Investors Simulation

Financial Computing Exam Project

Caio Mescouto Terra de Souza

March 12, 2021

1 Introduction

The purpose of the project was to model a investor simulation based on three investor's mode (aggressive, mixed and defensive) and two types of investments (bonds and stocks). The portfolio of investment is built according with the investor's mode and a budget. (1)aggressive invests in stocks, (2)defensive in bonds and (3)mixed in both.

Within investments, (1)there are two types of bonds: Short and Long, both have minimum term, minimum value and yearly interest rate. Bonds are also compounded yearly and it is possible to calculate the interest rate for a given period. (2)There are seven types of stocks, all of them have the same behavior and attributes: term, price, amount and name. in addition, it is possible to get the price and return on investment for given start and end dates.

Investor has only two attributes, mode and budget, but the portfolio is the manner of bring investor and investments together. (1)The defensive portfolio has only bonds randomly selected. (2)The aggressive invest only on stocks that are randomly chosen. Finally, (3) the mixed portfolio is a weighted portfolio that has bonds and stocks.

2 Project Structure

2.1 The guideline structure

Any investment can be understood as a relationship between price (or present value) and future value. This relationship is a mathematical function that as follow (1):

$$f(PV, rate, period) \rightarrow FV \tag{1}$$

Sometimes the function is unknown, or the interest rate varies along the period. Therefore, the way to deal with this limitation is approach any investment as a cash flow. the first assumption is define the minimum period (by now 1 day). Therefore, the equation can be solved for each day, because for any date we now the "PV" (the "FV" of the day - 1), the period and the interest rate or the FV. Finally, period (or term) is basically the difference between dates.

The structure started with a object called "Investment" and "Bonds" and "Stocks" are objects whose inherit its attributes and methods. The role of "Investment" is abstract because we can't calculate the cash flow without a "mathematical" function that each investment type has its own. The "Investor" has only two attributes and the choice was to use a "namedtuple" to create it. and the last main structure is

the “Portfolio”. It is an object that combines Investor with Investments. The concept in which portfolio was thought is based on functions as First-Class objects in Python (Ramalho 2015, 167).

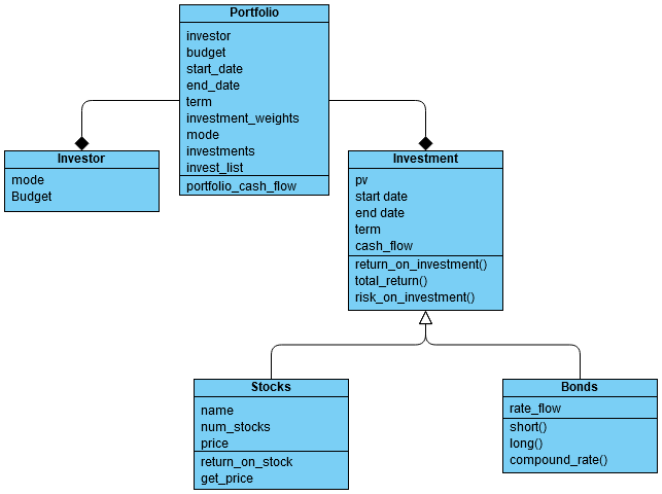

Figure 1: UML of main objects

2.2 Bonds

As the interest rate are annual and the unit of period was set as day, it was necessary define a strategy to deal with the transformation, since years do not have all the same numbers of days. As the trade-off between having exactly the daily rate or adopting a convention (Martellini, Priaulet, and Priaulet 2003, 4) is very small, compared to the work of coding it, the assumption adopted was day-count 365 (2).

$$i_{\text{daily interest rate}} = (1 + i_{\text{annual interest rate}})^{\frac{1}{365}} - 1 \tag{2}$$

Two different bonds were required, but as the behavior are the same, it was adopted a class method decorator to call each type of bond as a Bonds’s method. A generic bond can also be called and we applied this approach to answer the part one (see Figure 6 - Appendix).

Finally, Bonds has a method for calculating the compounded interest (compound_rate) for a given end date fulfilling the final requirement.

2.3 Stocks

Within the requirements it is important to highlight the capacity to deal with dates when not a business day. The approach was to use BDay pandas’ method to always bring one business day before the given date if it is not Business day. The consequence of this method is that the cash flow attribute has values to any day, or the real historical price or the last available price. This approach also achieve the same behavior of the Bonds’ cash flow attribute that is necessary to integrate investments into a portfolio. The Figure 7 (Appendix) presents price evolution for all stocks into the required period.

2.4 Investor and Portfolio

The investor was designed as a “namedtuple” with the required attributes and was designed a Portfolio object that brings one investor with investments (bonds and/or stocks). Portfolio is a object that has only one method (portfolio_cash_flow) and it call the mode function for building itself. The object also has a attribute to control the mixed distribution mode between Stocks and Bonds.

2.4.1 Defensive mode

It is defensive portfolio building function. It receives a portfolio and calls other function (accounting_investment) that handles bonds or stocks to mount the investment set. It is a recursive function that call itself to build temporary portfolios at the due of each bond. Finally it returns a dictionary with all bonds including that ones of the temporary portfolio.

2.4.2 Aggressive mode

As stocks do not have due dates, It receives a portfolio and call the same function cited above to mount the investment set. It returns a dictionary with all stocks.

2.4.3 Mixed mode

This function first randomly weighted chose between bond or stock, call the function that pick the investment metrics (pick_bond or pick_stock) and build a dictionary with all names(keys) and values of the investments. While the budget is bigger than the target required (short term bond) it runs. It is also a recursive function that works with a similar behavior presented for Defensive mode. In addition it handles one exception: TypeError. when pick_stock returns nothing.

2.4.4 Portfolio requirements

To deal with the requirements to build the three portfolios type more three functions were developed (accounting_investment, pick_bond and pick_stock). (1) Pick_bond deal with the random chose between short and long term and return a dictionary with one type of bond and the value. (2) pick_stock randomly chose one stock, if the price is bigger than the budget it returns nothing, else it randomly chose between one and the maximum number of stock that the budget can buy. It also handle a exception that is a KeyError, as Stocks are called trough API and the period between start and end date are handle into the function and also depends on the Bonds due dates, sometimes it is not possible to bring a new stock. On this situation the function just pass. (3) Finally accounting_investment works mounting Defensive or Aggressive set of investments to be bought by them own functions.

2.5 Other assumptions and issues

The main problem that I had developing the project was the definition of how to deal with dates. As the "term" is an required attribute, the first approach was call bonds and stocks by start date and the term, but TimeDelta class is an absolute time definition and for dealing with period, only days are allowed. As I would like to work with years, I tried to use DateOffset that is a relative class from Pandas, that one can set years and according with the start or end date and it calculates the relative value. However dealing with the DateOffset is not straightforward and required a lot of type manipulation that was polluting the code. It also required a lot of exceptions. In the part 4 I gave up the DateOffset and refactored the code with start and end dates. In addition, during the refactoring process I also set short and long bonds with defined terms, because it's a more realistic, so I also solved all simulations with this feature.

I chose to use Descriptor classes to ensure the behavior of all objects, so only specific entries are allowed. It is not necessary, but without it the objects might mistated the entries. As the idea was to build a simulatior that any one can try different combinations - regarding the scope - it is important to ensure that the entries are well defined.

Finally, I might worked with numpy instead of pandas because it's more efficient, but it probably would require more coding work and is definitely more verbose, so pandas turns the process less painfull. I also tried to use list and dict compreehension as much as I could, because it is more efficient and readable than loops.

3 Simulations

3.1 Part 3

First was built 500 portfolios of each mode and then was calculated the mean returns and volatility for each mode as follow (see Table 1). As expected, there is a trade-off between return and volatility.

Table 1: Mean returns and daily volatilities

	Defensive	Aggressive	Mixed
Total Return	0.1159	0.3081	0.3012
Daily Volatility	0.0002	0.0116	0.0107

Finally it is also presented the portfolios values through the period (see Figure 4).

Figure 2: Portfolio Value

3.2 Part 4

The first requirement asked on this part is the mean yearly return for every investor mode (see Figure 5). The second point was to change the investment weight into the mixed portfolio. As the increase in budget does not modify the general behavior, The result summarized into the Table 2 has calculated with the budget of 50,000. The same trade-off is observed but the return and volatility of the mixed portfolio are - in percentage - lower than the first simulation compared to the aggressive portfolio.

Table 2: Mean returns and daily volatilities

	Defensive	Aggressive	Mixed
Total Return	0.1277	0.3234	0.2829
Daily Volatility	0.0000	0.0116	0.0104

Figure 3: Portfolio Annual Returns

3.3 Bonus

The last simulations was with randomly distributed budget $N \sim (20,000; 5,000)$. Below is presented the value for the portfolios values through the period (see Figure 6). The behavior is similar through simulations, but we can observe a bigger gap between Aggressive and Mixed investors and a lower difference in volatility (see Table 3).

Table 3: Mean returns and daily volatilities

	Defensive	Aggressive	Mixed
Total Return	0.1268	0.3625	0.2964
Daily Volatility	0.0001	0.0116	0.0111

In addition, the best year for an aggressive investor was 2020 (see Figure 5) and was the same pattern through all simulations. Finally, the best stock to have in 2017 was FDX followed by GOOGL and MS (see Figure 7).

References

Hilpisch, Yves. 2015. *Python for Finance*. 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly Media, Inc.

Martellini, Lionel, Philippe Priaulet, and Stephane Priaulet. 2003. *Fixed-Income Securities : Valuation, Risk Management, and Portfolio Strategies*. The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England: John Wiley & Sons Ltd.

Michael Waskom. 2021. "Seaborn: Statistical Data Visualization." https://seaborn.pydata.org/index.html#.

Python Software Foundation. 2021. "Python 3.9.2 Documentation." https://docs.python.org.

Ramalho, Luciano. 2015. *Fluent Python*. 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly Media, Inc.

The Matplotlib development team. 2021. "Matplotlib: Visualization with Python." https://matplotlib.org/stable/index.html.

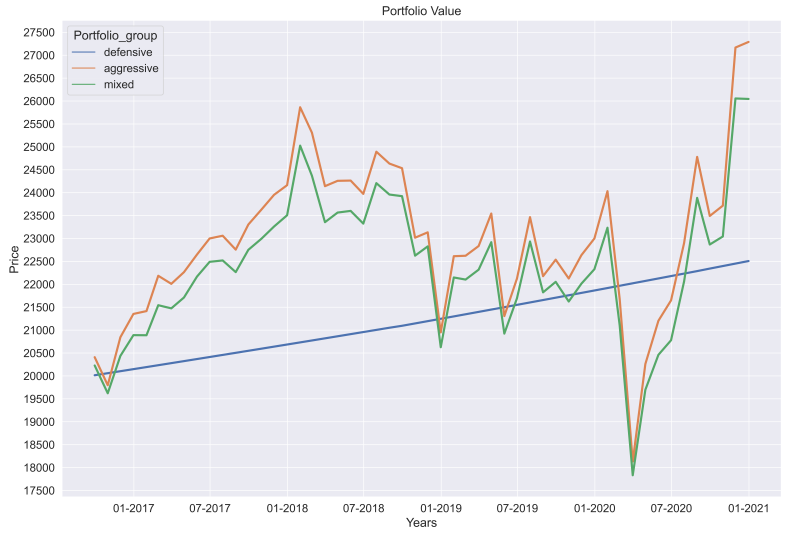

Figure 4: Portfolio Value

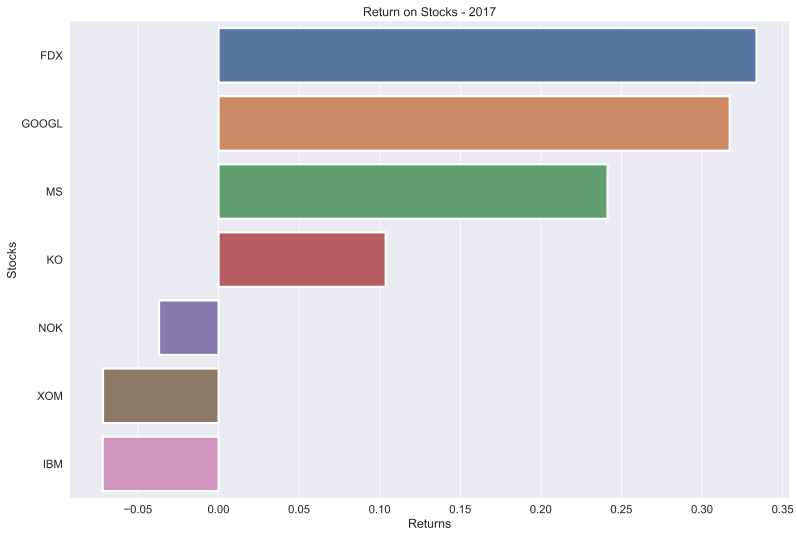

Figure 5: Return on Stocks

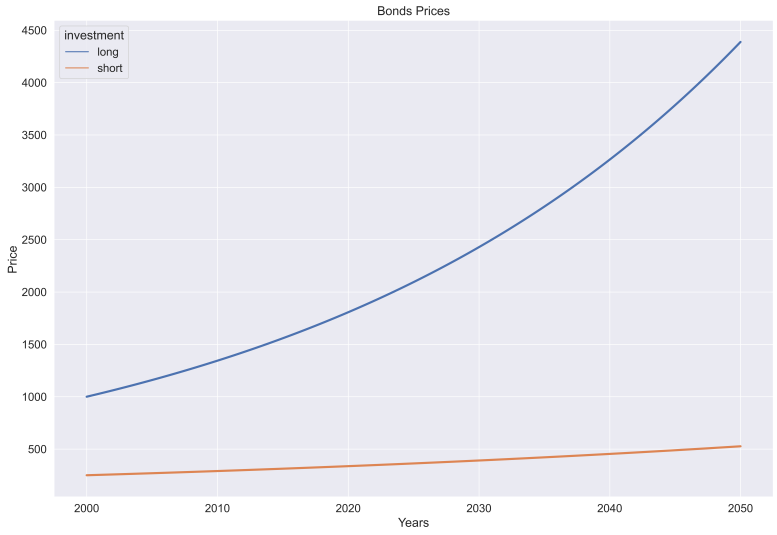

Figure 6: Bonds over 50 years

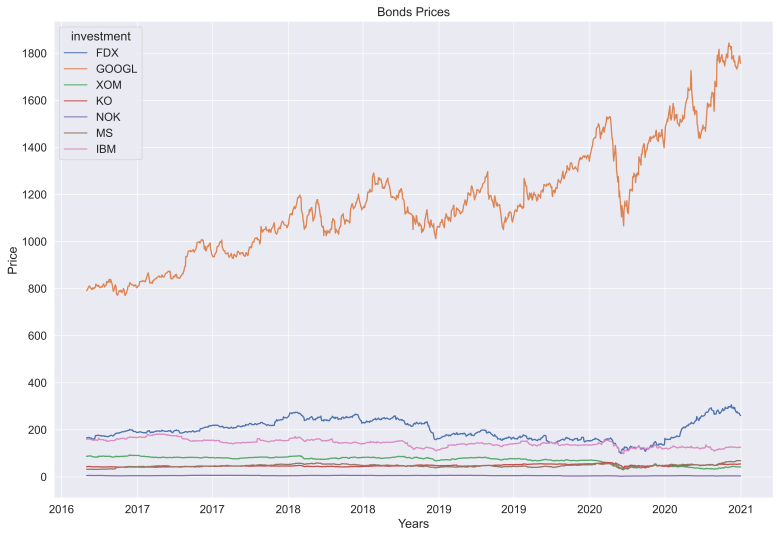

Figure 7: Stock Prices