FUNDAÇÃO ESCOLA DE COMÉRCIO ALVARES PENTEADO

Lucca Brandão RA: 23024740

Murilo Lopes RA:24026183

Rodrigo Cruz RA: 24026578

Vinicius Kingo RA: 24026141

Busca Gulosa

A busca gulosa é um algoritmo que faz escolhas imediatas, optando sempre pela alternativa que parece ser mais vantajosa no momento. Essa estratégia prioriza o ganho local.

Na análise de dados de restaurantes, esse princípio pode ser ligado à determinação dos horários de maior movimento, focando no período com o maior volume de vendas sem considerar combinações de horários que poderiam ser mais significativas. Da mesma forma, ao avaliar produtos, uma estratégia gananciosa optaria apenas pelo item mais lucrativo ou mais vendido, desconsiderando que outros itens, em conjunto, poderiam elevar a receita de maneira mais sólida.

Assim, a busca gulosa é considerada uma abordagem simplificada de tomada de decisão na análise de restaurantes, com foco no retorno imediato.

Nosso código realiza uma análise dos dados de pedidos, identificando padrões de desempenho por empresa, plataforma, horário e dia da semana. Após o tratamento dos dados e cálculo de métricas como tempo de retirada, ticket médio e volume de pedidos, aplicamos o conceito de Busca Gulosa (Greedy Search) para determinar o horário e o dia da semana mais lucrativos. Essa abordagem escolhe a melhor opção imediata, ou seja, o momento com maior valor total de vendas, sem considerar combinações futuras. Os resultados são apresentados em gráficos que destacam visualmente os picos de vendas e o desempenho das empresas e plataformas.

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


plt.style.use("seaborn-v0_8")
plt.rcParams["figure.figsize"] = (12,6)


orders = pd.read_csv("Order_semicolon.csv", delimiter=";")
customers = pd.read_csv("Customer_semicolon.csv", delimiter=";")
campaigns = pd.read_csv("Campaign_semicolon.csv", delimiter=";")
campaign_queue = pd.read_csv("CampaignQueue_semicolon.csv", delimiter=";")


orders["totalAmount"] = pd.to_numeric(orders["totalAmount"], errors="coerce")
orders["takeOutTimeInSeconds"] = pd.to_numeric(orders["takeOutTimeInSeconds"], errors="coerce")
orders["preparationTime"] = pd.to_numeric(orders["preparationTime"], errors="coerce")


orders["tempo_retirada_min"] = orders["takeOutTimeInSeconds"].fillna(orders["preparationTime"]) / 60


orders["createdAt"] = pd.to_datetime(orders["createdAt"], errors="coerce", dayfirst=True)

```

```python
ranking_empresas = (
    orders.groupby("companyId")
    .agg(
        pedidos=("id", "count"),
        mediana_tempo=("tempo_retirada_min", "median"),
        p90_tempo=("tempo_retirada_min", lambda x: x.quantile(0.9)),
        ticket_medio=("totalAmount", "mean")
    )
    .reset_index()
    .sort_values("mediana_tempo")
)


ranking_plataformas = (
    orders.groupby("engineName")
    .agg(
        pedidos=("id", "count"),
        ticket_medio=("totalAmount", "mean"),
        mediana_tempo=("tempo_retirada_min", "median")
    )
    .reset_index()
    .sort_values("pedidos", ascending=False)
)

```

```python
pedidos_dia = (
    orders.groupby(orders["createdAt"].dt.date)
    .agg(
        pedidos=("id", "count"),
        receita_total=("totalAmount", "sum"),
        ticket_medio=("totalAmount", "mean")
    )
    .reset_index()
)


orders["hour"] = orders["createdAt"].dt.hour
vendas_hora = (
    orders.groupby("hour")["totalAmount"]
    .sum()
    .reset_index()
)

```

```python

melhor_horario = vendas_hora.loc[vendas_hora["totalAmount"].idxmax()]


print("=== Busca Gulosa (Horário) ===")
print(f"Horário mais lucrativo: {int(melhor_horario['hour'])}h")
print(f"Total de vendas: R$ {melhor_horario['totalAmount']:.2f}")


sns.barplot(data=vendas_hora, x="hour", y="totalAmount", palette="Blues_d")
plt.axvline(x=melhor_horario["hour"], color="red", linestyle="--",
            label=f"Melhor hora (gulosa): {int(melhor_horario['hour'])}h")


plt.legend()
plt.title("Vendas por Hora - Busca Gulosa escolhe o pico imediato")
plt.xlabel("Hora do dia")
plt.ylabel("Total de vendas (R$)")
plt.show()


orders["weekday"] = orders["createdAt"].dt.day_name(locale="pt_BR")
vendas_semana = (
    orders.groupby("weekday")["totalAmount"]
    .sum()
    .reset_index()
)


melhor_dia = vendas_semana.loc[vendas_semana["totalAmount"].idxmax()]


print("\n=== Busca Gulosa (Dia da Semana) ===")
print(f"Dia mais lucrativo: {melhor_dia['weekday']}")
print(f"Total de vendas: R$ {melhor_dia['totalAmount']:.2f}")

```

Como o código funciona

Leitura e tratamento dos dados:

Os arquivos CSV com as informações de pedidos e clientes são importados e limpos, garantindo que os valores numéricos e datas estejam padronizados.

Cálculo de métricas importantes:

Tempo médio e mediano de preparo;

Ticket médio (valor médio de cada pedido);

Quantidade de pedidos por empresa e plataforma;

Receita total por hora e por dia.

Aplicação do algoritmo de Busca Gulosa (Greedy Search):

O método seleciona diretamente o valor máximo em cada conjunto de dados — ou seja, identifica:

O horário mais lucrativo, com maior soma de vendas;

O dia mais lucrativo da semana.

Essa abordagem é chamada de “gulosa” porque sempre escolhe a melhor opção imediata, sem buscar combinações ou cenários futuros.

Visualização dos resultados:

O código gera gráficos de barras, destacando visualmente:

As horas do dia com maior volume de vendas;

O horário de pico (em vermelho);

Os dias da semana com melhor desempenho.

Benefícios para a empresa Cannoli

Ações promocionais direcionadas: Aplicar descontos ou anúncios nos horários e dias de maior movimento.

Melhor experiência do cliente: Redução de tempo de espera e aumento da eficiência da entrega.

Análise contínua: O código pode ser automatizado para rodar diariamente, alimentando dashboards internos.

Conclusão

Nosso código demonstra como o uso de análise de dados e algoritmos gulosos pode trazer insights rápidos e práticos para a gestão da Cannoli.

Ele permite que decisões sejam baseadas em dados reais, apoiando o crescimento sustentável e a otimização das operações comerciais da empresa.