

experimento 8: run 1 - não rodou - 15c17c8213

USER

A base de dados anexada 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.

Resolva este problema usando o método húngaro. Execute o código.

BOT

O problema descrito é um problema de alocação que pode ser resolvido utilizando o **método húngaro**. Este método é designed to solve assignment problems where the goal is to minimize the cost of assigning tasks (or jobs) to agents. In this case, the cost matrix can be based on either `PERCENT_PRI`, `DISTANCE_KM`, or a combination of both.

Steps to Solve the Problem

1. **Define the Cost Matrix**:

- If you want to minimize `PERCENT_PRI`, the cost matrix will be based on this variable.
- If you want to minimize `DISTANCE_KM`, the cost matrix will be based on this variable.
- If you want to minimize both, you can create a weighted combination of the two variables (e.g., `cost = w1 * PERCENT_PRI + w2 * DISTANCE_KM`).

2. **Apply the Hungarian Algorithm**:

- The Hungarian algorithm will find the optimal assignment that minimizes the total cost.

3. **Implementation**:

- I'll implement the Hungarian algorithm using Python's `scipy.optimize.linear_sum_assignment` function, which is an efficient implementation of the Hungarian method.

Implementation

Below is the Python code to solve the problem using the Hungarian algorithm. The code will:

- Read the data.
- Create a cost matrix based on the chosen criteria.
- Solve the assignment problem using the Hungarian algorithm.

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

Load the data
data = """IDENTIFICADOR;CARGO;PERCENT_PRI;DISTANCE_KM
1;1;0.85417;1414.13
1;2;0.85417;1414.13
1;3;0.85417;1414.13
1;4;0.85417;1414.13
1;5;0.85417;1414.13
1;6;0.85417;1414.13
1;7;0.85417;1414.13
1;8;0.85417;1414.13
1;9;0.85417;1414.13
1;10;0.85417;1414.13
1;11;0.85417;1414.13
1;12;0.85417;1414.13
1;13;0.85417;1414.13
1;14;0.85417;1414.13
1;15;0.85417;1414.13
1;16;0.85417;1414.13

```

```

1;17;0.85417;1414.13
1;18;0.85417;1414.13
1;19;0.85417;1414.13
1;20;0.85417;1414.13
1;21;0.85417;1414.13
1;22;0.85417;1414.13
1;23;0.85417;1414.13
1;24;0.85417;1414.13
1;25;0.85417;1414.13
1;26;0.85417;1414.13
1;27;0.85417;1414.13
1;28;0.85417;1414.13
1;29;0.85417;1414.13
1;30;0.85417;1414.13
1;55;0.89583;1472.53
1;56;0.89583;1472.53
1;57;0.89583;1472.53
1;58;0.89583;1472.53
1;59;0.89583;1472.53
1;60;0.89583;1472.53
1;61;0.89583;1472.53
1;62;0.89583;1472.53
1;63;0.89583;1472.53
1;64;0.89583;1472.53
1;65;0.89583;1472.53
1;66;0.89583;1472.53
1;67;0.89583;1472.53
1;68;0.89583;1472.53
1;69;0.89583;1472.53
1;70;0.89583;1472.53
1;71;0.89583;1472.53
1;72;0.89583;1472.53
1;73;0.89583;1472.53
1;74;0.89583;1472.53
"""

```

```
Parse the data
```

```
from io import StringIO
```

```
df = pd.read_csv(StringIO(data), sep=";")
```

```
Normalize the data (optional, for combining criteria)
```

```
df['PERCENT_PRI'] = df['PERCENT_PRI'].str.replace(',', '.').astype(float)
```

```
df['DISTANCE_KM'] = df['DISTANCE_KM'].str.replace(',', '.').astype(float)
```

```
Create a cost matrix
```

```
Example: Minimize both PERCENT_PRI and DISTANCE_KM with equal weights
```

```
w1, w2 = 0.5, 0.5
```

```
df['COST'] = w1 * df['PERCENT_PRI'] + w2 * (df['DISTANCE_KM'] / df['DISTANCE_KM'].max())
```

```
Pivot the data to create a cost matrix
```

```
cost_matrix = df.pivot(index='IDENTIFICADOR', columns='CARGO', values='COST').fillna(1e9).values
```

```
Solve the assignment problem
```

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

```
Output the results
```

```
assignments = list(zip(row_ind + 1, col_ind + 1)) # Adding 1 to match original IDs
```

```
print("Optimal Assignments (IDENTIFICADOR -> CARGO):")
```

```
for ident, cargo in assignments:
```

```
 print(f"IDENTIFICADOR {ident} -> CARGO {cargo}")
```

```
...
```

```
Explanation of the Code
```

```
1. **Data Loading**:
```

```
- The data is loaded into a pandas DataFrame.
```

```
- The `PERCENT_PRI` and `DISTANCE_KM` columns are converted to numeric values.
```

```
2. **Cost Matrix**:
```

```
- A weighted combination of `PERCENT_PRI` and `DISTANCE_KM` is used to create the cost matrix.
```

- The weights `w1` and `w2` can be adjusted based on the importance of each criterion.

### 3. **Hungarian Algorithm**:

- The `linear\_sum\_assignment` function from `scipy.optimize` is used to solve the assignment problem.

### 4. **Output**:

- The optimal assignments are printed, showing which `IDENTIFICADOR` (employee) is assigned to which `CARGO` (job).

### ### Next Steps

- Run the code to get the optimal assignments.
- Adjust the weights `w1` and `w2` if you want to prioritize one criterion over the other.
- If you have a larger dataset, ensure the cost matrix is not too sparse or large for computational efficiency.

Execution Error: AttributeError

Traceback (most recent call last)

Cell In[3], line 63

```
60 df = pd.read_csv(StringIO(data), sep=";")
62 # Normalize the data (optional, for combining criteria)
--> 63 df['PERCENT_PRI'] = df['PERCENT_PRI'].str.replace(',', '.').astype(float)
64 df['DISTANCE_KM'] = df['DISTANCE_KM'].str.replace(',', '.').astype(float)
66 # Create a cost matrix
67 # Example: Minimize both PERCENT_PRI and DISTANCE_KM with equal weights
```

...

AttributeError: Can only use .str accessor with string values!