Metrics, Ratios, Analysis, R2 Scores and Others

[A Finance Project; An experiment of skillset and in know-how]

Abstract
[What if I analyzed all the stocks on the s&p500 index in various forms to see statement ratios for investors looking for long-shot investment and stock/dividend analysis for immediate-returns investments]

MHT
Huzaifatariq.4815@gmail.com

Table of Contents:

Metrics, Ratios, Analysis, R2 Scores and others

(First of all, my deepest apologies for the naming conventions and the folder structure and hierarchy mess)

A. Outline:

This project automatically takes all the stocks on the s&p500 index as of August, 2024 and does various sorts of analysis on them. These are given as output in the form of csv's and can also be accessed through flask. The details on all the files are listed below.

[MAIN.py will do it all. It took 26 mins on my CPU. Did not use GPU. Half the time is fetching the API data from yfinance and the other half is RFR Linear Regression]

B. Dependencies Used:

- Pandas
- Flask
- yFinance
- FredAPI
- NumPy
- Glob
- Scikit-Learn

C. Main Methodology & Process:

1. First a list of 500 tickers is given that corresponds with the S&P500 to the yFinance library. The data that is used dates from Jan 2013 to Jan 2023 (Daily Data) – ['Opening', 'Closing', 'Volume', 'Dividends' (quarterly or yearly), 'Stock Splits']. I kept the stocks historical data limited as the data for the other items that were also downloaded was limited to 5 years (when doing regression on equivalent timeframe data was used for macroeconomic indicators). These files included the balance sheet, financials and the cashflow of the company (Yearly Data). *This was all the main and primary data that all he further analyses were committed on (aside from macro data as secondary data).* It is saved separately in the form of .csv files

2. Secondly, the Macroeconomic Data is downloaded using the FredAPI. (You must use your own key in the code – its costs nothing). The indicators I choose were the very basic ones; GDP, Unemployment, Inflation and CCI. The data was for 10 years – same base as stocks, with the key difference of the data being given on quarterly rather than daily basis

3. After that the analysis is done on stocks for the following metrics:
    - Avg stock over the past ten years
    - If the stock trend is increased in more number of years or decreased in more. (not overall but a year count)
    - Average returns for the stocks i.e. it's *mean*
    - Standard Deviation – the deviation on average of a stock from the mean
    - Covariance (Stock, Market). A note here that the market data was taken from 'IRX' ticker which is for the us treasury bills. It was taken only for one year to represent the latest market conditions. (The entire US Treasury bill file is also saved to a csv)
    - Variance of each stock is given
    - Based on the Cov / Var, the beta is calculated the represents the return of that particular stock's performance in contrast to the market.
    - Finally, the R2_Score is given based on the 10-year average of each stock. The Independent variables are the Macroeconomic Indicators; [GDP, Unemployment, inflation, CCI]. The score given doesn't matter much as all these obviously had major influences in the past 5 years. Some other combination of metric must be chosen in the future for a more usable measurement.

4. The next file analyzed is the dividends. Items that an investor who prefers dividends over capital gains could look out for. The metrics for the dividends are as follows:
    - Dividends count per year
    - Latest dividend paid ($) per share
    - Years that dividends were not paid. Non-cumulative. (Remember the data is for 10 years)
    - If more years had dividends increasing or decreasing over time. (read this carefully please)
    - All years average payment of dividends
    - Dividend Yield i.e. dividend per year/ stock price (average not current)
    - The price to earnings ratio that shows the relation between market price of share and the earnings per share of a stock
    - Free Cash (in millions $) – to show how easy it is for that company to cover the dividend payments.

5. Then the ratios were computed based on balance sheets, financials and cashflow statements for the given period of five but sometimes 4 years (limited availability). These included:

    - Current Assets
    - Quick Ratio
    - Debt To Equity Ratio
    - Total Debt to Total Capitalization Ratio
    - Debt To Assets Ratio
    - Net Debt to Equity Ratio
    - Return On Equity
    - Return On Assets
    - Return On Investment Gross Profit Margin
    - Asset Turnover
    - Inventory Turnover
    - Receivables Turnover

    Each and every stock received a separate file for these computations to better grasp them.

6. Then the averages were taken for each an every stock on the 500 index for each category as listed above. This gives a general idea of the past 5 years performance of these ratios. The high performers and the low performers can be sorted out. The data is finally saved to a single file for each ticker for ease of comparison.

7. A risk analysis on all the ratios calculations as listed in step 5 is done in an aggregate file. The risk judges the Macroeconomic Indicators influence on each of the ratios separately. The given result is the R2_Score which shows how much impact those independent variables have on each and every ratio taken as the dependent variable. The R2 scores are then averaged out for all ratios and saved to a separate file. The M.E indicators listed at the end of both files also have scores attached to them. These scores are the influence that each indicator (Indep Var.) has had on the ratios as a whole, which is why they are the same in the both files and will sum up to a total of 1.

8. A separate file is included both as a .py file and as a flask file on which you can look up various tickers within the index for comparison and they return all the above listed metrics in a new csv file or a flask table on your browser.

9. The Flask app was mostly made of borrowed code and due to its current simplicity, its functions are limited to looking up tickers – separated by commas – and presenting the

top files based on my averages. There are two averages upon which the top 33 – randomly chosen number – companies are chosen but more on this later.

D. All Files Included:

1. Main Files to Run:

[The main files that you just need to run to make everything work especially if you want to update all ticker (stock symbol) data]

I. MAIN.py – Just execute it. It does everything from start to end.
II. TICKER_LOOKER.py – Separate file to look up specific tickers
III. SIMPLE_FLASK_API.py – Same as above but in API. The browser address will be given in the file when executed
IV. Fred_api_key_file.txt – not that important if csv already available but for latest data get the free api and put the number, as is, in this file before running main.py.

2. Accessory Python Scripts:

[They all run in Main.py]

I. analyze_the_stocks.py
II. companydata.py – *this is where the 500 S&P tickers are stored taken from Wikipedia. Change them as you see fit. Then just run the MAIN.py file again. That's it.*
III. dividends_analyzed.py
IV. figuringouttheratios.py – *adjust the ratios or add new ones to any of these files; up and below*
V. macroeconomic_data.py
VI. market_return.py
VII. merging_the_data.py
VIII. r2_score_calculator.py
IX. ratioAverages.py
X. risk_analysis_on_ratios.py
XI. standardizing_merged_files.py
XII. the_top_ones.py

3. Folders and Others:

[All respective calculated ratios are within these folders. Please navigate them and apologies for the naming convention]

I. All_Ratios – {ratios, ratioAverages, Risk_analysison_Ratios}
II. All_Stock {dividends_etc, stock_metrics}
III. macroindicators
IV. maindata

V. Result_for_tickers.csv – For when you used TICKER_LOOKER.py
VI. Two Top Performing ratios CSV's (Normal and Spicy)

E. R2_Score Calculation & The Tool Used:

To calculate the R2 scores in the file that concern it, scikit-learn's LinearRegression is the usual approach. Although the approach that I used was Random Forest Regressor with n_estimators set to 1000; meaning a thousand different trees. I choose this approach because it gave more diverse answers than the linear regression which I think was not accounting for the overfitting in the model and could not handle the limited availability of the data. RFR also handles non-linearity better and gives better accuracy for the model along with the features performance due to the countless decision trees it can run.

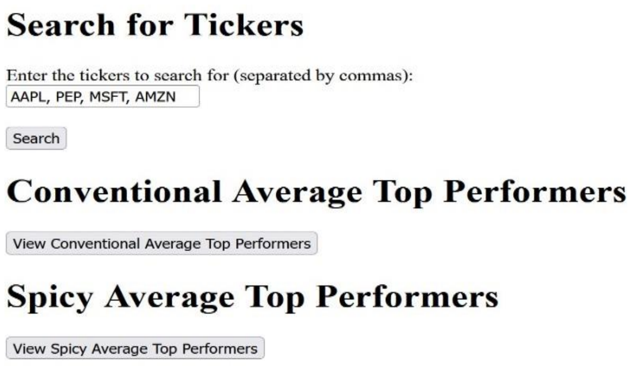
Even though RFR presented better and more in tune results, it still had a lot of the r2 scores very close to each other because of the simple and obvious reason that the data was for the past 5 years and due to the impact of covid in that time the macroeconomic indicators as suggested had a great impact on almost every stock. This, for now, only means that they mostly moved in the same direction i.e. it does not imply causation.

The interesting discovery was that many stock greatly affected by it both over and underperformed and the same for vice-versa scenario. Meaning that though it had a correlation with the stocks, the M.E indicators did not actually have great influence in the performance of the stocks. To repeat; this only applies to my understanding of the s&p500 stocks index. For others the variability explained may actually have had an influence which is more like than not.

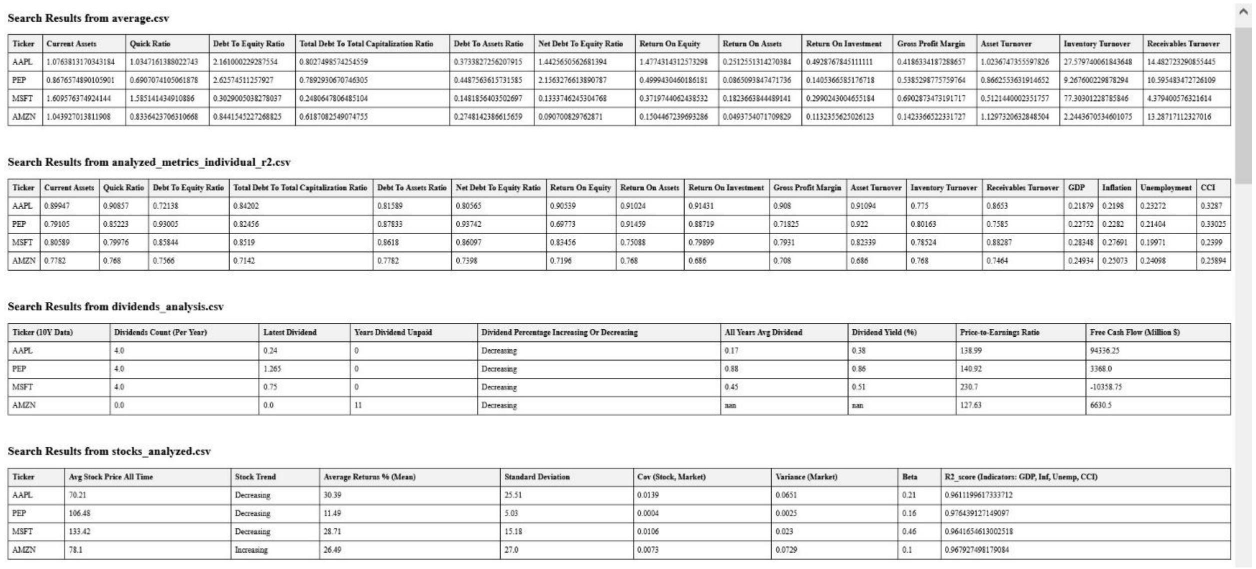
(p.s data was obviously standardized)

F. Flask API & Some Visuals:

1. Run the SIMPLE_FLASK_API.py
2. Open the browser address suggested in the code (or alternatively; localhost:5000)
3. It has three options:
    a. Compare ticker for many files using the tickers. It will return back 4 main metrics
    b. It will return a result of the top 33 performers by conventional ratio average methods
    c. It will also return a result of top 33 companies via a spicy 'trendy' method I used; more on it just after this.


Search for Tickers
Enter the tickers to search for (separated by commas):
AAPL, PEP, MSFT, AMZN
Search
Conventional Average Top Performers
View Conventional Average Top Performers
Spicy Average Top Performers
View Spicy Average Top Performers

4. The options are simple just for convenience, I know the Flask on a very elemental level right now and, as previously stated, had to borrow some of the code and stich it together.
5. I plan on including to scale it in the future in 2 months' time perhaps. (busy schedule).
6. The new features will include the same comparisons but being able to be searched by company name too instead of just the ticker and also to add comparisons of the top performers in a category in a visualized form. (Too many metrics right now to apply visualizations on them all.)



Search Results from average.csv

Ticker	Current Assets	Quick Ratio	Debt To Equity Ratio	Total Debt To Total Capitalization Ratio	Debt To Assets Ratio	Net Debt To Equity Ratio	Return On Equity	Return On Assets	Return On Investment	Gross Profit Margin	Asset Turnover	Inventory Turnover	Receivables Turnover
AAPL	1.0763813170343184	1.0347161388022743	2.161000229287554	0.8027498574254559	0.3733827256207915	1.4425650562681394	1.4774314312573298	0.2512551314270384	0.4928767845111111	0.4186334187288657	1.0236747355597826	27.579740061843648	14.482723290855445
PEP	0.8676574890105901	0.6907074105061878	2.62574511257927	0.7892930670746305	0.4487563615731585	2.1563276613890787	0.4999430460186181	0.0865093847471736	0.1405366585176718	0.5385298775759764	0.8662553631914652	9.267600229878294	10.595483472726109
MSFT	1.609576374924144	1.585141434910886	0.3029005038278037	0.2480647806485104	0.1481856403502697	0.1333746245304768	0.3719744062438532	0.1823663844489141	0.2990243004655184	0.6902873473191717	0.5121440002351757	77.30301228785846	4.379400576321614
AMZN	1.043927013811908	0.8336423706310668	0.8441545227268825	0.6187082549074755	0.2748142386615659	0.090700829762871	0.1504467239693286	0.0493754071709829	0.1132355625026123	0.1423366522331727	1.1297320632848504	2.2443670534601075	13.28717112327016

Search Results from analyzed_metrics_individual_r2.csv

Ticker	Current Assets	Quick Ratio	Debt To Equity Ratio	Total Debt To Total Capitalization Ratio	Debt To Assets Ratio	Net Debt To Equity Ratio	Return On Equity	Return On Assets	Return On Investment	Gross Profit Margin	Asset Turnover	Inventory Turnover	Receivables Turnover	GDP	Inflation	Unemployment	CCI
AAPL	0.89947	0.90857	0.72138	0.84202	0.81589	0.80565	0.90539	0.91024	0.91431	0.908	0.91094	0.775	0.8653	0.21879	0.2198	0.23272	0.3287
PEP	0.79105	0.85223	0.93005	0.82456	0.87833	0.93742	0.69773	0.91459	0.88719	0.71825	0.922	0.80163	0.7585	0.22752	0.2282	0.21404	0.33025
MSFT	0.80589	0.79976	0.85844	0.8519	0.8618	0.86097	0.83456	0.75088	0.79899	0.7931	0.82339	0.78524	0.88287	0.28348	0.27691	0.19971	0.2399
AMZN	0.7782	0.768	0.7566	0.7142	0.7782	0.7398	0.7196	0.768	0.686	0.708	0.686	0.768	0.7464	0.24934	0.25073	0.24098	0.25894

Search Results from dividends_analysis.csv

Ticker (10Y Data)	Dividends Count (Per Year)	Latest Dividend	Years Dividend Unpaid	Dividend Percentage Increasing Or Decreasing	All Years Avg Dividend	Dividend Yield (%)	Price-to-Earnings Ratio	Free Cash Flow (Million $)
AAPL	4.0	0.24	0	Decreasing	0.17	0.38	138.99	94336.25
PEP	4.0	1.265	0	Decreasing	0.88	0.86	140.92	3368.0
MSFT	4.0	0.75	0	Decreasing	0.45	0.51	230.7	-10358.75
AMZN	0.0	0.0	11	Decreasing	nan	nan	127.63	6630.5

Search Results from stocks_analyzed.csv

Ticker	Avg Stock Price All Time	Stock Trend	Average Returns % (Mean)	Standard Deviation	Cov (Stock, Market)	Variance (Market)	Beta	R2_score (Indicators: GDP, Inf, Unemp, CCI)
AAPL	70.21	Decreasing	30.39	25.51	0.0139	0.0651	0.21	0.9611199617333712
PEP	106.48	Decreasing	11.49	5.03	0.0004	0.0025	0.16	0.976439127149097
MSFT	133.42	Decreasing	28.71	15.18	0.0106	0.023	0.46	0.9641654613002518
AMZN	78.1	Increasing	26.49	27.0	0.0073	0.0729	0.1	0.967927498179084

G. Spicy Method:

This is a series of methods of ratios analyses and stocks/dividends analyses done by me using mostly the conventional metrics but some are new and some done differently. The files are too unprepared and there are many things to think through that is why I did not add it here yet. Not much will be said about them more. Although in the main directory I did add the result from my calculations based on the ‘Spicy Average’ of 33 companies to be compared with the ones produced from the conventional methods.

The results when compared on the internet and through code were that, although both sets have 11 overlapping companies out of 33, my Spicy set has the same hand on average returns but a lower volatility (less fluctuations) compared to the Conventional set. Spicy set has a lower overall Beta too, meaning less correlation with the movements of the market in general. It performed better on other metrics too especially when the time frame was extended to 10 years and actually showed an increase in average returns (lower timeframe – 1 year- showed the other set with insignificantly higher returns). The Conventional set however showed a lower Sharpe-Ratio meaning a better tolerance for risk.

Now I have to be fair in saying that all these comparisons, whether higher or lower, were, even on different timescales, just minor percentage worth of difference; in some cases the same even. My Spicy set has nothing big in it yet but I plan to improve in the direction I am going and hopefully by the next time around have an actual non-decimal measurable percentage increase in average returns relative to the market and the Conventional set to show. This however will too take some months of my time; that is if it pans out further: but enough on this.

H. How Error Handling Was Dealt With:

Errors for any files in which many rows contain ‘Null’, nan or 0 values are removed from the set for that specific analysis. The handling of this is different on a file-by-file basis. If too few data is present, then the ticker maybe removed or the file processed on the limited data which would provide skewed results (Once again it depends how much is missing and not too much or many values). *The errors out of the 500 that had to be removed were about 32.* Further error handling exceptions or methods could be added to compensate for these files too but because of the sheer amounts of categories going one by one was too much to be done just to handle a few files. On a larger dataset this will matter more.

Another inference could be made that companies with more readily available data and in larger volumes (timeframes) were performing better.

I. Am Not A Clairvoyant But In The Future:

I plan on putting in as many companies as possible with proper data presentation in APIs like Flask and along with visualizations is the idea right now. This would also be better for a quick comprehension when comparing various companies with one other. Plus, the larger dataset will provide better insights for analysis. For now, I want to perfect it on a smaller scale so that it functions as perfect as can be on a larger scale when it is done.

Tools For Data:

1. **yFinance Library** for the stocks and statements related information [Historicals, Balance Sheets, Cashflows, Financials]
2. **FredAPI** for the macroeconomic data [GDP, Inflation, Unemployment, CCI] (https://fred.stlouisfed.org/)
3. **Various Out-Python Libraries** i.e. Glob, NumPy, Pandas, Sklearn, Flask
4. **Special Mention:** Sublime Text Editor