

experimento 6: llama - nao rodou - f808a30cf

USER

A base de dados anexada no arquivo dados_50_50.csv possui quatro variáveis: IDENTIFICADOR, CARGO, PERCENT_PRI E DISTANCE_KM. A primeira variável é o código que identifica o funcionário, a segunda apresenta os códigos dos cargos, a terceira é um índice que varia de 0 a 1, sendo o 0 o melhor e, por fim, a última variável é a distância em KM entre o funcionário e o cargo que ele poderá assumir, ou seja, quanto maior a distância, pior. Gostaria que me propusesse uma metodologia para solucionar meu problema de alocação de forma a minimizar o PERCENT_PRI ou a DISTANCE_KM ou os dois juntos. Determine e execute o código para solucionar o problema e indique o custo total minimizado.

BOT

Uma abordagem para resolver esse problema de alocação é utilizar a programação linear, que é uma técnica amplamente usada para otimizar problemas de alocação de recursos. Nesse caso, queremos minimizar o custo total, que pode ser representado pelo `PERCENT_PRI` ou pela `DISTANCE_KM`, ou uma combinação de ambos.

Uma forma de abordar isso é criar um modelo de programação linear que represente o problema. Nesse caso, podemos definir as seguintes variáveis:

- `x_ij`: uma variável binária que indica se o funcionário `i` é alocado ao cargo `j` (1 se sim, 0 se não)

O objetivo é minimizar o custo total, que pode ser representado pela seguinte função:

- `minimize $\sum (i, j) (\text{PERCENT_PRI}_{ij} * x_{ij} + \text{DISTANCE_KM}_{ij} * x_{ij})$ `

Sujeito às seguintes restrições:

- Cada funcionário só pode ser alocado a um cargo: $\sum_j x_{ij} = 1$ para cada `i`
- Cada cargo só pode ser ocupado por um funcionário: $\sum_i x_{ij} = 1$ para cada `j`

Aqui está um exemplo de código em Python que utiliza a biblioteca `pulp` para resolver o problema:

```
```python
import pandas as pd
from pulp import LpMaximize, LpProblem, LpSum, LpVariable, value

Carregar os dados
df = pd.read_csv('dados_50_50.csv')

Criar o modelo
model = LpProblem(name="alocacao", sense=LpMaximize)

Criar as variáveis
x = LpVariable.dicts("x", (df['IDENTIFICADOR'], df['CARGO']), 0, 1, cat='Integer')

Definir o objetivo
model += LpSum([-(df.loc[i, 'PERCENT_PRI'] + df.loc[i, 'DISTANCE_KM']) * x[df.loc[i, 'IDENTIFICADOR']][df.loc[i, 'CARGO']] for i in range(len(df))])

Definir as restrições
for i in df['IDENTIFICADOR'].unique():
 model += LpSum([x[i][j] for j in df['CARGO'].unique()]) == 1
```

```

for j in df['CARGO'].unique():
 model += lpSum([x[i][j] for i in df['IDENTIFICADOR'].unique()]) == 1

Resolver o modelo
status = model.solve()

Imprimir o resultado
if status:
 print("Custo total minimizado:", -value(model.objective))
 for i in df['IDENTIFICADOR'].unique():
 for j in df['CARGO'].unique():
 if x[i][j].varValue:
 print(f"Funcionário {i} alocado ao cargo {j}")
else:
 print("Não foi possível encontrar uma solução ótima")
'''

```

Esse código carrega os dados do arquivo `dados\_50\_50.csv`, cria um modelo de programação linear que representa o problema, e resolve o modelo utilizando a biblioteca `pulp`. O resultado é o custo total minimizado e a alocação de cada funcionário a um cargo.

Lembre-se de que a complexidade do problema pode afetar o tempo de execução do código. Além disso, a qualidade da solução pode depender da escolha dos parâmetros e da formulação do modelo.

Created/Modified files during execution:

- dados\_50\_50.csv

```Executed 1st Code Block

Execution Error: KeyError

Traceback (most recent call last)

Cell In[3], line 11

```

8 model = LpProblem(name="alocacao", sense=LpMaximize)
10 # Criar as variáveis
--> 11 x = LpVariable.dicts("x", (df['IDENTIFICADOR'], df['CARGO']), 0, 1, cat='Integer')
13 # Definir o objetivo
14 model += lpSum([-(df.loc[i, 'PERCENT_PRI'] + df.loc[i, 'DISTANCE_KM']) * x[df.loc[i, 'IDENTIFICADOR']]
[df.loc[i, 'CARGO']] for i in range(len(df))])

```

...

KeyError: 'IDENTIFICADOR'