

experimento 8: qwen viajou - ade18a88c

USER

o arquivo dados_50_50.csv anexo possui 4 colunas: 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. Resolva o problema de alocação codificando um algoritmo com o método húngaro de forma a minimizar o PERCENT_PRI ou a DISTANCE_KM ou os dois juntos. E seguida execute o código

BOT

To solve the allocation problem using the Hungarian algorithm, we need to decide which criteria to minimize: `PERCENT_PRI`, `DISTANCE_KM`, or a combination of both. Since both criteria are important, we can create a composite cost matrix that combines these two metrics. One way to do this is to normalize both metrics and then sum them to create a single cost value for each pair of `IDENTIFICADOR` and `CARGO`.

Here's a step-by-step approach to solve this problem:

1. ****Load the Data****: Read the data from the CSV file.
2. ****Normalize the Metrics****: Normalize `PERCENT_PRI` and `DISTANCE_KM` to a scale between 0 and 1.
3. ****Create a Composite Cost Matrix****: Sum the normalized `PERCENT_PRI` and `DISTANCE_KM` to create a composite cost matrix.
4. ****Apply the Hungarian Algorithm****: Use the Hungarian algorithm to find the optimal allocation that minimizes the composite cost.
5. ****Output the Results****: Print the optimal allocation.

Let's implement this in Python:

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

# Step 1: Load the Data
data = pd.read_csv('dados_50_50.csv', sep=';', decimal=',')

# Step 2: Normalize the Metrics
percent_pri_normalized = (data['PERCENT_PRI'] - data['PERCENT_PRI'].min()) / (data['PERCENT_PRI'].max() - data['PERCENT_PRI'].min())
distance_km_normalized = (data['DISTANCE_KM'] - data['DISTANCE_KM'].min()) / (data['DISTANCE_KM'].max() - data['DISTANCE_KM'].min())

# Step 3: Create a Composite Cost Matrix
composite_cost = percent_pri_normalized + distance_km_normalized

# Reshape the composite cost into a matrix
num_identificadores = data['IDENTIFICADOR'].nunique()
num_cargos = data['CARGO'].nunique()

composite_cost_matrix = composite_cost.values.reshape(num_identificadores, num_cargos)

# Step 4: Apply the Hungarian Algorithm
row_ind, col_ind = linear_sum_assignment(composite_cost_matrix)

# Step 5: Output the Results
optimal_allocation = pd.DataFrame({
    'IDENTIFICADOR': data['IDENTIFICADOR'].unique()[row_ind],
    'CARGO': data['CARGO'].unique()[col_ind],
    'PERCENT_PRI': composite_cost_matrix[row_ind, col_ind].flatten(),
    'DISTANCE_KM': data['DISTANCE_KM'].values.reshape(num_identificadores, num_cargos)[row_ind, col_ind].flatten()
})

print(optimal_allocation)

# Created/Modified files during execution:
print("dados_50_50.csv")
```

Explanation:

- ****Normalization****: We normalize both `PERCENT_PRI` and `DISTANCE_KM` to ensure they are on the same scale. This is important because `PERCENT_PRI` ranges from 0 to 1, while `DISTANCE_KM` can vary widely.
- ****Composite Cost Matrix****: By summing the normalized values, we create a single cost matrix that reflects both criteria.
- ****Hungarian Algorithm****: The `linear_sum_assignment` function from `scipy.optimize` is used to find the optimal assignment that minimizes the total cost.
- ****Output****: The results are printed in a DataFrame showing the optimal allocation of `IDENTIFICADOR` to `CARGO` along with the

corresponding `PERCENT_PRI` and `DISTANCE_KM`.

Note:

- The code assumes that the CSV file is named `dados_50_50.csv` and is located in the current working directory.
- The Hungarian algorithm is efficient for solving assignment problems where the goal is to minimize the total cost.

Would you like to proceed with the execution of this code? If so, please ensure the CSV file is available in the correct directory.

Execution Result:

	IDENTIFICADOR	CARGO	PERCENT_PRI	DISTANCE_KM
0	1	3	1.208408	1414.13
1	2	2	1.288682	2934.83
2	3	14	1.515824	3415.07
3	4	4	0.839757	1758.24
4	5	55	0.817450	789.76
5	6	5	1.863692	3355.79
6	7	56	1.013386	742.76
7	8	6	1.390309	1604.70
8	9	57	1.013777	670.80
9	10	7	1.924162	3590.74
10	11	70	0.746142	1394.51
11	12	9	0.024067	93.51
12	13	10	0.633933	1575.76
13	14	11	1.812549	3266.16
14	15	58	1.042316	1487.11
15	16	59	1.448493	1830.77
16	17	60	1.336212	1394.51
17	18	13	1.657677	2991.58
18	19	71	1.358910	1394.51
19	20	28	1.211720	2894.83
20	21	61	1.449481	1919.11
21	22	16	1.429826	1758.24
22	23	18	1.894353	3693.04
23	24	17	0.629734	765.83
24	25	25	1.043053	1137.24
25	26	62	0.024117	0.00
26	27	19	1.066712	1758.24
27	28	20	1.503106	3590.74
28	29	21	1.339053	1758.24
29	30	22	1.058423	579.73
30	31	23	1.002093	801.75
31	32	63	1.313524	1394.51
32	33	64	1.218374	1113.01
33	34	24	1.017566	1831.83
34	35	65	1.336212	1394.51
35	36	73	1.176612	1290.59
36	37	66	1.377413	1552.73
37	38	26	0.600462	123.06
38	39	27	1.290117	1391.76
39	40	67	1.290180	1472.82
40	41	29	1.022432	1382.29
41	42	69	0.644953	1815.15
42	43	68	0.949669	333.52
43	44	30	0.992849	765.83
44	45	15	1.163365	1604.70
45	46	8	1.575822	2455.42
46	47	12	0.997656	1137.24
47	48	72	1.313524	1394.51
48	49	1	0.808797	1637.95
49	50	74	1.109268	1394.51

dados_50_50.csv

Attachments:

- dados_50_50.csv (text/csv)