. O projeto também inclui a exposição do agrupamento de preços atuais via Flask em 3 segmentos (Ecomômico, intermediário e Premium) e a geração de dashboards no Power BI com média, preços máximos e mínimos por grupo.

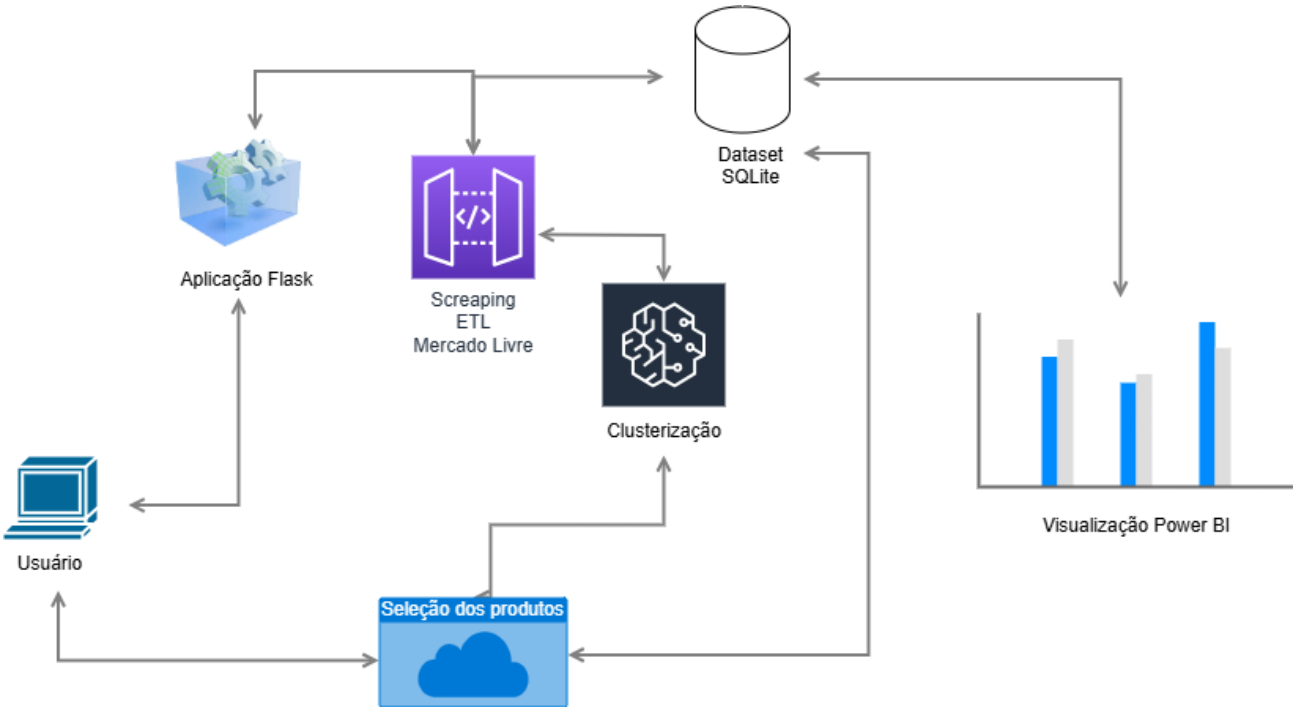


Figura 1: Fluxo de dados do projeto

2. Pipeline do Projeto

1. **Extração dos dados**: Coleta de informações sobre produtos e preços do Mercado Livre (API)
2. **Armazenamento**: Utilização de um banco de dados SQLite para persistência dos dados.
3. **Análise exploratória**: Limpeza e verificação de valores faltantes; Estudo das distribuições estatísticas das variáveis. (id, produto, preço e grupo)
4. **Normalização dos preços com MinMaxScaler:**

scaler = MinMaxScaler() df['preco_normalizado'] = scaler.fit_transform(df[['preco']])

Essa normalização transforma os valores da coluna `"preco"` para um intervalo de 0 a 1 com o objetivo de facilitar comparações e melhorar a performance do modelo.

5. **Clusterização**: Uso dos métodos K-Means e DBSCAN para agrupar produtos com base no preço além de sua comparação.
6. **Detecção de outliers**: Identificação de valores discrepantes na base de preços.
7. **Disponibilização via Flask**: API e página web para visualização dos clusters e outlaiers.

8. **Integração com Power BI**: Uso do banco de dados SQLite como fonte para visualização dos dados.

3 Estatísticas Descritivas

Para entender a distribuição dos preços: `print(df['preco'].describe())`

Principais estatísticas calculadas:

- Média (mean): Valor médio dos preços.
- Mediana (median): Valor central da distribuição.
- Desvio padrão (std): Indica a dispersão dos preços.
- Variância (var): Mede a variabilidade dos dados.

4. Clusterização e Avaliação

A clusterização agrupa dados semelhantes sem supervisão com base em características comuns.

4.1 Método K-Means

Aplicamos o método do cotovelo para determinar o melhor número de clusters e realizamos a clusterização:

```
from sklearn.cluster import KMeans

best_k = 3
kmeans = KMeans(n_clusters=best_k, random_state=42, n_init=10)
df['cluster_kmeans'] = kmeans.fit_predict(df[['preco_normalizado']])
```

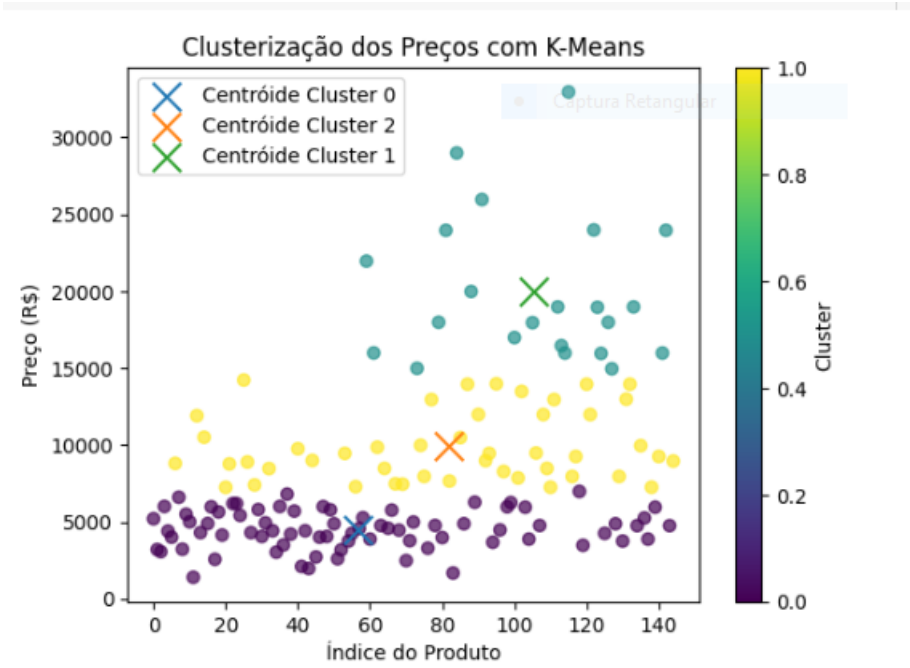


Figura 3: Clusterização k-means com centróides

4.2 Método DBSCAN

DBSCAN é aplicado para detectar outliers:

```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=0.3, min_samples=5)
df['cluster_dbscan'] = dbscan.fit_predict(df[['preco_normalizado']])
```

4.3 Comparando os Métodos (Silhouette)

Métrica de Silhouette Score para avaliar a qualidade dos clusters:

```
from sklearn.metrics import silhouette_score

silhouette_kmeans = silhouette_score(df[['preco_normalizado']], df['cluster_kmeans'])
silhouette_dbscan = silhouette_score(df[['preco_normalizado']], df['cluster_dbscan'])
print(f'Silhouette Score - K-Means: {silhouette_kmeans}')
print(f'Silhouette Score - DBSCAN: {silhouette_dbscan}')
```

A **pontuação de silhueta** mede o quão bem os pontos estão agrupados dentro de seus clusters e o quão separados estão dos outros clusters.

Valores próximos de 1 → Clusters bem separados e compactos (ótima clusterização).

Valores próximos de 0 → Sobreposição entre clusters (fraca separação)

Valores negativos → Pontos mal classificados (estão mais próximos de outro cluster do que do próprio).

Resultado:

Silhouette Score - KMeans: 0.591

Silhouette Score - DBSCAN: 0.701

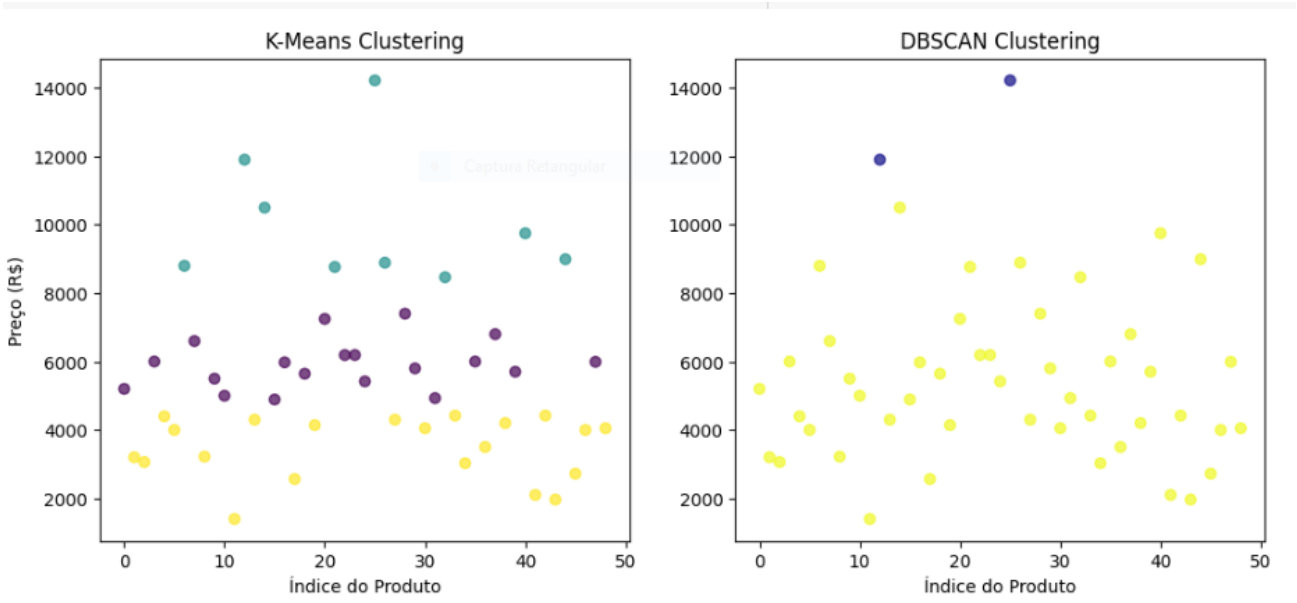


Figura 2: Comparação da clusterização

5. Visualização da Distribuição dos Clusters

5.1 K-Means com Outliers

```
import matplotlib.pyplot as plt
import numpy as np

x_values = np.linspace(0, 1, len(df))
outliers = df['preco'] > df['preco'].quantile(0.95)  # Considerando os 5% mais altos como outliers

plt.figure(figsize=(9,5))
plt.scatter(x_values, df['preco'], c=df['cluster_kmeans'], cmap='viridis', alpha=0.7)
plt.scatter(x_values[outliers], df['preco'][outliers], color='red', marker='x', s=100, label='Outliers')
plt.xlabel("Distribuição de Produtos")
plt.ylabel("Preço (R$)")
plt.title("Clusterização dos Preços com K-Means")
plt.colorbar(label="Cluster")
plt.legend()
plt.show()
```

5.2 DBSCAN

```
plt.figure(figsize=(9,5))
plt.scatter(x_values, df['preco'], c=df['cluster_dbscan'], cmap='plasma', alpha=0.7)
plt.xlabel("Distribuição de Produtos")
plt.ylabel("Preço (R$)")
plt.title("Clusterização dos Preços com DBSCAN")
plt.colorbar(label="Cluster")
plt.show()
```

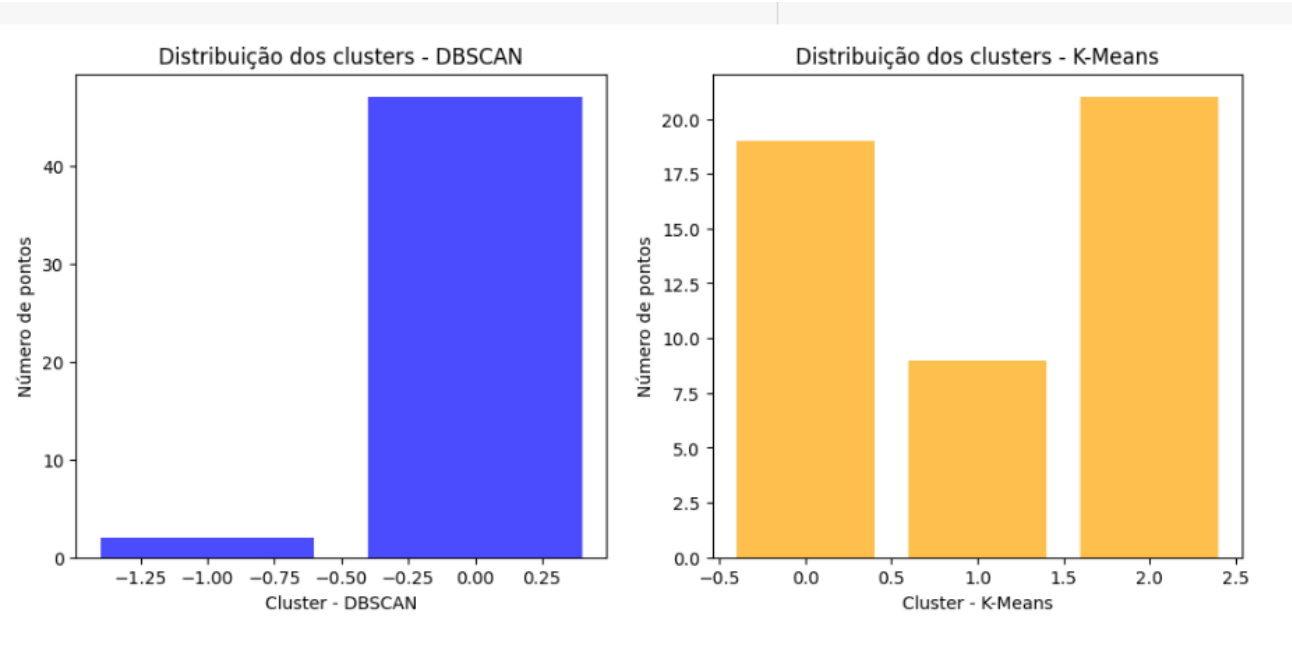


Figura 3: Distribuição dos clusters

5.3 Coeficiente de Variação (CV) - Dispersão dos Tamanhos dos Clusters (comparação)

```
cv_dbscan = np.std(dbscan_counts) / np.mean(dbscan_counts)
cv_kmeans = np.std(kmeans_counts) / np.mean(kmeans_counts)

print(f"Coeficiente de Variação - DBSCAN: {cv_dbscan:.2f}")
print(f"Coeficiente de Variação - K-Means: {cv_kmeans:.2f}")
```

O Coeficiente de Variação (CV) mede a dispersão relativa dos tamanhos dos clusters em relação à média. Ele é útil para avaliar o equilíbrio entre os clusters formados.
Se os clusters forem de tamanhos semelhantes, o CV será baixo. Se houver muita variação nos tamanhos dos clusters, o CV será alto.

Resultado:
Coeficiente de Variação - DBSCAN: 0.97
Coeficiente de Variação - K-Means: 0.47

6. Testes

Foram realizados testes com várias parâmetros para n_init (número de ajustes de centróides). Foi observado que o valor "auto" não realiza bem a identificação do cluster para uma nova observação.

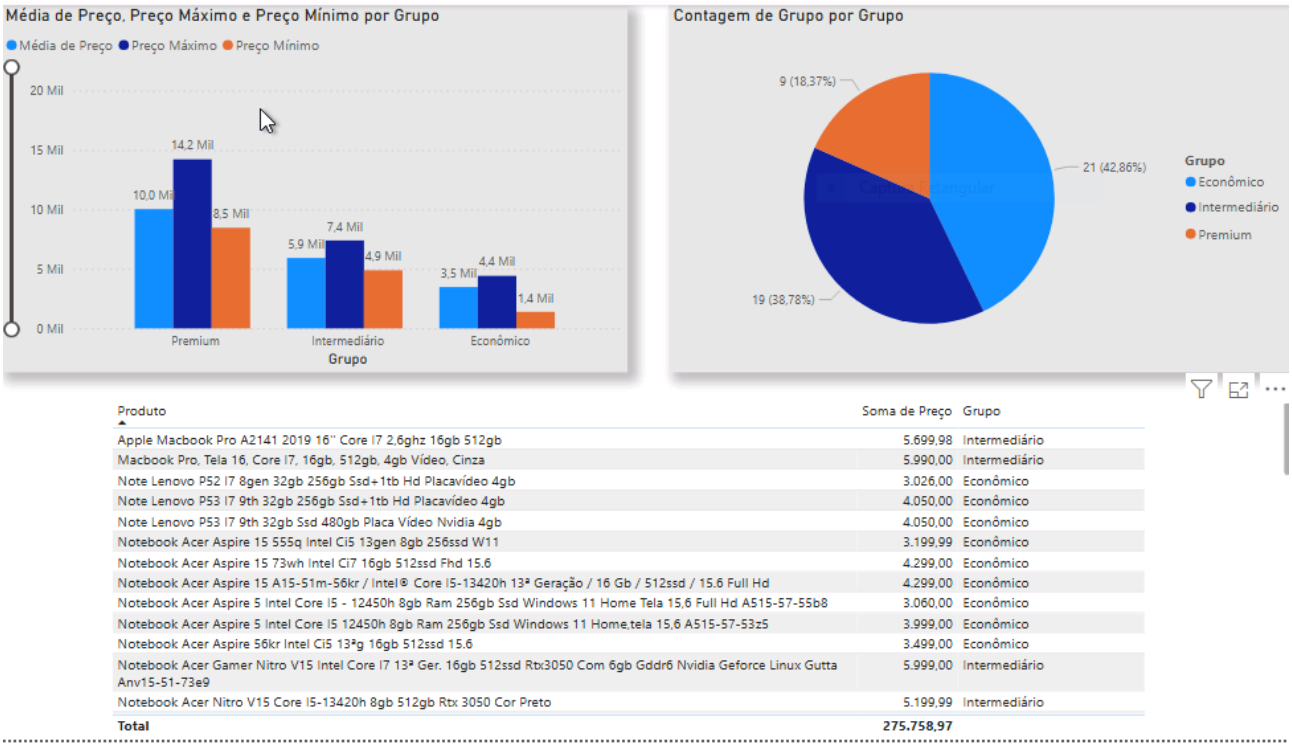
6. Disponibilização dos Dados via Flask

Criamos uma API Flask para acessar os dados clusterizados com o produto escolhido.

```
from flask import Flask, jsonify
import sqlite3

app = Flask(__name__)

@app.route('/dados', methods=['GET'])
def get_dados():
    conn = sqlite3.connect("mercado_livre.db")
    df = pd.read_sql_query("SELECT * FROM produtos", conn)
    conn.close()
    return jsonify(df.to_dict(orient='records'))

if __name__ == '__main__':
    app.run(debug=True)
```

7. Integração com Power BI

O banco de dados SQLite é utilizado como fonte de dados no Power BI foram criadas medidas relativas aos preços mínimo, máximo, médio e mediana dos produtos de cada grupo. As tabelas/dimensões são Produto e Grupo. Foram gerados relatórios e dashboards com gráficos de barras e pizza para análise dos preços médios e máximos por categoria.



Produto	Soma de Preço	Grupo
Apple Macbook Pro A2141 2019 16" Core I7 2.6ghz 16gb 512gb	5.699,98	Intermediário
Macbook Pro, Tela 16, Core I7, 16gb, 512gb, 4gb Vídeo, Cinza	5.990,00	Intermediário
Note Lenovo P52 I7 8gen 32gb 256gb Ssd+1tb Hd Placavídeo 4gb	3.026,00	Econômico
Note Lenovo P53 I7 9th 32gb 256gb Ssd+1tb Hd Placavídeo 4gb	4.050,00	Econômico
Note Lenovo P53 I7 9th 32gb Ssd 480gb Placa Vídeo Nvidia 4gb	4.050,00	Econômico
Notebook Acer Aspire 15 555q Intel Ci5 13gen 8gb 256ssd W11	3.199,99	Econômico
Notebook Acer Aspire 15 73wh Intel Ci7 16gb 512ssd Fhd 15.6	4.299,00	Econômico
Notebook Acer Aspire 15 A15-51m-56kr / Intel® Core I5-13420h 13ª Geração / 16 Gb / 512ssd / 15.6 Full Hd	4.299,00	Econômico
Notebook Acer Aspire 5 Intel Core I5 - 12450h 8gb Ram 256gb Ssd Windows 11 Home Tela 15,6 Full Hd A515-57-55b8	3.060,00	Econômico
Notebook Acer Aspire 5 Intel Core I5 12450h 8gb Ram 256gb Ssd Windows 11 Home,tela 15,6 A515-57-53z5	3.999,00	Econômico
Notebook Acer Aspire 56kr Intel Ci5 13ªg 16gb 512ssd 15.6	3.499,00	Econômico
Notebook Acer Gamer Nitro V15 Intel Core I7 13ª Ger. 16gb 512ssd Rtx3050 Com 6gb Gddr6 Nvidia Geforce Linux Gutta Anv15-51-73e9	5.999,00	Intermediário
Notebook Acer Nitro V15 Core I5-13420h 8gb 512gb Rtx 3050 Cor Preto	5.199,99	Intermediário
Total	**275.758,97**	

Figura 4: Dashboards no Power BI

8. Conclusão

O método K-Means foi escolhido por proporcionar uma segmentação mais equilibrada, identificando padrões claros de precificação. Ele apresentou menor coeficiente de variação, indicando clusters mais homogêneos. Além disso, seus centróides bem definidos facilitaram a análise e a estratégia de precificação no Mercado Livre. Embora o DBSCAN tenha melhor coeficiente de silhueta, sua menor quantidade de clusters e alta taxa de pontos classificados como ruído dificultaram a segmentação. Assim, o K-Means demonstrou ser a abordagem mais eficiente e interpretável para otimização de preços no e-commerce:

Analisando os resultados da clusterização, podemos observar três grupos bem definidos:

1. Cluster 0 (Econômico):
    - Preço médio: R$ 2.583,16
    - Faixa de preço: R$ 1.395,81 - R$ 3.219,00
    - Característica: Notebooks básicos e de entrada
    - 9 notebooks neste grupo
2. Cluster 1 (Intermediário):
    - Preço médio: R$ 4.624,54
    - Faixa de preço: R$ 4.050,00 - R$ 5.419,00
    - Característica: Notebooks gamers de entrada e notebooks intermediários

    - 9 notebooks neste grupo
3. Cluster 2 (Premium):
    - Preço médio: R$ 10.492,14
    - Faixa de preço: R$ 8.799,00 - R$ 14.219,00
    - Característica: Notebooks gamers de alto desempenho e notebooks premium
    - 7 notebooks neste grupo

A clusterização mostrou uma boa separação entre os grupos, que corresponde bem com as categorias originais (Econômico, Intermediário e Premium). Os clusters formados são coerentes com as características dos produtos e seus preços.

8. Pontos sensíveis relativos à qualidade dos dados

- Variação na forma de escrever especificações dos produtos
- Não consideração do custo-benefício ou outros fatores qualitativos na classificação dos produtos.

9. Melhorias futuras

- Padronização das especificações dos produtos
- Implantação de Sistema de pontuação baseado em especificações técnicas
- Amostra de produtos de outros marketplaces
- Inclusão de informações como quantidade de vendas, avaliações, reputação do vendedor, dados sobre garantia, frete e condições de pagamento
- Análise de variações por região