experimento7: 4o - 100x100 nao rodou - e5ffdb3cf

USER

A base de dados anexada no arquivo dados_100_100.csv possui quatro variáveis: IDENTIFICADOR, CARGO, PERCENT_PRI E DISTANCE_KM. A primeira variável é o codigo que identifica o funcionário, a segunda apresenta os códigos dos cargos, a terceira é um indice 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

To solve the problem of allocating employees to positions while minimizing either `PERCENT_PRI`, `DISTANCE_KM`, or both together, we can approach this as an **optimization problem**. Specifically, this is a **linear assignment problem** where we aim to minimize the cost of assigning employees to positions based on the given criteria.

### Step 1: Problem Formulation
The problem can be formulated as follows:
1. **Decision Variables**:
   - Let \(x_{ij} \) be a binary variable where \(x_{ij} = 1 \) if employee \(i \) is assigned to position \(j \), and \(x_{ij} = 0 \) otherwise.

2. **Objective Function**:
   - Minimize the total cost of assignment. The cost can be based on:
     - `PERCENT_PRI` alone.
     - `DISTANCE_KM` alone.
     - A weighted combination of both (e.g., \(\text{Cost} = w_1 \cdot \text{PERCENT_PRI} + w_2 \cdot \text{DISTANCE_KM} \)).

3. **Constraints**:
   - Each employee is assigned to exactly one position.
   - Each position is assigned to exactly one employee.

### Step 2: Methodology
We can use the **Hungarian Algorithm** or a **Linear Programming (LP)** solver to solve this assignment problem. Python provides libraries like `scipy.optimize` and `pulp` to handle such problems efficiently.

### Step 3: Implementation
We will:
1. Parse the data from the CSV file.
2. Construct a cost matrix based on the chosen objective (e.g., `PERCENT_PRI`, `DISTANCE_KM`, or a combination).
3. Solve the assignment problem using an optimization library.
4. Output the minimized cost and the optimal assignment.

Here is the Python code to implement the solution:

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

# Step 1: Load the data
file_name = "dados_100_100.csv"
data = pd.read_csv(file_name, sep=";", header=None, names=["IDENTIFICADOR", "CARGO", "PERCENT_PRI", "DISTANCE_KM"])

# Step 2: Create the cost matrix
# Normalize PERCENT_PRI and DISTANCE_KM to bring them to the same scale
data["PERCENT_PRI"] = data["PERCENT_PRI"].str.replace(",", ".").astype(float)
data["DISTANCE_KM"] = data["DISTANCE_KM"].str.replace(",", ".").astype(float)
```

```
# Normalize the columns
data["PERCENT_PRI_NORM"] = data["PERCENT_PRI"] / data["PERCENT_PRI"].max()
data["DISTANCE_KM_NORM"] = data["DISTANCE_KM"] / data["DISTANCE_KM"].max()

# Define weights for the combined objective
w1 = 0.5  # Weight for PERCENT_PRI
w2 = 0.5  # Weight for DISTANCE_KM

# Compute the combined cost
data["COMBINED_COST"] = w1 * data["PERCENT_PRI_NORM"] + w2 * data["DISTANCE_KM_NORM"]

# Pivot the data to create a cost matrix
cost_matrix = data.pivot(index="IDENTIFICADOR", columns="CARGO",
values="COMBINED_COST").fillna(1e6).values

# Step 3: Solve the assignment problem
row_ind, col_ind = linear_sum_assignment(cost_matrix)

# Step 4: Calculate the total minimized cost
total_cost = cost_matrix[row_ind, col_ind].sum()

# Step 5: Output the results
assignments = list(zip(row_ind, col_ind))
print("Optimal Assignments (Employee -> Position):", assignments)
print("Total Minimized Cost:", total_cost)

# Save the results to a file
output_file = "optimal_assignments.csv"
results = pd.DataFrame(assignments, columns=["Employee", "Position"])
results["Cost"] = cost_matrix[row_ind, col_ind]
results.to_csv(output_file, index=False)
print(f"Results saved to {output_file}")
```

### Step 4: Explanation of the Code
1. **Data Loading**:
   - The CSV file is read, and the `PERCENT_PRI` and `DISTANCE_KM` columns are normalized to ensure they are on the same scale.

2. **Cost Matrix Construction**:
   - A cost matrix is created where each row corresponds to an employee and each column corresponds to a position. The values in the matrix represent the cost of assigning an employee to a position.

3. **Optimization**:
   - The `linear_sum_assignment` function from `scipy.optimize` is used to solve the assignment problem. This function implements the Hungarian Algorithm, which is efficient for such problems.

4. **Output**:
   - The optimal assignments and the total minimized cost are printed. The results are also saved to a CSV file for further analysis.

### Step 5: Results
- The output will include:
  - The optimal assignment of employees to positions.
  - The total minimized cost based on the chosen objective (e.g., combined `PERCENT_PRI` and `DISTANCE_KM`).

### Step 6: Next Steps
- If you want to prioritize one variable over the other, adjust the weights \(w_1 \) and \(w_2 \) accordingly.
- If the dataset is large, ensure the cost matrix is sparse to optimize memory usage.

Let me know if you'd like me to run this code or adjust it further!

```
Execution Error: ValueError                               Traceback (most recent call last)

Cell In[3], line 11
      7 data = pd.read_csv(file_name, sep=";", header=None, names=["IDENTIFICADOR", "CARGO",
"PERCENT_PRI", "DISTANCE_KM"])
      9 # Step 2: Create the cost matrix
```

```
     10 # Normalize PERCENT_PRI and DISTANCE_KM to bring them to the same scale
---> 11 data["PERCENT_PRI"] = data["PERCENT_PRI"].str.replace(",", ".").astype(float)
     12 data["DISTANCE_KM"] = data["DISTANCE_KM"].str.replace(",", ".").astype(float)
     14 # Normalize the columns

...

ValueError: could not convert string to float: 'PERCENT_PRI'
```