Hello! Today we’re going to learn about how to solve another linear programming problem by using LpMinimize. Let’s read the problem (read the problem).

So from this picture, we can see the 6 ingredients that make up a 100 gram can of cat food.

(read the next markdown “each ingredient…”) From this table we see that for example chicken, has the following nutritional values of protein, fat, fiber, and salt.

(scroll to code that’s already entered) Before the prob variable or type of problem are defined, the key problem data is entered into dictionaries. This includes the list of ingredients, followed by the cost of each ingredient, and its percentage of each of the four nutrients. The ingredients are the reference keys, with the numbers as the data. Along with the lists and dictionaries, I also already added the import statement to use pulp.

First we need to create the prob variable to contain the problem data (type):

(create new cell)

# Create the 'prob' variable to contain the problem data

prob = LpProblem("The\_Whiskas\_Problem", LpMinimize)

A variable called prob (although its name is not important) is created using the LpProblem function. It has two parameters, the first being the arbitrary name of this problem (as a string), and the second parameter being either LpMinimize or LpMaximize depending on the type of LP you are trying to solve.

Now let’s define the referenced variables in the problem (type):

ingredient\_vars = LpVariable.dicts("Ingr", Ingredients, 0)

The problem variables Ingr\_BEEF, Ingr\_CHICKEN etc. are created using the LpVariable class. It has four parameters, the first is the arbitrary name of what this variable represents which in our case is “Ingr”, the second is the lower bound on this variable, the third is the upper bound, and the fourth is essentially the type of data (discrete or continuous). We didn’t specify anything as the third parameter so there is no upper bound. The options for the fourth parameter are LpContinuous or LpInteger, with the default as LpContinuous. A dictionary called ingredient\_vars is created which contains the LP variables, with their defined lower bound of zero. The reference keys to the dictionary are the ingredient names, and the data is Ingr\_IngredientName. Let’s print this just to show the dictionary (type):

Print(ingredient\_vars)

(run cell)

Here we see the dictionary

(delete print statement)

Now we have to add the objective function. The objective function is just the cost of each ingredient multiplied by the decision variable and for all these values to be summed. Let’s add that objective function and utilize a list comprehension to make that happen (type):

# The objective function is added to 'prob' first

prob += (

lpSum([costs[i] \* ingredient\_vars[i] for i in Ingredients]),

‘Total Cost of Ingredients per can’

)

The variable prob now begins collecting problem data with the += operator. The objective function is logically entered first, with an important comma at the end of the statement and a short string explaining what this objective function is. Since costs and ingredient\_vars are now dictionaries with the reference keys as the ingredient names, the data can be simply extracted with a list comprehension as shown. lpSum() is a function from the PuLP library that calculates the sum of a list of expressions. Let’s see this printed out (type):

Print(prob)

(run the cell)

Here we see the objective function equation and the cost of each is next to the correct decision variable

(delete the print statement)

The constraint is logically entered after this, with a comma at the end of the constraint equation and a brief description of the cause of that constraint. Further list comprehensions are used to define the other 5 constraints, which are also each given names describing them. Let’s take a look one more time at the problem (scroll up). The constraints are these nutritional requirements so we need at least 8g of protein, 6g of fat, no more than 2g of fiber and 0.4g of salt. There may look to only be 4 constraint equations, but all the ingredient percentages must equal 100 so that’s our fifth equation. Let’s write out the constraints (type):

# The five constraints are added to 'prob'

prob += lpSum([ingredient\_vars[i] for i in Ingredients]) == 100, "PercentagesSum"

prob += (

lpSum([proteinPercent[i] \* ingredient\_vars[i] for i in Ingredients]) >= 8.0,

"ProteinRequirement"

)

prob += (

lpSum([fatPercent[i] \* ingredient\_vars[i] for i in Ingredients]) >= 6.0,

"FatRequirement"

)

prob += (

lpSum([fiberPercent[i] \* ingredient\_vars[i] for i in Ingredients]) <= 2.0,

"FiberRequirement"

)

prob += (

lpSum([saltPercent[i] \* ingredient\_vars[i] for i in Ingredients]) <= 0.4,

"SaltRequirement"

)

Print(prob)

(run cell)

Now when we print prob, we see the words “SUBJECT TO” and that shows our constraint equations.

(delete print statement)

We now have enough to solve the problem so let’s go ahead and write (type):

prob.solve()

(run cell)

Here we see an optimal value for the objective value which is 0.89 or 89 cents which is the minimum value needed to meet the nutritional requirements. The results of the solver call can be displayed as output to us using the LpStatus function which can be one of “Not Solved”, “Infeasible”, “Unbounded”, “Undefined”, or “Optimal”. LpStatus is a dictionary, so its input must be in square brackets. Let’s print the status (type):

print("Status:", LpStatus[prob.status])

(run cell)

Here we see neatly displayed that the status is optimal

But what if we want to find the percentage of each ingredient in the can? We can use a for loop to cycle through all the problem variable names and extract the value using the varValue function (type):

for v in prob.variables():

print(v.name, "=", v.varValue)

Name and varValue are properties of the object variable.

(run cell)

Here we see the percentages of each ingredient and it looks like mutton has the highest percentage with 50 percent of the can being mutton.

Even though we already saw the total cost of ingredients, we can use the value function to extract the value for the total cost of ingredients per can (type):

print("Total Cost of Ingredients per can = ", value(prob.objective))

(run cell)

objective is an attribute of the object prob.

So that’s a wrap for this video, hope you learned something and thanks for watching.