

experimento 6: gemini - nao rodou - 2d96f53eb

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

Here's a proposed methodology and solution for your allocation problem:

****1. Problem Definition:****

You're dealing with an assignment problem. You want to assign each employee (IDENTIFICADOR) to a single job (CARGO) in a way that minimizes either PERCENT_PRI, DISTANCE_KM, or a combination of both.

****2. Methodology:****

****Separate Optimization:**** If you want to minimize only one of the objectives (PERCENT_PRI or DISTANCE_KM), the Hungarian algorithm is a good choice. It's efficient for solving assignment problems with a single objective.

****Combined Optimization:**** If you want to minimize both, you have a multi-objective optimization problem. There isn't a single "best" solution in this case, but rather a set of Pareto optimal solutions. A Pareto optimal solution is one where you can't improve one objective without worsening the other. One approach is to create a weighted sum objective function. This involves assigning weights to each objective and summing them. The weights reflect the relative importance of each objective.

****3. Chosen Method and Implementation (Weighted Sum):****

Given that you mentioned minimizing "one or both," I'll implement a weighted sum approach, allowing you to tune the weights. This provides flexibility.

```
```python
import pandas as pd
from scipy.optimize import linear_sum_assignment

Load the data
df = pd.read_csv("dados_50_50.csv", sep=";")

Convert ',' to '.' and to numeric
df['PERCENT_PRI'] = pd.to_numeric(df['PERCENT_PRI'].str.replace(',', '.'))
df['DISTANCE_KM'] = pd.to_numeric(df['DISTANCE_KM'].str.replace(',', '.'))

Define weights for the objectives
weight_percent_pri = 0.5 # Example: Equal weight
weight_distance_km = 0.5

Create the cost matrix
cost_matrix = df.pivot(index='IDENTIFICADOR', columns='CARGO', values=['PERCENT_PRI', 'DISTANCE_KM'])

Normalize the objectives (important for weighted sum)
cost_matrix = cost_matrix.apply(lambda x: (x - x.min()) / (x.max() - x.min()))

Calculate the weighted sum cost
cost_matrix['weighted_cost'] = weight_percent_pri * cost_matrix['PERCENT_PRI'] + weight_distance_km * cost_matrix['DISTANCE_KM']
cost_matrix = cost_matrix['weighted_cost'].values

Apply the Hungarian algorithm
```

```
row_ind, col_ind = linear_sum_assignment(cost_matrix)
```

```
Calculate the total minimized cost
```

```
total_cost = cost_matrix[row_ind, col_ind].sum()
```

```
Print the results
```

```
print(f"Total minimized weighted cost: {total_cost}")
```

```
Map the solution back to the original data
```

```
solution = pd.DataFrame({'IDENTIFICADOR': row_ind + 1, 'CARGO': col_ind + 1}) # +1 because indices start from 0
```

```
print("\nOptimal Assignment:")
```

```
print(solution)
```

```
Created/Modified files during execution:
```

```
No files created/modified
```

```
...
```

#### **\*\*4. Explanation and Next Steps:\*\***

The code first loads and preprocesses the data, converting the string representations of numbers to actual numeric values. It then defines weights for each objective. A cost matrix is created and normalized (crucial for the weighted sum to work correctly). The Hungarian algorithm is then used to find the optimal assignment. Finally, the total minimized cost and the assignment are printed.

You can adjust `weight\_percent\_pri` and `weight\_distance\_km` to change the importance of each objective. For example, setting `weight\_percent\_pri` to 1 and `weight\_distance\_km` to 0 would minimize only PERCENT\_PRI. Experiment with different weights to find a solution that best suits your needs. If you have additional constraints or requirements, let me know, and I can refine the code.

Execution Error: ValueError

Traceback (most recent call last)

```
/tmp/ipykernel_1655/375967982.py in ?()
```

```
18 # Normalize the objectives (important for weighted sum)
```

```
19 cost_matrix = cost_matrix.apply(lambda x: (x - x.min()) / (x.max() - x.min()))
```

```
20
```

```
21 # Calculate the weighted sum cost
```

```
---> 22 cost_matrix['weighted_cost'] = weight_percent_pri * cost_matrix[('PERCENT_PRI',)] + weight_distance_km *
cost_matrix[('DISTANCE_KM',)]
```

```
23 cost_matrix = cost_matrix['weighted_cost'].values
```

```
24
```

```
25 # Apply the Hungarian algorithm
```

```
/opt/venv/lib/python3.12/site-packages/pandas/core/frame.py in ?(self, key, value)
```

```
4297 self._setitem_frame(key, value)
```

```
4298 elif isinstance(key, (Series, np.ndarray, list, Index)):
```

```
4299 self._setitem_array(key, value)
```

```
4300 elif isinstance(value, DataFrame):
```

```
-> 4301 self._set_item_frame_value(key, value)
```

```
4302 elif (
```

```
4303 is_list_like(value)
```

```
4304 and not self.columns.is_unique
```

```
/opt/venv/lib/python3.12/site-packages/pandas/core/frame.py in ?(self, key, value)
```

```
4455
```

```
4456 return self.isetitem(locs, value)
```

```
4457
```

```
4458 if len(value.columns) > 1:
```

```
-> 4459 raise ValueError(
```

```
4460 "Cannot set a DataFrame with multiple columns to the single "
```

```
4461 f"column {key}"
```

```
4462)
```

ValueError: Cannot set a DataFrame with multiple columns to the single column weighted\_cost