import pandas as pd

import numpy as np

from scipy.optimize import linprog

# Данные

data = {

'Store': ['Store 1', 'Store 2', 'Store 3', 'Store 4'],

'Input1': [300, 800, 700, 500],

'Input2': [15, 25, 20, 17],

'Input3': [2000, 3000, 1500, 2500],

'Output1': [130, 200, 300, 250],

'Output2': [2000, 2500, 1500, 3000],

'Output3': [200, 400, 350, 250],

'Output4': [16, 20, 35, 15]

}

df = pd.DataFrame(data)

inputs = df[['Input1', 'Input2', 'Input3']].values

outputs = df[['Output1', 'Output2', 'Output3', 'Output4']].values

stores = df['Store'].tolist()

efficiencies = []

for i in range(len(stores)):

# Переменные: веса входов a\_k и веса выходов b\_m

# Целевая функция: максимизировать эффективность -> max sum(b\_m \* y\_m^p)

c = -outputs[i] # коэффициенты b\_m для max, преобразуем в min

# Целевая функция: вектор коэффициентов для минимизации.

# c = [-y1, -y2, -y3, -y4, 0, 0, 0] — первые 4 переменные — это веса выходов b\_j, которые мы хотим МАКСИМИЗИРОВАТЬ (поэтому ставим минус).

# последние 3 переменные — это веса входов a\_k, они в целевой функции не участвуют (поэтому 0).

# Мы объединяем вектор коэффициентов для выходов и входов в один вектор переменных: [b1, b2, b3, b4, a1, a2, a3]

# Ограничения:

# Для всех магазинов j: sum(b\_m \* y\_m^j) - sum(a\_k \* x\_k^j) <= 0

A = []

b = []

for j in range(len(stores)):

constraint = np.concatenate([outputs[j], -inputs[j]])

A.append(constraint)

b.append(0)

A = np.array(A)

b = np.array(b)

# Ограничение нормализации: sum(a\_k \* x\_k^p) = 1

A\_eq = np.array([np.concatenate([np.zeros(outputs.shape[1]), inputs[i]])])

b\_eq = np.array([1])

# Границы: все веса >= 0

bounds = [(0, None)] \* (outputs.shape[1] + inputs.shape[1])

res = linprog(

c=np.concatenate([c, np.zeros(inputs.shape[1])]),

# Целевая функция: вектор коэффициентов для минимизации.

A\_ub=A, b\_ub=b,

# Ограничения-неравенства (<=):

# Для всех магазинов j: (b1\*y1^j + ... + b4\*y4^j) - (a1\*x1^j + a2\*x2^j + a3\*x3^j) <= 0 или A\_ub\*x<=b\_ub

# Это условие гарантирует, что ни один другой магазин не может показать эффективность больше 1 при тех же весах.

A\_eq=A\_eq, b\_eq=b\_eq,

# Ограничение-равенство (==):

# Для текущего магазина i: a1\*x1^i + a2\*x2^i + a3\*x3^i = 1 или A\_eq\*x=b\_eq

# Это нормализация "стоимости" входов — чтобы оценка эффективности была адекватной (входы оцениваются в 1 единицу).

bounds=bounds,

# Границы переменных:

# Все веса (и входов a\_k, и выходов b\_j) должны быть >= 0.

# Это логично: отрицательные веса экономически неинтерпретируемы.

method='highs'

# Метод решения — 'highs'

)

efficiency = -res.fun if res.success else None

efficiencies.append(efficiency)

result\_df = df[['Store']].copy()

result\_df['Efficiency Score'] = efficiencies

result\_df['Status'] = result\_df['Efficiency Score'].apply(lambda x: 'Efficient' if x is not None and round(x, 3) == 1 else 'Inefficient')

result\_df

Результат :

| **Store** | **Efficiency Score** | **Status** |
| --- | --- | --- |
| 0 | Store 1 | 1.000000 | Efficient |
| 1 | Store 2 | 0.974522 | Inefficient |
| 2 | Store 3 | 1.000000 | Efficient |
| 3 | Store 4 | 1.000000 | Efficient |