Introdução

Este projeto teve como foco a análise de pedidos cancelados em um contexto de serviço de entregas, com o objetivo de identificar os principais fatores e canais de venda que contribuem para esses cancelamentos. Utilizando abordagens de inteligência artificial e estatística, buscou-se não apenas compreender a dinâmica dos cancelamentos, mas também desenvolver ferramentas para a priorização de ações corretivas e a otimização de processos, minimizando perdas financeiras e melhorando a satisfação do cliente.

Motivo da Análise

Pedidos cancelados representam uma perda direta de receita, desperdício de recursos e um impacto negativo na experiência do cliente e na reputação da marca. A compreensão das razões por trás desses cancelamentos e a identificação dos pontos mais críticos são essenciais para:

-Reduzir custos operacionais.

-Aumentar a retenção de clientes.

-Melhorar a eficiência dos canais de venda e da preparação de pedidos.

-Orientar o desenvolvimento de estratégias de mitigação e cupons de incentivo de forma mais assertiva.

Algoritmos Utilizados

Busca Gulosa: Priorizar os canais de vendas que registram o maior número de cancelamentos, permitindo à empresa concentrar seus esforços de forma estratégica.

Regressão Linear Simples: Investigar a existência de uma relação linear entre o tempo de preparo e o valor total dos pedidos que foram cancelados.

```
1 import numpy as np
2 import json
3 from collections import Counter
4 import matplotlib.pyplot as plt
5 from sklearn.linear_model import LinearRegression
6 import pandas as pd
```

```python
#Carregando os dados do Order_semicolon.csv
try:
    df = pd.read_csv('Order_semicolon.csv', sep=';')
    print("CSV 'Order_semicolon.csv' carregado com sucesso!")
    print(f"Número total de registros: {len(df)}")
except FileNotFoundError:
    print("ERRO: Arquivo 'Order_semicolon.csv' não encontrado. Certifique-se de que ele foi carregado no Colab.")
    data_example = {
        'id': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
        'saleschannel': ['IFOOD', 'WHATSAPP', 'IFOOD', 'SITE', 'IFOOD', 'ANOTAAI', 'WHATSAPP', 'IFOOD', 'SITE', 'ANOTAAI'],
        'status': ['CONCLUDED', 'CANCELED', 'CONCLUDED', 'CANCELED', 'CANCELED', 'DISPATCHED', 'CANCELED', 'CONCLUDED', 'PE
        'preparationtime': [30, 45, 20, 60, 35, 15, 50, 25, 40, 55],
        'totalamount': [50.0, 75.0, 40.0, 100.0, 60.0, 30.0, 80.0, 45.0, 70.0, 90.0]
    }
    df = pd.DataFrame(data_example)
    print("Usando dados de exemplo para demonstração.")

df['preparationTime'] = pd.to_numeric(df['preparationTime'], errors='coerce')
df['totalAmount'] = pd.to_numeric(df['totalAmount'], errors='coerce')

# Filtrar apenas os pedidos cancelados
pedidos_cancelados = df[df['status'] == 'CANCELED'].copy()
print(f"Número de pedidos cancelados encontrados: {len(pedidos_cancelados)}")

cancelamentos_por_canal = pedidos_cancelados['salesChannel'].value_counts()
print("\nFrequência de Cancelamentos por Canal de Vendas:")
print(cancelamentos_por_canal)
```

```python
class Vertice:
  def __init__(self, rotulo, distancia_objetivo):
    self.rotulo = rotulo
    self.visitado = False
    self.distancia_objetivo = distancia_objetivo
    self.adjacentes = []

  def adiciona_adjacente(self, adjacente):
    self.adjacentes.append(adjacente)

  def mostra_adjacentes(self):
    for i in self.adjacentes:
      print(i.vertice.rotulo, i.custo)

class Adjacente:
  def __init__(self, vertice, custo):
    self.vertice = vertice
    self.custo = custo

class VetorOrdenado:

  def __init__(self, capacidade):
    self.capacidade = capacidade
    self.ultima_posicao = -1
    self.valores = np.empty(self.capacidade, dtype=object)

  def insere(self, vertice):
    if self.ultima_posicao == self.capacidade - 1:
      return

    posicao = 0

    for i in range(self.ultima_posicao + 1):
      posicao = i
      if self.valores[i].distancia_objetivo > vertice.distancia_objetivo:
        break
      if i == self.ultima_posicao:
        posicao = i + 1

    x = self.ultima_posicao
    while x >= posicao:
      self.valores[x + 1] = self.valores[x]
      x -= 1

    self.valores[posicao] = vertice
    self.ultima_posicao += 1

  def imprime(self):
    if self.ultima_posicao == -1:
      print('O vetor está vazio')
    else:
      for i in range(self.ultima_posicao + 1):
        print(f'{i} - {self.valores[i].rotulo} - Distância/Prioridade: {self.valores[i].distancia_objetivo} (Frequência C
```

```
class Gulosa:
  def __init__(self, objetivo):
    self.objetivo = objetivo
    self.encontrado = False

  def buscar(self, atual):
    print('\n-------')
    print('Atual (Nó em análise): {}'.format(atual.rotulo))
    atual.visitado = True

    if atual == self.objetivo:
      self.encontrado = True
    else:
      canais_a_avaliar = [adj.vertice for adj in atual.adjacentes if not adj.vertice.visitado]
      if not canais_a_avaliar:
          return
      vetor_ordenado = VetorOrdenado(len(canais_a_avaliar))
      for vertice_canal in canais_a_avaliar:
        vetor_ordenado.insere(vertice_canal)
      vetor_ordenado.imprime()

      if vetor_ordenado.ultima_posicao != -1:
        proxima_prioridade = vetor_ordenado.valores[0]
        proxima_prioridade.visitado = True
        print(f'\n-> Próxima maior prioridade selecionada: {proxima_prioridade.rotulo} (Frequência Cancelamento:
        self.buscar(proxima_prioridade)
      else:
        print('Nenhum próximo item a ser priorizado.')

vertices_canais = []
for canal, count in cancelamentos_por_canal.items():
    vertices_canais.append(Vertice(canal, -count))

start_analysis_node = Vertice('Análise de Cancelamentos', 0)
for v_canal in vertices_canais:
    start_analysis_node.adiciona_adjacente(Adjacente(v_canal, 1))
objective_node = Vertice('Análise Concluída', 0)
busca_gulosa_cancelamento = Gulosa(objective_node)
busca_gulosa_cancelamento.buscar(start_analysis_node)
print("\nOrdem de Prioridade dos Canais de Vendas com Cancelamentos (do mais ao menos crítico):")
vetor_final_prioridade = VetorOrdenado(len(vertices_canais))
for v_canal in vertices_canais:
    vetor_final_prioridade.insere(v_canal)
vetor_final_prioridade.imprime()
```

```
df_regressao = pedidos_cancelados.dropna(subset=['preparationTime', 'totalAmount']).copy()

if df_regressao.empty:
    print("\nNão há dados suficientes ou válidos para realizar a regressão linear após a filtragem de pedidos cancelados e
else:
    X = df_regressao['preparationTime'].values.reshape(-1, 1)
    y = df_regressao['totalAmount'].values
    model = LinearRegression()
    model.fit(X, y)
    y_pred = model.predict(X)
    plt.figure(figsize=(10, 6))
    plt.scatter(X, y, color='blue', label='Pedidos Cancelados', alpha=0.6)
    plt.plot(X, y_pred, color='red', linewidth=2, label=f'Linha de Regressão (y = {model.coef_[0]:.2f}x + {model.intercept
    plt.title('Relação entre Tempo de Preparo e Valor Total em Pedidos Cancelados')
    plt.xlabel('Tempo de Preparo (minutos)')
    plt.ylabel('Valor Total (R$)')Ss
    plt.legend()
    plt.grid(True)
    plt.show()
```

Resultados da Análise-Algoritmo da busca gulosa

Priorização de Canais de Venda: A aplicação da Busca Gulosa revelou a seguinte ordem de prioridade para os canais de vendas com maior incidência de cancelamentos (do mais crítico ao menos):

EPADOCA -55 cancelamentos

ANOTAAI - 48 cancelamentos

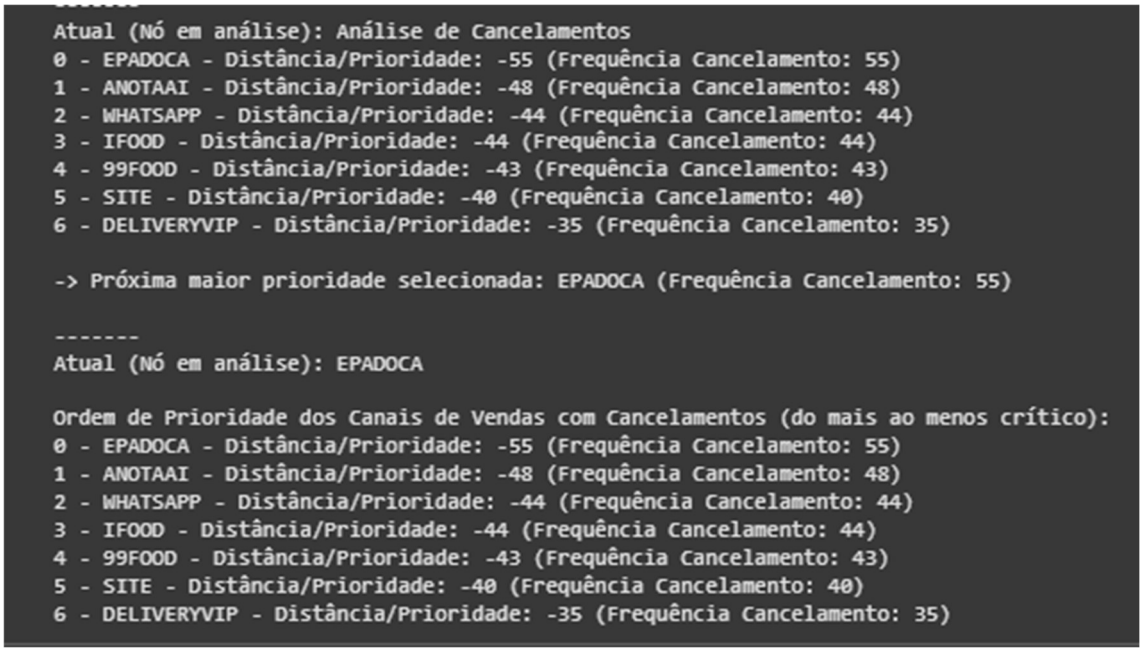
WHATSAPP -44 cancelamentos

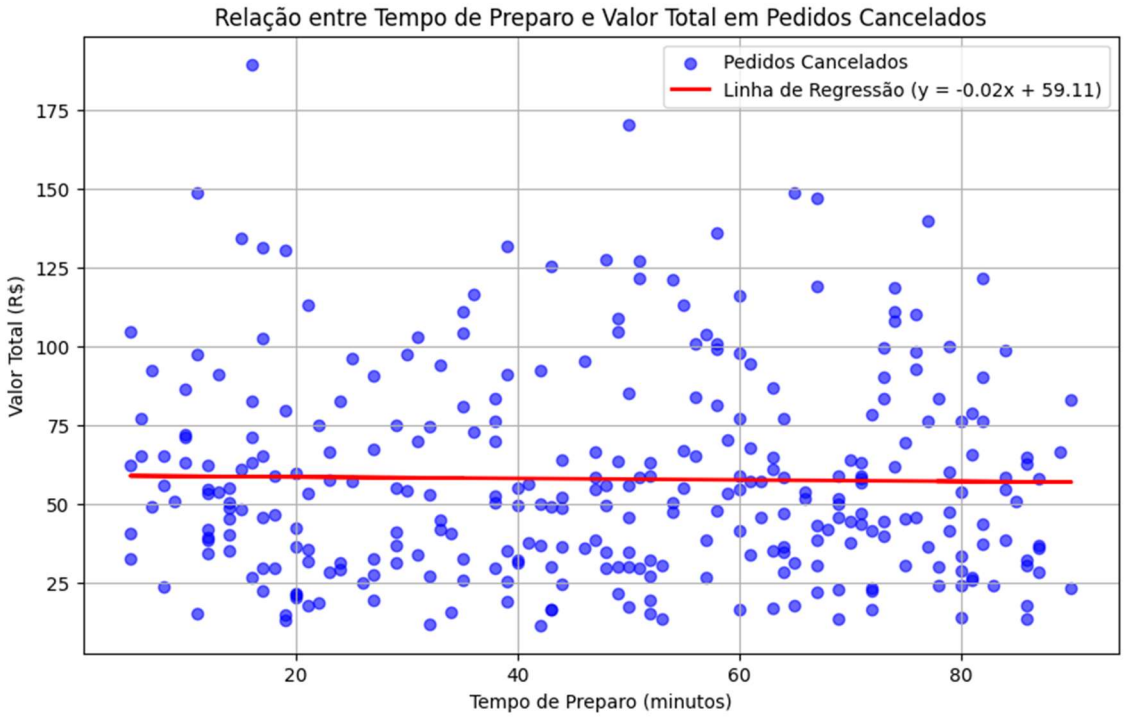
IFOOD -44 cancelamentos

99FOOD -43 cancelamentos

SITE -40 cancelamentos

DELIVERYVIP (35 cancelamentos) Este resultado aponta o canal EPADOCA como o principal foco de atenção imediata.





Conclusão

Este projeto demonstrou a aplicação prática da Busca Gulosa para a priorização de problemas e da Regressão Linear para a compreensão de relações entre variáveis em pedidos cancelados.

Os resultados indicam a necessidade de focar os esforços nos canais de vendas mais críticos e de expandir a análise para outros fatores de cancelamento, além do tempo de preparo. A abordagem gulosa provou ser eficaz para a identificação rápida das áreas de maior impacto, permitindo uma resposta ágil e direcionada na otimização da operação de entregas.