Optimize your Investments using Math and Python

Using Linear Optimization in Python's PuLP

Andrew Hershy Follow
Aug 14, 2019 · 6 min read ★

Source

uring the MBA, we learned all about predictive modeling techniques using

D Excel and University of Waikato's free software, WEKA. We learned the foundational concepts but never ventured into the hard skills required for advanced calculation. After some time studying python, I thought it would be fun to rework one of my linear optimization projects I originally did in Excel's solver. The goal of this article is to recreate the project in python's PuLP, share what I learn along the way, and compare python's results to Excel's.

> *The real world benefits /applications of linear optimization are endless. I would highly recommend following along closely, at least on a conceptual level, and making an effort to learn this skill if you are not already familiar with it.*

. . .

Table of Contents

4. Linear Optimization using PuLP

. . .

What is Linear Optimization?

According to Wikipedia, linear programming is “a method to achieve the best outcome (such as maximum profit or lowest cost) in a mathematical model whose requirements are represented by linear relationships.” These lecture notes from Carnegie Mellon University were very helpful in my own understanding of the topic.

In my own words, I would describe it as being a way to solve minimum / maximum solutions for a particular variable (decision variable), intertwined with other linear variables, to an extent that it would be very difficult to solve the problem with a pen and paper.

. . .

The Assignment / Review:

This is a linear optimization problem with regard to risk and return of a portfolio. Our objective is to minimize portfolio risk while simultaneously satisfying 5 constraints:

1. The sum of the investments will be $100,000

2. The portfolio has an annual return of at least 7.5%

3. At least 50% of the investments are A-rated

4. At least 40% of the investments are immediately liquid

5. No more than $30,000 are in savings accounts and certificates of deposit

The detailed instructions are below:

Source: pg 127

To review the process in solving a linear optimization problem, there are 3 steps:

1. **Decision Variables:** Here, there are 8 decision variables. They are our investment options.
2. **Objective Function:** We want to **minimize** the risk for the 8 investments. Below are the investments multiplied by their respective risk coefficients.

3. **Constraints:** Lastly, we want to define exactly what our constraints are. These are algebraically expressed below in the same order as we listed the constraints previously:

> *In case you are confused about the "7,500" in constraint #2, that would be the 7.5% annual return we are looking for multiplied by our $100,000 investment.*

. . .

Data Upload and Clean:

Now that we have the problem set up, let's upload the data into pandas and import pulp:

```
from pulp import *
import pandas as pd
df = pd.read_excel(r"C:\Users\Andrew\Desktop\Fin_optimization.xlsx")
df
```

Output 1

Looking good. There are a few formatting changes that must be made in order to move forward, however.

1. Turn the “Liquidity” and “Ratings” columns into binary values. This is in regard to constraints #3 and #4. The relevant string values in these columns are “Immediate” for Liquidity and “A” for Rating. Distinguishing these string values from the others is necessary for further calculation.

2. Create a new binary column for Investment Type. Constraint #5 focuses on the savings and CD investment types, so distinguishing them from the other investment types will help later.
3. Create a column of all 1's for Amt_invested. This will be useful for constraint #1: the $100,000 total portfolio constraint.

```
#1a
df['Liquidity'] = (df['Liquidity']=='Immediate')
df['Liquidity'] = df['Liquidity'].astype(int)


#1b
df['Rating'] = (df['Rating']=='A')
df['Rating']= df['Rating'].astype(int)


#2
savecd = [1,1,0,0,0,0,0,0]
df['Saving&CD'] = savecd


#3
amt_invested = [1]*8
df['Amt_Invested'] = amt_invested
df
```

Perfect. Let’s move on.

. . .

Linear Optimization using PuLP:

The first step using PuLP is to define the problem. The code below simply defines our problem as minimization (with regard to risk)and gives it the title, “Portfolio_Opt”. We will add more to this ‘prob’ variable later.

```
prob = LpProblem("Portfolio_Opt",LpMinimize)
```

Next we will create a list of our decision variables (investments options). Then we will use that list to create dictionaries for each feature:

```
#Create a list of the investment items
inv_items = list(df['Potential Investment'])

#Create a dictionary of risks for all inv items
risks = dict(zip(inv_items,df['Risk']))

#Create a dictionary of returns for all inv items
returns = dict(zip(inv_items,df['Expected Return']))

#Create dictionary for ratings of inv items
ratings = dict(zip(inv_items,df['Rating']))

#Create a dictionary for liquidity for all inv items
liquidity = dict(zip(inv_items,df['Liquidity']))

#Create a dictionary for savecd for inve items
savecd = dict(zip(inv_items,df['Saving&CD']))

#Create a dictionary for amt as being all 1's
amt = dict(zip(inv_items,df['Amt_Invested']))
risks
```

Output of "risks" dictionary for reference

Next, we are defining our decision variables as investments and are adding a few parameters to it,

- **Name**: To label our decision variables
- **Lowbound = 0:** To make sure there is no negative money in our solution
- **Continuous:** Because we are dealing with cents to the dollar.

```
inv_vars = LpVariable.dicts("Potential
Investment",inv_items,lowBound=0,cat='Continuous')
```

Finally, we add the modified decision variable to our problem variable we made earlier and additionally enter the constraints. We are iterating over dictionaries using “for loops” for each investment item.

```
#Setting the Decision Variables
prob += lpSum([risks[i]*inv_vars[i] for i in inv_items])


#Constraint #1:
prob += lpSum([amt[f] * inv_vars[f] for f in inv_items]) == 100000,
"Investments"


Constraint #2
prob += lpSum([returns[f] * inv_vars[f] for f in inv_items]) >= 7500,
"Returns"


Constraint #3
prob += lpSum([ratings[f] * inv_vars[f] for f in inv_items]) >=
50000, "Ratings"


Constraint #4
prob += lpSum([liquidity[f] * inv_vars[f] for f in inv_items]) >=
40000, "Liquidity"
```

```
Constraint #5
prob += lpSum([savecd[f] * inv_vars[f] for f in inv_items]) <= 30000,
"Save and CD"
prob
```

Below is the problem:

Result:

```
prob.writeLP("Portfolio_Opt.lp")
print("The optimal portfolio consists of\n"+"-"*110)
for v in prob.variables():
    if v.varValue>0:
        print(v.name, "=", v.varValue)
```


This is exactly the same outcome Excel’s solver gave.

Refer to my Github to see the full notebook file.

. . .

Note from Towards Data Science's editors: *While we allow independent authors to publish articles in accordance with our rules and guidelines, we do not endorse each author's contribution. You should not rely on an author's works without seeking professional advice. See our Reader Terms for details.*

Please subscribe if you found this helpful. If you enjoy my content, below are some projects I've worked on:

Uber Reviews Text Analysis

Excel vs SQL: A Conceptual Comparison

Simple Linear vs Polynomial Regression

Is Random Forest better than Logistic Regression? (a comparison)

Gini Index vs Information Entropy

Predicting Cancer with Logistic Regression in Python

Bivariate Logistic Regression Example (python)

Calculating R-squared from scratch (using python)

Data Science | Portfolio | Optimization | Python | Guides And Tutorials

WRITTEN BY

Andrew Hershy

Follow

LinkedIn:https://www.linkedin.com/in/andrew-hershy-a7779199/

Towards Data Science

Follow

A Medium publication sharing concepts, ideas, and codes.

More From Medium

How I Got 4 Data Science Offers and Doubled my Income 2 Months after being Laid Off

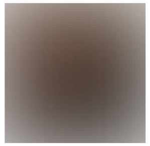
Emma Ding in Towards Data Science

How I'd Learn Data Science if I Could Start Over (2 years in)

Terence S in Towards Data Science

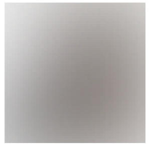
3 Programming Books Every Data Scientist Must Read

Dario Radečić in Towards Data Science



10 Cool Python Project Ideas for Python Developers

Claire D. Costa in Towards Data Science



4 Pandas Tricks that Most People Don't Know

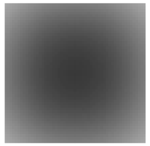
Christopher Tao in Towards Data Science

The Beauty of Bayesian Optimization, Explained in Simple Terms

Andre Ye in Towards Data Science

Long Short-Term Memory Networks Are Dying: What's Replacing It?

Andre Ye in Towards Data Science



Python Books You Must Read in 2020

Claire D. Costa in Towards Data Science