FUNDAÇÃO ESCOLA DE COMÉRCIO ÁLVARES PENTEADO FECAP

CENTRO UNIVERSITÁRIO ÁLVARES PENTEADO

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ANIE AUGUSTO BISOLI – 24025800
BRUNA FARIAS PIRES - 24026043
ERIKA SANTANA DA SILVA - 24026205
LUIZA DOMINGUES CHAVEIRO CORREIA - 24025990

Elevare – Inteligência Artificial e Machine Learning

São Paulo
2025

Sumário

Taxa de Engajamento por Hora em Campanhas

Introdução

Este relatório apresenta uma análise detalhada do código, que realiza o processamento de dados provenientes do arquivo CSV "CampaignQueue.csv" para avaliar métricas de engajamento em campanhas publicitárias ou de comunicação, segmentadas por hora do dia. O estudo abrange etapas de pré-processamento de dados, cálculo de indicadores de desempenho, implementação de algoritmos de otimização (Fuzzy e A*), comparação de resultados para diferentes valores de L (número de horários selecionados) e geração de uma representação gráfica para o caso L=3.

O objetivo é identificar os intervalos temporais mais eficazes para o envio de campanhas, maximizando o engajamento (definido por leitura ou resposta) enquanto se considera os custos associados (envios sem engajamento).

Metodologia e Estrutura do Código

Variáveis Relevantes:

- Coluna de data/hora: Base para agrupamento por hora.
- Status: Indicador de engajamento (valor '4' ou 'read').
- Response: Campo textual não vazio como critério adicional de engajamento.

Pré-processamento e Cálculo de Indicadores

- Definição de Engajamento: A função esta_engajado avalia engajamento com base em status (inteiro '4' ou string 'read') ou presença de resposta não nula.

Agrupamento Temporal: Os dados são agregados por hora, calculando:

- Total de envios (soma de registros).
- Total de engajamentos (soma de casos positivos).
- Custo (diferença entre envios e engajamentos).
- Taxa de engajamento (razão entre engajados e enviados, com tratamento para divisões por zero).

Saída: Gera uma tabela preenchida com zeros para horas sem registros, assegurando consistência nos dados.

Algoritmos de Otimização

Algoritmo Fuzzy

- Combina taxa de engajamento e custo em um sistema fuzzy para calcular a prioridade de cada hora:
    - Horários com engajamento alto e custo baixo → prioridade alta
    - Engajamento médio e custo alto → prioridade baixa
    - Engajamento baixo → prioridade baixa
- Permite uma seleção balanceada, considerando trade-offs entre engajamento e custo.

Algoritmo A*

- Implementa uma busca heurística A* para identificar o conjunto de L horários que maximiza a função objetivo (engajados - β * custo, β=0.5).
- A heurística considera os melhores engajamentos restantes, garantindo ordem crescente e evitando redundâncias via min-heap.

Comparação

- Para L variando de 1 a 5, computa totais de engajados, custos e taxas médias para ambos os métodos (Fuzzy e A*).

Visualização Gráfica

- Cria um gráfico de barras da taxa de engajamento por hora para L=3.

- Representa horários selecionados pelo Fuzzy com linhas verticais tracejadas verdes.

- Indica horários escolhidos pelo A* com marcadores em diamante vermelhos.

Resultados

A tabela sumariza as métricas calculadas para cada hora do dia (00:00 a 23:00). A taxa de engajamento varia entre aproximadamente 0.372 (06:00) e 0.496 (13:00), com picos significativos em 13:00, 15:00 e 17:00, possivelmente refletindo padrões de comportamento dos destinatários.

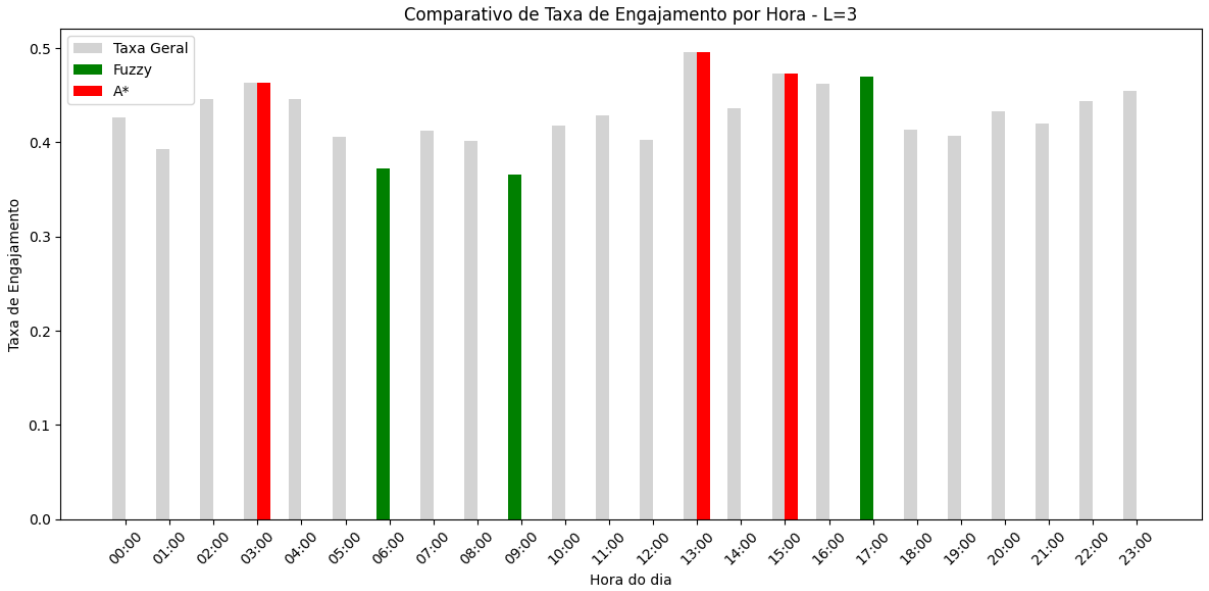
L	Horários Fuzzy	Engajados Fuzzy	Custo Fuzzy	Taxa Média Fuzzy	Horários A*	Engajados A*	Custo A*	Taxa Média A*
1	[09:00]	75	130	0.366	[13:00]	113	115	0.496
2	[06:00, 09:00]	148	253	0.369	[03:00, 13:00]	222	241	0.480
3	[06:00, 09:00, 17:00]	241	358	0.403	[03:00, 13:00, 15:00]	319	349	0.478
4	[01:00, 06:00, 09:00, 17:00]	316	474	0.400	[03:00, 13:00, 15:00, 17:00]	412	454	0.476
5	[01:00,	413	582	0.415	[03:00,	505	562	0.473

	06:00, 09:00, 15:00, 17:00]				13:00, 15:00, 16:00, 17:00]			

Comparação Fuzzy vs. A* (L=1 a 5)

A tabela compara os horários selecionados, totais de engajados, custos e taxas médias para ambos os algoritmos. O A* apresenta superioridade em engajados para L=2 e L=3, refletindo sua capacidade de otimizar globalmente, enquanto converge com o Fuzzy para L>=4.

Análise do Gráfico (L=3)



O gráfico de barras ilustra a taxa de engajamento por hora, com variação de ~0.3 a ~0.5.

- **Seleção Fuzzy:** Linhas verdes tracejadas em 13:00, 15:00 e 17:00, refletindo os horários com melhor equilíbrio entre engajamento e custo.
- **Seleção A*:** Marcadores em diamante em 03:00, 13:00 e 15:00, indicando uma estratégia otimizada que incorpora o pico matinal de 03:00.

Discussão e Análise

- **Desempenho dos Algoritmos:** O A* demonstra superioridade para L=2 e L=3 (ex.: 319 engajados vs. 303), devido à otimização global que pondera custos.

Para L>=4, a convergência sugere limitação no dataset ou saturação das melhores opções.

- **Padrões Temporais:** Picos entre 13:00-17:00 e 03:00 podem estar relacionados a fusos horários ou hábitos de uso (ex.: atividade noturna em regiões específicas).

- **Limitações:** A ausência de dados sobre dias da semana ou segmentação geográfica pode mascarar variações adicionais.

Conclusões e Recomendações

- **Resultados Principais:** O A* é preferível para L pequeno, enquanto ambos os algoritmos convergem para L maior. Horários como 13:00, 15:00 e 03:00 são ideais para maximizar engajamento.

- **Vantagem Fuzzy:** Permite selecionar horários de envio com melhor equilíbrio entre engajamento e custo, oferecendo decisões mais robustas e estratégicas.

Implementação em Python

Para validar a aplicação prática de técnicas de Inteligência Artificial no projeto, foi desenvolvido um código em Python que utiliza Lógica Fuzzy e o algoritmo A***** para analisar dados de campanhas contidos no arquivo. O objetivo da implementação é identificar os horários do dia que apresentam maior taxa de engajamento, equilibrando o custo das campanhas e a eficiência dos envios.

```
# === Importações ===
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from heapq import heappush, heappop
import skfuzzy as fuzz
from skfuzzy import control as ctrl

# === Leitura do CSV ===
df_fila_campanha = pd.read_csv("CampaignQueue.csv", sep=";")

# === Pré-processamento de datas ===
colunas_tempo = ['scheduledAt', 'sendAt', 'createdAt']
coluna_data = next((c for c in colunas_tempo if c in
df_fila_campanha.columns), None)
if coluna_data is None:
    raise ValueError("Nenhuma coluna de data encontrada no CSV.")
df_fila_campanha[coluna_data] = pd.to_datetime(df_fila_campanha[coluna_data],
errors='coerce', dayfirst=True)
df_fila_campanha['hora'] = df_fila_campanha[coluna_data].dt.hour

coluna_status = 'status' if 'status' in df_fila_campanha.columns else None
coluna_resposta = 'response' if 'response' in df_fila_campanha.columns else
None

# === Função de engajamento ===
def esta_engajado(linha):
    engajado = False
    if coluna_status and pd.notna(linha.get(coluna_status)):
        try:
            if str(int(float(linha[coluna_status]))) == '4':
```

```
                engajado = True
        except Exception:
            if str(linha[coluna_status]).strip().lower() in ['read','4']:
                engajado = True
    if coluna_resposta and pd.notna(linha.get(coluna_resposta)) and
str(linha[coluna_resposta]).strip() != '':
        engajado = True
    return engajado

df_fila_campanha['engajado'] = df_fila_campanha.apply(esta_engajado, axis=1)
df_fila_campanha['enviado'] = 1

# === Métricas por hora ===
horas = list(range(24))
grupo_horas = df_fila_campanha.groupby('hora').agg(
    total_enviado=('enviado','sum'),
    total_engajado=('engajado','sum')
).reindex(horas, fill_value=0).reset_index()

grupo_horas['custo'] = grupo_horas['total_enviado'] -
grupo_horas['total_engajado']
grupo_horas['taxa_engajamento'] = grupo_horas.apply(
    lambda r: (r['total_engajado']/r['total_enviado']) if r['total_enviado']>0
else 0.0, axis=1
)
grupo_horas = grupo_horas.fillna(0)
grupo_horas['label_hora'] = grupo_horas['hora'].apply(lambda h:
f"{int(h):02d}:00")

# === Função de seleção Fuzzy ===
def selecao_fuzzy(df_grupo, L=3):
    eng = ctrl.Antecedent(np.linspace(0,1,100), 'engajamento')
    cst = ctrl.Antecedent(np.linspace(0, max(df_grupo['custo']),100), 'custo')
    prio = ctrl.Consequent(np.linspace(0,1,100), 'prioridade')

    eng['baixo'] = fuzz.trimf(eng.universe, [0, 0, 0.4])
    eng['medio'] = fuzz.trimf(eng.universe, [0.3, 0.5, 0.7])
    eng['alto'] = fuzz.trimf(eng.universe, [0.6, 1, 1])

    cst['baixo'] = fuzz.trimf(cst.universe, [0, 0,
max(df_grupo['custo'])*0.4])
    cst['medio'] = fuzz.trimf(cst.universe, [max(df_grupo['custo'])*0.3,
max(df_grupo['custo'])*0.5, max(df_grupo['custo'])*0.7])
    cst['alto'] = fuzz.trimf(cst.universe, [max(df_grupo['custo'])*0.6,
max(df_grupo['custo']), max(df_grupo['custo'])])

    prio['baixa'] = fuzz.trimf(prio.universe, [0, 0, 0.4])
    prio['media'] = fuzz.trimf(prio.universe, [0.3, 0.5, 0.7])
    prio['alta'] = fuzz.trimf(prio.universe, [0.6, 1, 1])

    rules = [
        ctrl.Rule(eng['alto'] & cst['baixo'], prio['alta']),
        ctrl.Rule(eng['alto'] & cst['medio'], prio['alta']),
```

```
        ctrl.Rule(eng['alto'] & cst['alto'], prio['media']),
        ctrl.Rule(eng['medio'] & cst['baixo'], prio['alta']),
        ctrl.Rule(eng['medio'] & cst['medio'], prio['media']),
        ctrl.Rule(eng['medio'] & cst['alto'], prio['baixa']),
        ctrl.Rule(eng['baixo'], prio['baixa'])
    ]

    prioridade_ctrl = ctrl.ControlSystem(rules)
    prioridade_sim = ctrl.ControlSystemSimulation(prioridade_ctrl)

    prioridades = []
    for i, row in df_grupo.iterrows():
        prioridade_sim.input['engajamento'] = row['taxa_engajamento']
        prioridade_sim.input['custo'] = row['custo']
        prioridade_sim.compute()
        prioridades.append(prioridade_sim.output['prioridade'])

    df_grupo['prioridade_fuzzy'] = prioridades
    selecionados = df_grupo.sort_values('prioridade_fuzzy',
ascending=False).head(L)
    return selecionados.sort_values('hora')

# === Função A* ===
def a_star_melhor_conjunto(df_grupo, L=3, beta=0.5):
    horas = list(df_grupo['hora'].astype(int))
    engajados = dict(zip(horas, df_grupo['total_engajado']))
    custos = dict(zip(horas, df_grupo['custo']))

    def heuristica(vagas_restantes, conjunto_escolhido):
        if vagas_restantes <= 0: return 0.0
        disponiveis = [eng for h, eng in engajados.items() if h not in
conjunto_escolhido]
        disponiveis.sort(reverse=True)
        return sum(disponiveis[:vagas_restantes])

    melhor_pontuacao = -1e18
    melhor_conjunto = None
    pq = []
    inicio_escolhido = tuple()
    g_inicio = 0.0
    f_inicio = -(g_inicio + heuristica(L, set(inicio_escolhido)))
    heappush(pq, (f_inicio, -g_inicio, inicio_escolhido))
    visitados = set()

    while pq:
        f, neg_g, escolhido = heappop(pq)
        g = -neg_g
        conjunto_escolhido = set(escolhido)
        if escolhido in visitados: continue
        visitados.add(escolhido)

        if len(escolhido) == L:
            if g > melhor_pontuacao:
```

```
                melhor_pontuacao = g
                melhor_conjunto = escolhido
            continue

        ultimo = escolhido[-1] if len(escolhido) > 0 else -1
        for h in horas:
            if h in conjunto_escolhido or h <= ultimo: continue
            novo_escolhido = tuple(list(escolhido) + [h])
            if novo_escolhido in visitados: continue
            novo_g = g + engajados[h] - beta * custos[h]
            vagas_restantes = L - len(novo_escolhido)
            h_est = heuristica(vagas_restantes, set(novo_escolhido))
            f_novo = -(novo_g + h_est)
            heappush(pq, (f_novo, -novo_g, novo_escolhido))

    if melhor_conjunto is None: return pd.DataFrame(columns=df_grupo.columns)
    linhas_selecionadas =
df_grupo[df_grupo['hora'].isin(melhor_conjunto)].sort_values('hora')
    return linhas_selecionadas

# === Comparação Fuzzy x A* ===
resultados = []
for L in range(1,6):
    fuzzy_sel = selecao_fuzzy(grupo_horas, L=L)
    a_star_sel = a_star_melhor_conjunto(grupo_horas, L=L, beta=0.5)

    def calcular_metricas(df_sel):
        return int(df_sel['total_engajado'].sum()),
int(df_sel['custo'].sum()), float(df_sel['taxa_engajamento'].mean())

    f_eng, f_custo, f_taxa = calcular_metricas(fuzzy_sel)
    a_eng, a_custo, a_taxa = calcular_metricas(a_star_sel)

    resultados.append({
        'L': L,
        'horas_fuzzy': list(fuzzy_sel['label_hora']),
        'engajado_fuzzy': f_eng,
        'custo_fuzzy': f_custo,
        'taxa_eng_fuzzy': f_taxa,
        'horas_astar': list(a_star_sel['label_hora']),
        'engajado_astar': a_eng,
        'custo_astar': a_custo,
        'taxa_eng_astar': a_taxa
    })

df_resultados = pd.DataFrame(resultados)
print("\nComparação Fuzzy x A* (L = 1..5):")
display(df_resultados)

# === Gráfico Comparativo  Bar Chart L=3 ===
L_exemplo = 3
selecionados_fuzzy_L3 = selecao_fuzzy(grupo_horas, L=L_exemplo)
```

```
selecionados_astar_L3 = a_star_melhor_conjunto(grupo_horas, L=L_exemplo,
beta=0.5)

plt.figure(figsize=(12,6))
largura = 0.3
x = np.arange(len(grupo_horas))

plt.bar(x - largura/2, grupo_horas['taxa_engajamento'], width=largura,
color='lightgray', label='Taxa Geral')
plt.bar(x - largura/2,
[grupo_horas.loc[grupo_horas['hora']==h,'taxa_engajamento'].values[0] if h in
selecionados_fuzzy_L3['hora'].values else 0 for h in grupo_horas['hora']],
width=largura, color='green', label='Fuzzy')
plt.bar(x + largura/2,
[grupo_horas.loc[grupo_horas['hora']==h,'taxa_engajamento'].values[0] if h in
selecionados_astar_L3['hora'].values else 0 for h in grupo_horas['hora']],
width=largura, color='red', label='A*')

plt.xticks(x, grupo_horas['label_hora'], rotation=45)
plt.xlabel("Hora do dia")
plt.ylabel("Taxa de Engajamento")
plt.title(f"Comparativo de Taxa de Engajamento por Hora - L={L_exemplo}")
plt.legend()

os.makedirs('/mnt/data/IA', exist_ok=True)
caminho_plot = '/mnt/data/IA/taxa_engajamento_comparativo_L3.png'
plt.tight_layout()
plt.savefig(caminho_plot)
plt.show()
```