
Developer Showcase: DCF Valuation with Python

By Shuuuux

FIN 3028 Python for Finance (2023 – 2024)





Content

| | |
|---|----------|
| Introduction | 2 |
| Instruction for Technical User | 2 |
| API Dependency..... | 2 |
| Pre-Runing Setup | 2 |
| Example..... | 3 |
| Key Points to Note..... | 4 |
| Additional Information for Non-Technical Users | 5 |
| Documentations..... | 5 |
| Four Steps to Generate Output..... | 5 |
| Plot Function Explanation List..... | 6 |
| Data source | 7 |
| Reflection | 8 |
| As a developer..... | 8 |
| As a Project Manager | 8 |

Introduction

A key component of a stock pitch involves the valuation of the stock. The most common approach is to download the three financial statements and then construct a model using Excel. While this method is visual, it's relatively slow and not very user-friendly for those who may forget certain formulas. To address this issue, I've considered utilizing Python to make this process automatic. The sole user input would be the stock code, allowing for automatic generation of results. My code is a DCF valuation modeling to estimate the price of the input stock, designing to project the cash flows for the next five years based on the actual business performance of the previous five years, and it can also generate charts to make the outcomes more visually comprehensible.

Next steps:

-  Develop isolated Discounted Cash Flow functions to enhance the readability
-  Construct a graph to compare actual versus estimated stock prices over different time intervals
-  Devise a script to find out an appropriate assumed growth rate for a company
-  Formalize sensitivity analysis

Instruction for Technical User

API Dependency

Alpha Vantage provides free APIs for accessing historical and real-time financial data, technical indicators, and analytical tools, offering similar functionality to yfinance. This project necessitates an API key from Alpha Vantage, which you can obtain at no cost [here](#). Within the sample code, the key is setting as 'demo', you need to change it before you run the code.

Pre-Runing Setup

- Before running the program, please verify that all necessary files are located within the same working directory.

- In Main.py, there are four cells that need to be executed sequentially. Output cell is at the end.

Example

To review historical DCF analyses for IBM, enter the stock code—such as IBM or ibm—when prompted with 'Please enter the stock code' in the console (you can try: GOOGL, SBUX, etc.).

```
Enter the stock code: IBM
```

This retrieves IBM's financial data for the past five years, up to the most recent fiscal year-end, to facilitate DCF modeling. It assumes a Compound Annual Growth Rate (CAGR) for the total revenue growth rate. Terminal outputs details:

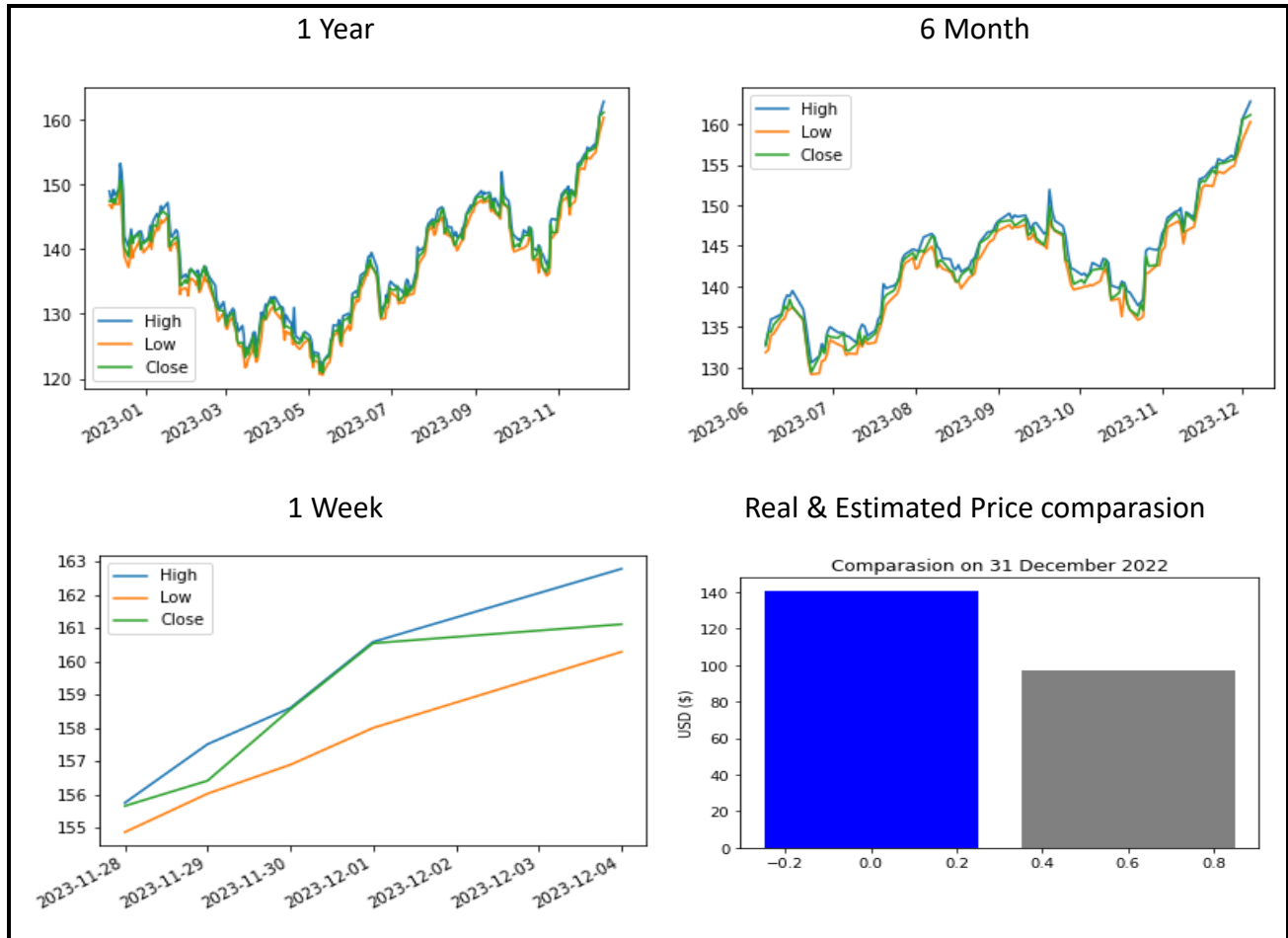
```
# Free cash flow model just as modeling in excel
      EFY6  EFY7  EFY8  EFY9  EFY10
TotalRevenue    56525  52786  49294  46033  42988
GrossProfit      30058  28070  26213  24479  22859
OperatingExpense 25606  23912  22330  20853  19473
EBIT             4452   4157   3882   3625   3385
D&A             2006   1873   1749   1633   1525
EBITDA          6458   6030   5632   5259   4911
UnleveragedFCF   8921   8331   7780   7265   6785

# Key parameters in the model
                        value
Perpetual growth rate    3.00%
WACC                     5.61%
Terminal Value           155026$m
implied Enterprise Value  188553$m
Implied Equity Value      88680$m
estimated stock price     97.118041
```

The code also offers real-time price data for three stocks over periods of one year, six months, and one week from today.

Additionally, the code provides a comparison chart at specific time points, showing actual values versus estimated values, where the `estimated` price is indicated in grey, while `real` is blue. It is

important to notice that because stock price fluctuations are influenced by factors beyond financials, the comparison of actual and estimated prices is not a judgment of the model's quality, but merely for contrast.



Key Points to Note

- **Model Application:** The model is designed specifically for U.S. stocks. Since perpetual growth rates differ for stocks in different countries, using this model for non-U.S. stocks may not yield accurate price estimates
- **Data Requirement:** Companies without publicly available financial data for at least five years cannot be evaluated using this model. An error will be shown in the initial lines of the output
- **Growth Rate Assumption:** The model uses the CAGR for total revenue, which may not be suitable for high-growth companies

Additional Information for Non-Technical Users

Documentations

- `main.py`: This is the main functional area, encompassing calculations and outputs
- `plot.py`: This file contains functions for generating plots
- `listing_status.csv`: This allows users to look up stock symbols

Four Steps to Generate Output



The file `main.py` is for execution, make sure you set the correct working directory. There are 4 cells need to be run separately. Cells 1 can be helpful for checking the working directory. Cell 2 is the user input cell, please enter the stock code in the console. The process from cell 3 & cell 4 involves calculations, estimated value output, unleveraged free cash flow model output, and visualization.

For Cell 3 & 4

Step 1: Data Retrieval

The code here is to get data from Alpha Vantage, including a company's three financial statements (income statement, balance sheet, cash flow), basic company information, and ten-year U.S. Treasury bond rates, and store them in local with JSON format.

Step 2: DCF Modeling

- a) Preparation: Extract key variables which required to calculate estimated future 5-year unleveraged free cash flow from financial statements and store each variable in a numpy list (e.g., past 5-year total revenue, gross profit, etc.).
- b) Build a free cash flow model to estimate 5-year unleveraged free cash flow. Here I assume the growth rate for total revenue is the past 5-year CAGR of total revenue.

Step 3: WACC Calculation

- Extract data from the income statement to calculate the effective tax rate.
- Use CAPM to calculate the cost of equity. Here, the market return is the 5-year average S&P 500 return (<https://tradethatswing.com/average-historical-stock-market-returns-for-sp-500-5-year-up-to-150-year-averages/>), and the risk-free rate is the Ten-year Treasury Yield at the nearest FY end date.
- Calculate the cost of debt based on data extracted from financial statements. Interest expense is from the income statement, while the total debt is from the annual report.
- Calculate the proportion of equity (market cap) to debt, using data from company fundamentals.
- Calculate WACC based on the above variables.

Step 4: Stock Price Estimation

A pure calculation process utilizes the estimated future 5-year free cash flow and WAC to estimate the terminal value using the DCF model to determine the stock price. Notice that you will also get three charts which show the historical time series daily of the stock price.

Lastly, Cell 4 Output:

You will get unleveraged free cash flow calculation model, key variables, and a real & estimated stock price comparison chart.

Plot Function Explanation List

| Function | Usage |
|--|---|
| <code>visual_timeseries(symbol)</code> | generating three stock price timeseries charts with time period 1 year, 6 month, 1 week, showing high, low, close price |
| <code>visual_compare_histoyical(symbol, estimated dprice)</code> | generating a bar chart to compare real price and estimated price at latest fiscal year end |

Data source

Alpha Vantage: <https://www.alphavantage.co/documentation/>

Perpetuity growth rate: [get from https://ycharts.com/indicators/us_real_gdp_growth](https://ycharts.com/indicators/us_real_gdp_growth) - basic info

market return: <https://tradethatswing.com/average-historical-stock-market-returns-for-sp-500-5-year-up-to-150-year-averages/>

Reflection

As a developer

I was inspired to use Python for building a DCF model, thinking it would streamline my next stock pitch interview, sparing me the cumbersome task of model building in Excel. While Excel is easy but time-consuming, Python proved complex but rewarding. I was surprised to find myself applying all the knowledge I've learned this semester in the code encompassing everything from basic lambdas to intricate loops, tidy data, and web scraping, all featured in my showcase code.

At the first stage of my coding, I find most of the data which Alpha Vantage provided was in JSON format, so I quickly learned how to extract data from JSON online. Writing the code involved self-teaching some functions to achieve specific functionalities, such as using `'df.index.get_indexer(_, method = 'nearest')'` to address missing values. After completing the code, I planned to organize it for readability. I reviewed exemplary codes on GitHub to learn about writing effective comments and discovered that I could separate visualization into a separate Python file, reducing redundancy in the main program.

Although my coding process seemed smooth, reviewing outstanding codes on GitHub made me realize there's much room for improvement. I should have studied others' exemplary work during the preparation phase. Reading other's code can be a learning process, not a plagiarism. A good developer not only needs solid basics and clear logic but also must learn to stand on the shoulders of giants. For future improvements, I could move data acquisition and DCF functions to separate supporting files, enhancing my program's simplicity and maintainability, and enabling more complex functionalities. Additionally, planning the comments while writing the code would be a significant time-saver.

As a Project Manager

I'm pleased with my assignment, as I not only met but also exceeded my initial goals by creating an

FCF model, which was better than expected. It's gratifying to solve problems using my own code, and I believe it will speed up valuation preparation for my next equity research interview. The assignment was slightly delayed due to preparations for a master's program and a job interview, but I still managed to complete it on time.

However, there is still a lot with this project that I can improve. Firstly, I should set more preparation time. I need to prepare more thoroughly by utilizing online resources earlier, such as checking GitHub for professional codes and learning efficient modeling and annotation techniques. For my next project, I'll spend more time on researching and learning from others' codes after forming a basic idea, aiming for more efficient and refined coding.

Secondly, I should be alert when I spend much more time than I expected and improve my time management skill. I shouldn't spend hours on trivial bugs, like the issue where a chart wouldn't generate when running the entire file. Despite spending 3-4 hours, I couldn't fix it and eventually moved on. Thanks to the professor's guidance, I finally realized my issue. Lastly, unexpected interviews disrupted my schedule, but starting early helped me finish. Next time, I'll allocate more time for a project consciously to get prepared for some unexpected circumstances.