# CSC111 Project: Decoding the Secrets of Successful Stocks

Yehyun Lee, Aung Zwe Maw, Wonjae Lee

April 1, 2023

## 1 Problem Description and Research Question

### 1.1 Project Goal

As an investor, the question is, **which stocks are most profitable in the long-term, and what factors contribute to their success? Our team will use a tree data structure to identify the promising factors and the most likely companies to invest in and utilize the backtesting approach to determine the profitability of the investments.**

We aim to determine the most promising stocks for investment by analyzing the factors which has the greatest correlation to the increase in the stock price of a company. Based on the analysis, we will rank the factors and create a binary tree from the factors. Then, we will filter stocks accordingly and invest in what the binary tree suggests. Lastly, using backtesting data we will draw conclusions on its performance.

### 1.2 Motivation

One of the reasons for this project is the increased interest in stocks during and after the pandemic. Many people have turned to investing in stocks to make a passive income or grow their savings during a time of economic uncertainty. However, with so many companies to choose from, it can be challenging to know which ones are most likely to provide a good return on investment. By using past datasets, we can analyze historical data, current market trends, and other relevant factors to identify the companies that will perform well in the future.

This can help investors make more informed decisions about where to put their money and maximize their returns. Another motivation for developing a stock market investment program is the passion of one of our partners, Yehyun Lee for stock market trends. His passion positively affected us and we were also willing to explore the trends along with him.

### 1.3 Context & Required Knowledge

Some background knowledge is required to understand the problem our team will be investigating. Stock investment involves buying and selling shares of companies with the expectation of earning a profit. When considering which stocks to invest in, that stores typically consider a range of factors, such as the price-to-earnings (PER), price-to-book ratio (PBR), and return on equity (ROE). These are three important factors.

PER can be calculated by share price / earnings per share. PBR is a company's stock price / book value per share. The book value of a company is calculated as its assets minus its liabilities. In general, lower PER and PBR indicate how undervalued or cheap the price is; whereas higher indicates how overvalued or expensive the price is.[1]

ROE measures profitability by calculating the amount of net income generated for each dollar of shareholder equity. A high ROE indicates that a company is generating significant profits for its shareholders, while a low ROE suggests that the company is not using its resources efficiently.[2]

Please note that there will be more factors our team will use to identify hidden factors that influence stock performance.[3]

[1]Further information regarding PER and PBR can be found in Appendix 6.1
[2]Further information regarding ROE can be found in Appendix 6.2
[3]Further information regarding other possible factors can be found in Appendix 6.3

## 2 Computational Overview

**General Architecture, Function, Data Type and Algorithm**

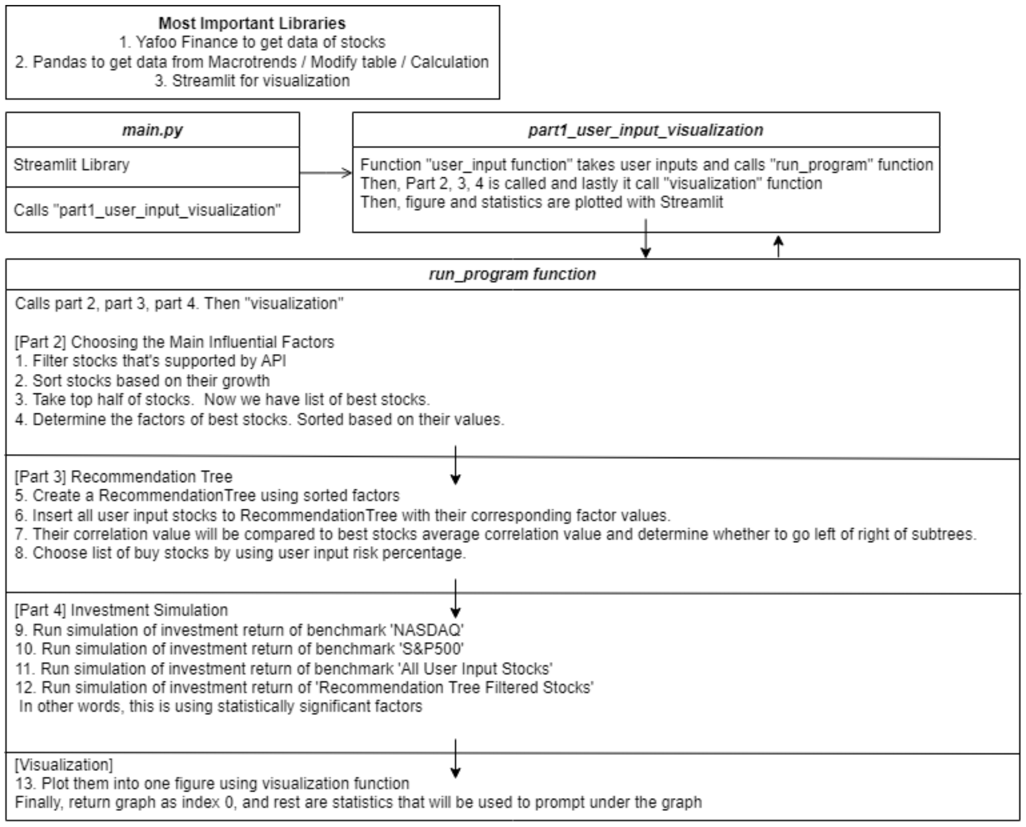
For simplicity, we have split our code into fives files: main.py, part1_user_input_visualization, part2_factor_processing, part3_recommendation_tree and part4_investment_simulation. Since, we want our program to run in a browser window using Streamlit, running our program calls the main.py file which in turn calls part1_user_input_visualization which contains the Streamlit modules. Then this file in turn calls the part2_factor_processing, part3_recommendation_tree and part4_investment_simulation.

The part2_factor_processing will take in all the top-performing stocks using the Yahoo Finance API [JEC23]. This part of the program determines which factors are the most important for determining what stocks to invest in (Data for factor value is taken by Macrotrends). It will return the factors in order of least important to most important by calculating average correlation values (Further information can be found in 2.1).

The part3_recommendation_tree contains a binary tree data structure and takes the input of the factors and their correlation values from part2_factor_processing and builds a binary tree in order of least important to most important features. This module contains a custom binary tree data structure with additional methods such as add_subtree, insert_stock and many more (Further information can be found 2.2). The generated tree is inserted with a list of all user input stocks. This tree helps classify which stocks to invest in according to risk percentage.

The part4_investment_simulation invests these stocks from the start year specified by the user until the most recent year e.g. 2023. Then, a graph of return on investment is given as output to part1_user_input using the plotly function (Further information can be found 2.3)

The part1_user_input takes the plotly output given by part4_investment_simulation and plots these in a browser using Streamlit. It also allows for users to input directly in the browser instead of using the terminal (Further information can be found 2.4). Below is a brief diagram of general architecture and it will make more sense if you read with code in part 1 run_program function.



**Most Important Libraries**
1. Yafoo Finance to get data of stocks
2. Pandas to get data from Macrotrends / Modify table / Calculation
3. Streamlit for visualization

*main.py*

Streamlit Library

Calls "part1_user_input_visualization"

*part1_user_input_visualization*

Function "user_input function" takes user inputs and calls "run_program" function
Then, Part 2, 3, 4 is called and lastly it call "visualization" function
Then, figure and statistics are plotted with Streamlit

*run_program function*

Calls part 2, part 3, part 4. Then "visualization"

[Part 2] Choosing the Main Influential Factors
1. Filter stocks that's supported by API
2. Sort stocks based on their growth
3. Take top half of stocks. Now we have list of best stocks.
4. Determine the factors of best stocks. Sorted based on their values.

[Part 3] Recommendation Tree
5. Create a RecommendationTree using sorted factors
6. Insert all user input stocks to RecommendationTree with their corresponding factor values.
7. Their correlation value will be compared to best stocks average correlation value and determine whether to go left of right of subtrees.
8. Choose list of buy stocks by using user input risk percentage.

[Part 4] Investment Simulation
9. Run simulation of investment return of benchmark 'NASDAQ'
10. Run simulation of investment return of benchmark 'S&P500'
11. Run simulation of investment return of benchmark 'All User Input Stocks'
12. Run simulation of investment return of 'Recommendation Tree Filtered Stocks'
In other words, this is using statistically significant factors

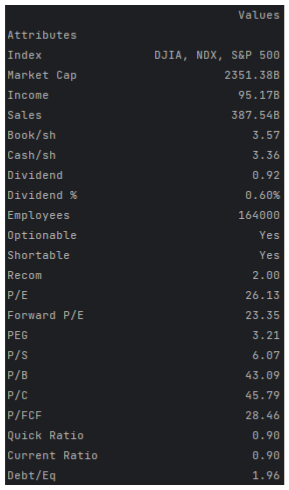
[Visualization]
13. Plot them into one figure using visualization function
Finally, return graph as index 0, and rest are statistics that will be used to prompt under the graph

## 2.1 Choosing main influential factors - part2_factor_processing

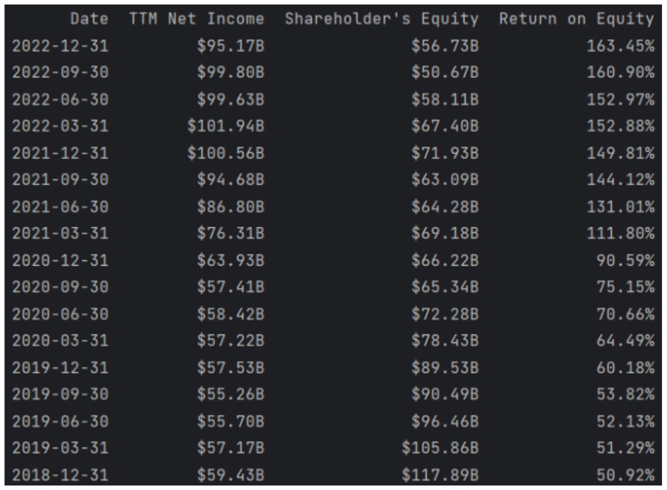
To calculate future investment stock returns, one of the foremost things to do is to get the dataset. This dataset would be obtained from "yahoofinancials" [JEC23], which is an Open Source, MIT Licensed library that uses an API to retrieve numerous information about the stock market from Yahoo Finance [23d]and also contains a variety of useful methods. Additionally, we will utilize a dataset from "Macrotrends"[23c], a research platform that enables users to screen and research stocks and global metrics. We obtained further datasets up to 2009 by using the Pandas library[23a] and its' "read_html()" function. We have found a tutorial regarding the extraction of data on a website written by Medium [Fin22].

This part of the module will filter out stocks does not have sufficient data from Yahoo finance, then using data from 2009 to the user input date, it will calculate the percentage of growth and sort in order of performance. This is done by using one of the yahoofinancials methods which is 'get_percentage_growth'. This function that takes in the stock name, start and end date and then returns how much the stock grew as a float value. After sorting the stocks in order of their growth, the 'top_half', function is used to cut down the list of stocks ad selects the top performing stocks.

The next imperative is to first identify the factors that influence investment planning. To determine these factors, the past datasets of the top-performing stocks are scrutinized. Then, the factors belonging to each stock such as "POE", "ROI" and more will be examined. The Pearson correlation[4] of the changes in these factors would be plotted against the increase of stock price per year for each of the chosen top-performing companies. This will be done using a method of pandas[23b] called 'corr' which takes the data frame and outputs a correlation value enabling us to evaluate the extent of correlation between each factor and the stock price. This will be repeated for each top companies and the average correlation value of factors will be taken. Below are the images of a sample dataset of a company and its various factors provided by the API. *(Dataset may differ according to actual results)*



```
                          Values
Attributes
Index          DJIA, NDX, S&P 500
Market Cap               2351.38B
Income                     95.17B
Sales                     387.54B
Book/sh                      3.57
Cash/sh                      3.36
Dividend                     0.92
Dividend %                  0.60%
Employees                  164000
Optionable                    Yes
Shortable                     Yes
Recom                        2.00
P/E                         26.13
Forward P/E                 23.35
PEG                          3.21
P/S                          6.07
P/B                         43.09
P/C                         45.79
P/FCF                       28.46
Quick Ratio                  0.90
Current Ratio                0.90
Debt/Eq                      1.96
```

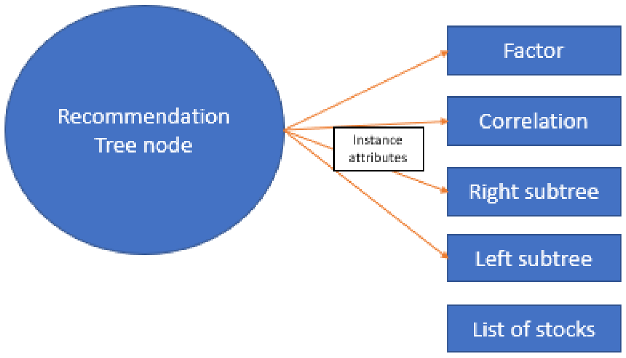


```
      Date  TTM Net Income  Shareholder's Equity  Return on Equity
2022-12-31         $95.17B               $56.73B           163.45%
2022-09-30         $99.80B               $50.67B           160.90%
2022-06-30         $99.63B               $58.11B           152.97%
2022-03-31        $101.94B               $67.40B           152.88%
2021-12-31        $100.56B               $71.93B           149.81%
2021-09-30         $94.68B               $63.09B           144.12%
2021-06-30         $86.80B               $64.28B           131.01%
2021-03-31         $76.31B               $69.18B           111.80%
2020-12-31         $63.93B               $66.22B            90.59%
2020-09-30         $57.41B               $65.34B            75.15%
2020-06-30         $58.42B               $72.28B            70.66%
2020-03-31         $57.22B               $78.43B            64.49%
2019-12-31         $57.53B               $89.53B            60.18%
2019-09-30         $55.26B               $90.49B            53.82%
2019-06-30         $55.70B               $96.46B            52.13%
2019-03-31         $57.17B              $105.86B            51.29%
2018-12-31         $59.43B              $117.89B            50.92%
```

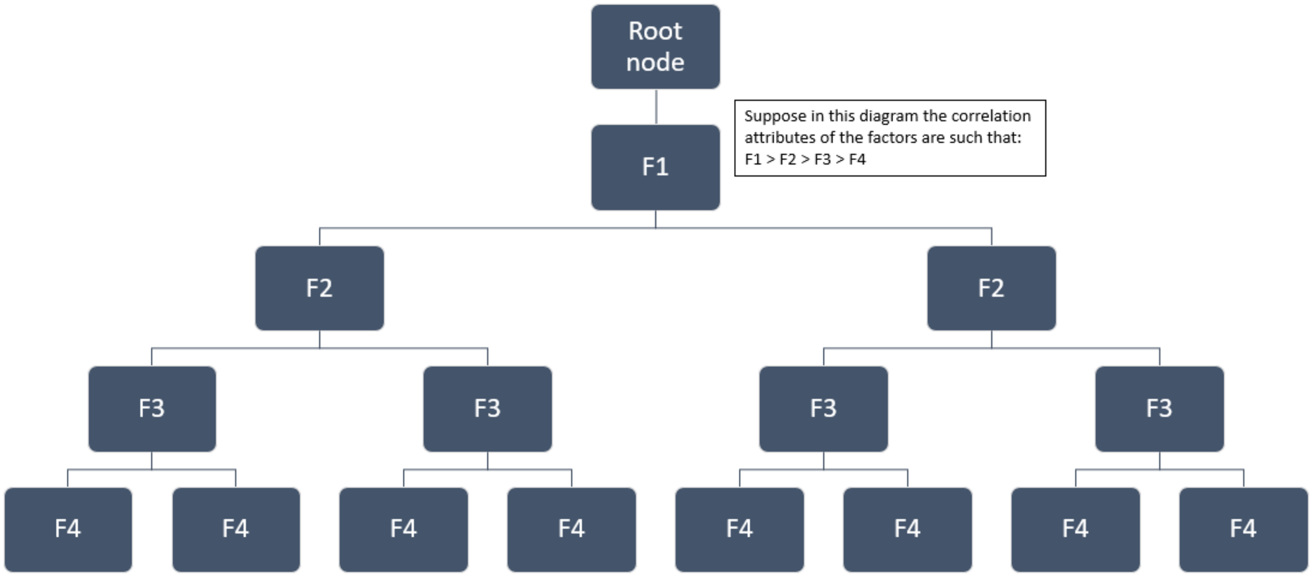
## 2.2 Generating the Recommendation Tree - part3_recommendation_tree

Now that the importat factors are known, we have to create a recommendation tree. The algorithm for creating a complete recommendation tree is inspired by generate_complete_game_tree from assignment 2, and it is customized to suit our RecommendationTree class. The recommendation tree is a tree data type that has four instance attributes: factor (string data type), left subtree (tree data type), right subtree (tree data type), correlation and list of stocks. The factor instance attribute stores the name of the factor which can be accessed while traversing the tree. Similarly, the correlation instance attributes store the average correlation of each factor. The left subtree and right subtree are the left and right child of a parent tree. The list of stocks instance attribute is only used in the leaf nodes where the final list of stocks is stored after classifying. Below is a visual diagram of a single tree node.

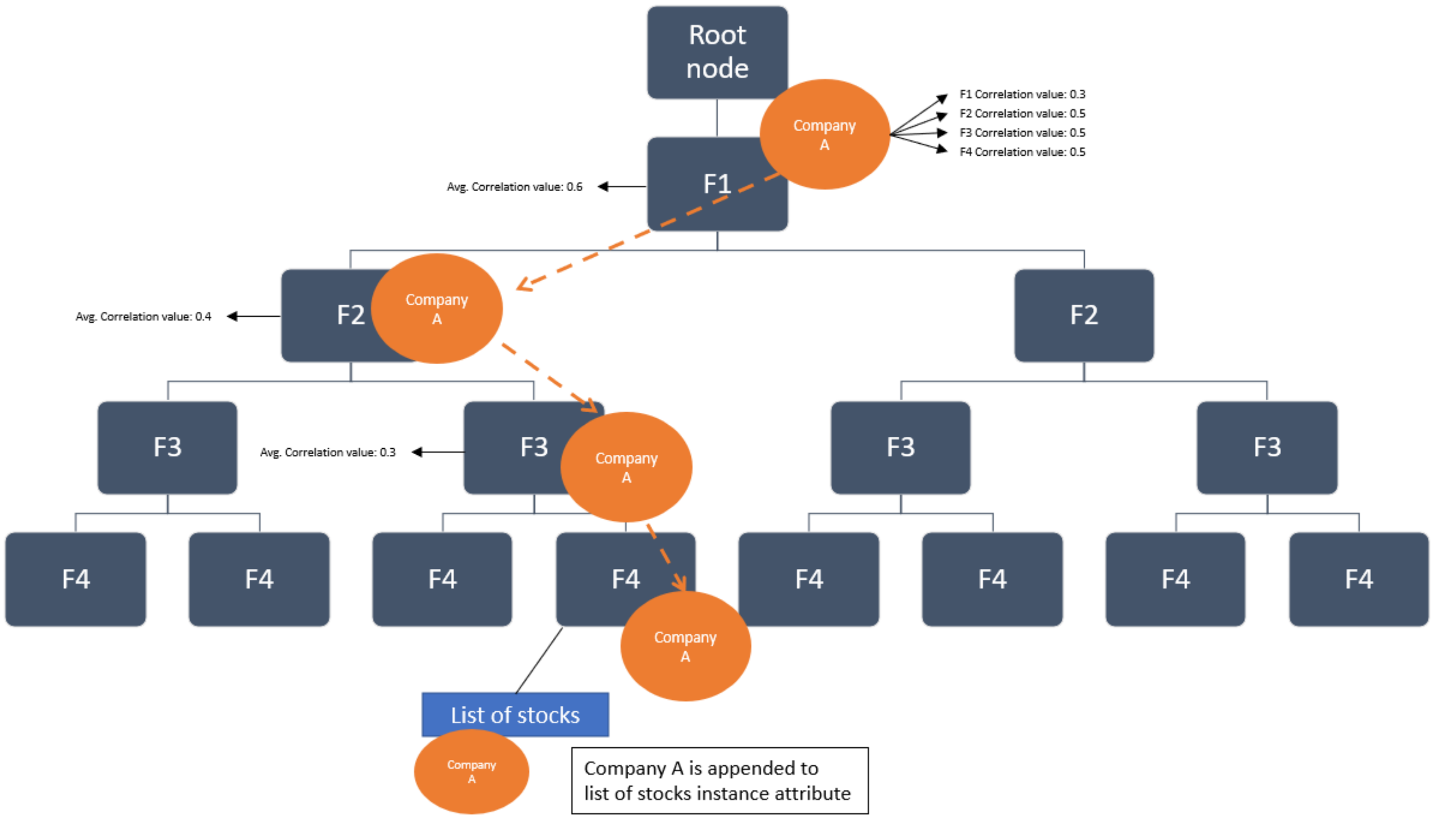
[4]Information on why Pearson correlation is chosen on Appendix 6.4



The recommendation tree is built by placing factors in a parent-child relation in the tree depending on the correlation value. For example, suppose the four factors are F1, F2, F3 and F4. F1 would be the parent of F2 if F1 has a higher correlation of F2. F2 will be the parent of F3 if F2 has a higher correlation value than F3 and so forth. Below is a diagram that demonstrates how a recommendation tree is built

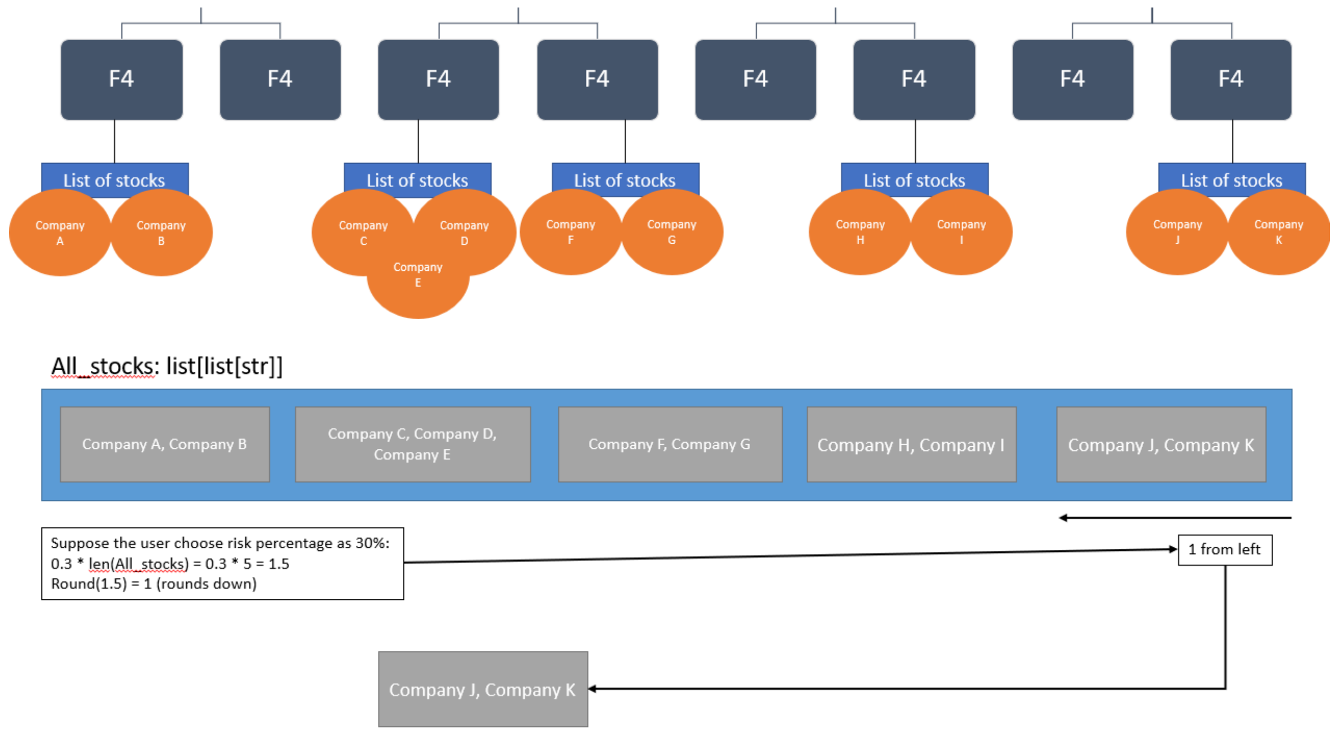

Root node
Suppose in this diagram the correlation attributes of the factors are such that: F1 > F2 > F3 > F4
F1
F2
F2
F3
F3
F3
F3
F4
F4
F4
F4
F4
F4
F4
F4

Next, the values that would decide whether to invest in the company or not needs to be found. Suppose we are placing company A into the subtree. It gets inserted into the recommendation tree and arrives at the F1 factor. To decide left or right, company A's F1 factor in correlation to the stock price is taken. If the F1 factor correlation of company A is less than the average F1 correlation value, it traverses to the left subtree. Otherwise, it goes into the right subtree. Company A now arrives at the second intersection and repeats these same steps until it arrives at a leaf. This tree diagram allows for easier classification when choosing the optimal company to invest in. Below is a sample diagram of how this classifying function works:


Root node
Company A
F1 Correlation value: 0.3
F2 Correlation value: 0.5
F3 Correlation value: 0.5
F4 Correlation value: 0.5
Avg. Correlation value: 0.6
F1
F2
Company A
Avg. Correlation value: 0.4
F2
F3
Avg. Correlation value: 0.3
F3
Company A
F3
F3
F4
F4
F4
F4
F4
F4
F4
F4
Company A
List of stocks
Company A
Company A is appended to list of stocks instance attribute

## 2.3 Choosing the Stocks - part3_recommendation_tree

With all the user input companies, the tree diagram would append them to a list in order of left to right. After all this is done, we call a function that gets appends all the list of stocks in the leaf nodes into a list (therefore the output is a list of lists). For simplicity, let's assume this output as a variable name 'All_stocks'. Next another function would take the risk percentage and multiply it by the length of 'All_stocks' and rounds it up. Assume this variable as 'i'. Finally, the program chooses the number of 'i' elements starting from the right of 'All_stocks' A diagram of this process is attached below for better visual clarity:



*At this point, you can assume the most far right is our top choice and the far left is the worst list of stock. In this example, the program will invest in Company J, K.

### 2.4 Simulation - part4_investment_simulation

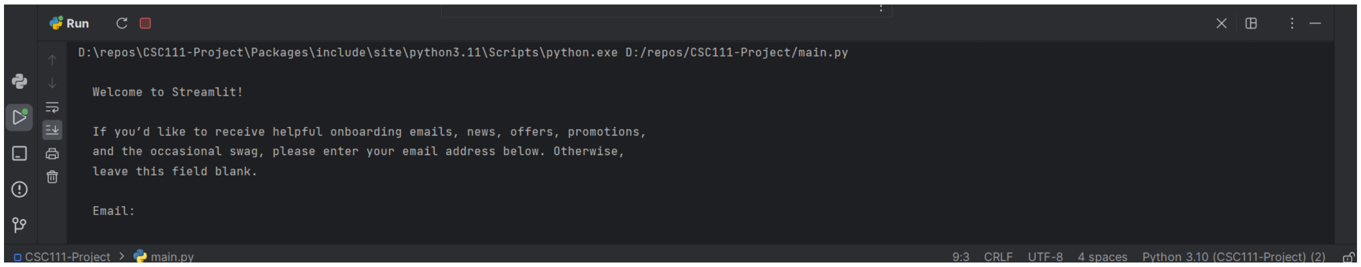
The selected stocks are then invested from the starting date provided given by the user during part 4. Then a function called 'recommendation_tree_simulation' invests them starting from the date given by the user until the current year which is 2023. It then outputs the values in a dictionary where the keys are the number of years and values are the percentage return over investment. Next, we need a comparison to understand how good our function is. Hence, another function called 'benchmark_simulation' is made which invests in takes input from the list of stocks and outputs its keys as the number of years and values as percentage return over investment. This function 'benchmark_simulation' is used to run the original list of stocks the user gave, as well as the NASDAQ and S&P500. In comparison, the recommendation tree only invests in what is determined as 'good stocks' from the list of stocks given by the user. The values of all these three as well as the output from 'recommendation_tree_simulation' is then plotted as line chart which is figure data type.

### 2.5 Visualization - part1_user_input

The Streamlit library is in a separate file called 'part1_user_input_visualization'. This module is solely responsible for running functions from part 2, part 3 and part 4 and then getting the line chart as figure data type. This is then displayed in a browser. One of the reasons of using Streamlit is that it allows users to input values directly from the browser and the user can simply click 'Run Program'. Please refer to **Section 3: User Instructions** for more details about the graph. Additionally, the PIL library has been imported to allow images to be displayed in the browser.

## 3 User Instructions

Firstly, make sure theme.png, good_bad_stocks.csv, part1, part2, part3, part4, main, and requirements files are at the same level. Then, download all required libraries from the requirements.txt. Then, run main.py. If the user is running Streamlit for the first time, Streamlit will prompt you to enter an email address to receive newsletters as shown in the image below.



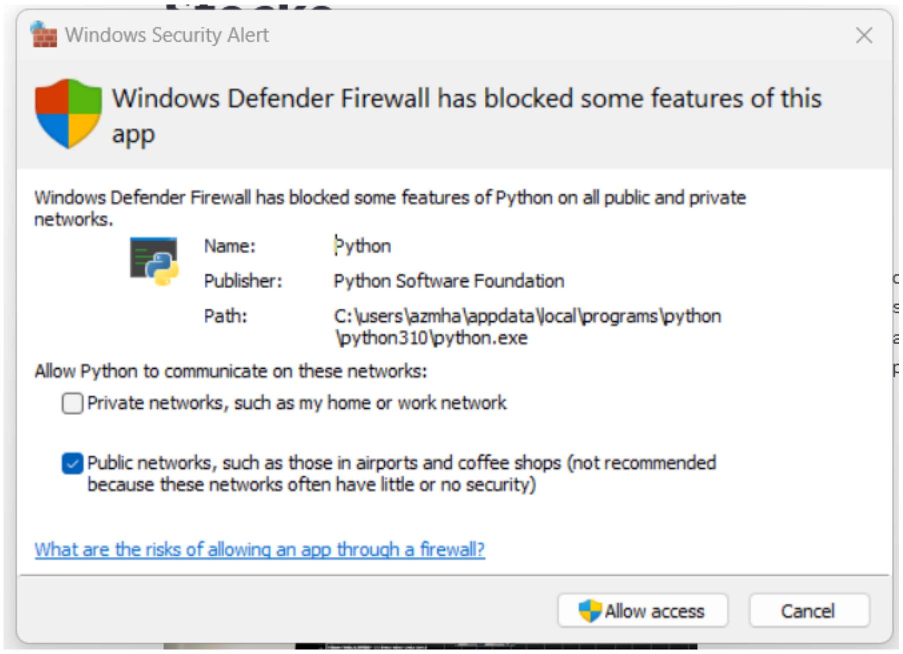
```
D:\repos\CSC111-Project\Packages\include\site\python3.11\Scripts\python.exe D:/repos/CSC111-Project/main.py

  Welcome to Streamlit!

  If you'd like to receive helpful onboarding emails, news, offers, promotions,
  and the occasional swag, please enter your email address below. Otherwise,
  leave this field blank.

  Email:
```

Skip this by clicking enter. (If this doesn't prompt and an error message is raised regarding python-ta, you need to comment out python-ta, then run again. This is simply because you are running Streamlit for the first time and it wants you to register, but python-ta causes an error regarding this registration process. But you keep python-ta once you are registered / or skipped registering.) Users might also get a prompt to allow firewall permissions. Just leave it on default and click on 'Allow access'.



Windows Security Alert

Windows Defender Firewall has blocked some features of this app

Windows Defender Firewall has blocked some features of Python on all public and private networks.

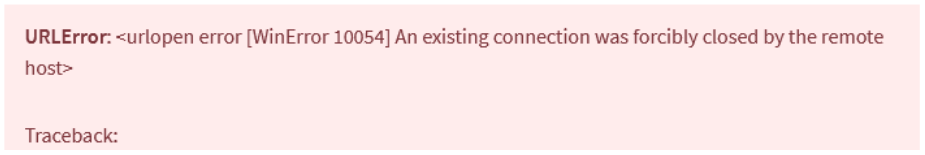
Name: Python
Publisher: Python Software Foundation
Path: C:\users\azmha\appdata\local\programs\python\python310\python.exe

Allow Python to communicate on these networks:

- [ ] Private networks, such as my home or work network
- [x] Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing an app through a firewall?

This will open up a browser that allows the user to input values and run the simulation. The user should choose a list of stocks, start date, risk percentage, and factors. For the list of stocks, as explained in **Section 4.2**, the user can run as auto, options (that allows users to choose stocks from the drop-down menu) and manual (that allows users to manually type stocks). We highly recommend avoiding using the manual option. Use either auto or options. (But since we already set all the default settings, you can skip all processes of selecting options.)
If you are running with check_contracts, choose a small number of factors, as many factors lead to a significant increase in running time due to the nature of the Recommendation Tree. Without check_contracts you can select all without worrying about running time. We recommend you comment out python-ta while running Streamlit, if not, it will keep opening up the python-ta due to Streamlit refreshing. After all the required values are given, click on 'Run Program' button and Streamlit should return the results in approximately 5 minutes. Note that pressing the **run button once is enough**. Pressing it multiple will cause the program to rerun which would result in even longer run times. The program is also coded to run with default values in the case of the user forgetting to specify a variable. **Please ensure that the user has a stable internet connection**. If a decent internet connection is not established the function will not run and may cause errors. The images below show the errors that the user will encounter without a proper connection.



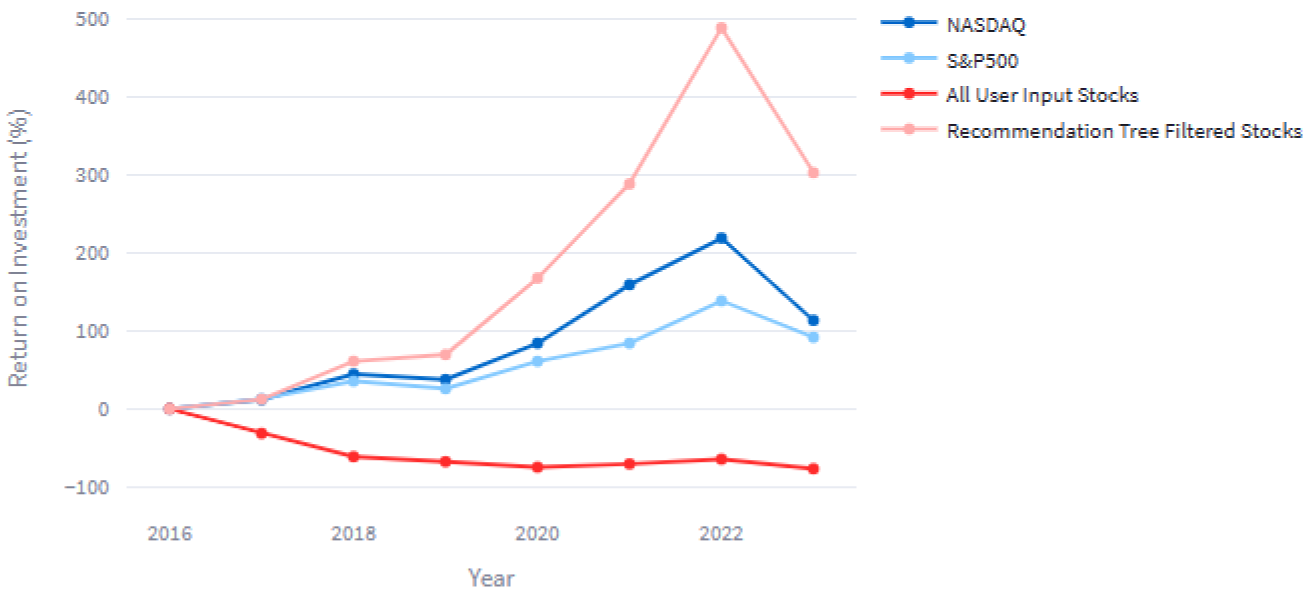
```
URLError: <urlopen error [WinError 10054] An existing connection was forcibly closed by the remote host>

Traceback:
```

With a proper connection, the user will see a graph displayed. The graph for display by the Streamlit outputs four distinct lines. It will show the return on investment for S&P500 and NASDAQ. It will also show what happens if a user buys all stocks in the given as the input represented by the 'All User Input Stocks' line. The line shown as 'Recommendation Tree Filtered Stocks' is the return over investment of what happens if the user inputs a list of stocks but the program only invests in what it determines to be 'a good stock'.

Here's what our program looks like when we input selecting all stocks, default date, risk percentage of 1, and all factors. It performs better than the S&P500 because it only takes the best-performing stocks based on their correlation value and invests in them. The program will also prompt ranked factors such that the user can identify which factors were secrets of successful stocks:

**Simulation Results**



Statistics:

Some stocks are not supported by APIs, thus, the program had to filter out the stocks. Here are stocks that were used to train the model: ['AAPL', 'MSFT', 'GOOG', 'AMZN', 'NVDA', 'TSLA', 'META', 'BABA', 'CRM', 'AMD', 'INTC', 'ATVI', 'EA', 'ZG', 'YELP', 'SQQQ', 'PCG', 'FSLR', 'MUR', 'MOS', 'DVN', 'MRO', 'RIG', 'RRC', 'FLR']

Here are ranked factors and their correlation values that were used to determine list of buy stocks: [('debt-equity-ratio', -0.18053667044548966), ('quick-ratio', 0.016785617960122918), ('shares-outstanding', 0.14755808873446238), ('pe-ratio', 0.15250369980239672), ('current-ratio', 0.19232333271614122), ('net-income', 0.33839538723714047), ('roa', 0.34468886977501284), ('roi', 0.35517620692519464), ('return-on-tangible-equity', 0.38707257503035447), ('roe', 0.4488235919296114), ('price-book', 0.47055364850580944), ('total-share-holder-equity', 0.5347582446258101), ('cash-on-hand', 0.6281679140595842), ('number-of-employees', 0.6750023142643624), ('price-sales', 0.6872237501959543), ('revenue', 0.7276925947033279), ('gross-profit', 0.8135327497120921), ('total-liabilities', 0.8954785853567573), ('average-price', 0.9543480036762754)]

Here are list of stocks that program decided to invest: ['AAPL', 'MSFT', 'GOOG']

# 4 Changes Made to the Initial Proposal

## 4.1 Regarding data processing and calculations

In our first proposal, we mentioned using standard deviation to obtain a correlation value for each factor which would help determine whether the recommendation tree would traverse to the left or right subtree. We decided against this and only took average values instead because standard deviation assumes the normal distribution pattern in the values. Our correlation values are obtained but examined the factors with top companies and do not have much to do with the normal distribution pattern.

## 4.2 User Input

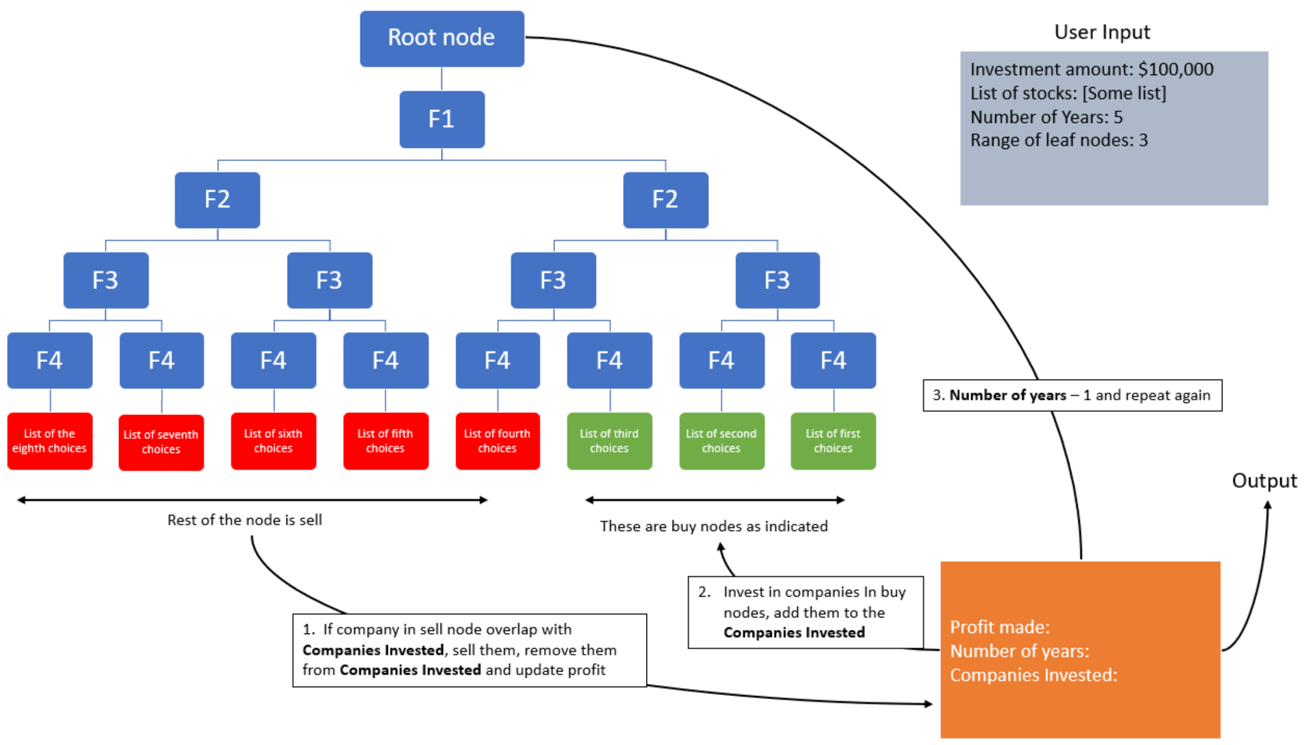
Second, we have changed what our user input would be for the function determining_buy_stocks. Originally, our idea was to have an input called 'range of leaf nodes' in which the user would input a number that would determine the range of leaf nodes chosen. However, through our simulations and testing, we found that for our submission, we changed our input and called it 'risk percentage'. The reason is that we would not have to limit this user input since we can calculate which stocks to choose based on the Recommendation Tree as the number of leaf nodes is dependent on the number of factors of user chooses.

User input was also slightly modified where in the original, the user is solely responsible for entering the list of stocks to invest, the number of years and factors they chose. However, we realized that we do not know the amount of knowledge the user has about the stock market. Therefore, we slightly modified the inputs by adding default options that allow user to run in the 'Auto' mode where a pre-selected list of stocks is available for the users to directly run. In good_bad_stocks.csv, we included around half companies with good stock and another half of companies with bad stock to have a balanced list of stocks of good and bad. Users can choose to use this list of stocks. We also have an 'Option' mode where the user can select multiple stocks from a drop-down menu. Finally, for users with expertise in the stock market, we included a manual mode where a company's stock name can be typed manually.

Another input we removed was the investment amount because we would like the user to understand how much return they are making. If the investment amount was included just by returning the amount made, the user's perception of how much profit might be influenced by how much they initially invested. Therefore, by removing the investment amount and just returning the percentage of return over investment, we are able to present a better picture of how much profit is made to the user.

## 4.3 Changes to Simulation

In the previous proposal, we decided to run the recommendation tree each year for the number of years given by the user. Then, we would buy and sell every year according to the recommendation diagram. Below is what we initially proposed:



Root node
F1
F2
F3
F4
List of the eighth choices
List of seventh choices
List of sixth choices
List of fifth choices
List of fourth choices
List of third choices
List of second choices
List of first choices
Rest of the node is sell
These are buy nodes as indicated
User Input
Investment amount: $100,000
List of stocks: [Some list]
Number of Years: 5
Range of leaf nodes: 3
3. Number of years – 1 and repeat again
Output
1. If company in sell node overlap with Companies Invested, sell them, remove them from Companies Invested and update profit
2. Invest in companies In buy nodes, add them to the Companies Invested
Profit made:
Number of years:
Companies Invested:

After much testing and simulation, we decided that we would only run the program once during the initial year of investment (given by the user) to determine which stocks to buy and hold these stocks for until the current year instead of buying and selling each year. This was agreed upon a long discussion with group members as it would be a better representation of whether our program performs well or not.

## 5 Discussion

### 5.1 Goals and Insights

Our team successfully used a tree data structure to identify the promising factors and the most likely companies to invest in and utilize the backtesting approach to determine the profitability of the investments.

We explored the question: which stocks are most profitable in the long-term, and what factors contribute to their success? Based on the result of the simulation, we were able to determine the most promising stocks for investment by analyzing the correlation between their performance metrics. Thus, we were able to find the secrets of successful stocks by studying factors' correlation values using a tree data structure.

The program ranks the stocks using a binary tree to group the stock by ranking for investment. Then it uses the recommendation tree and invests according to what it suggests and draws conclusions on its performance. As an example, it performs better than the S&P500 as it only takes the best-performing stocks based on their correlation value and invests in them. Then, it prompts the factors in ranked order, enabling users to identify what factors contribute to their success. From the simulation results image from page 9, based on that one example, we can determine that 'average-price' had a 0.95 correlation value, meaning 'average-price' above 0.95 is a strong sign of good stock. Other factors like 'total-liabilities', 'gross-profit', 'revenue' also had a statistically significant contribution to the success of stocks, and that 'AAPL', 'MSFT' and 'GOOG' are the ideal stocks to invest in.

After a series of debugging and extensive testing done to the program, we came to a few conclusions regarding the program. Our program compares the current trends of stock market and invests accordingly which means a user investing with our program will not experience significant risks. However, sudden disruptions would greatly affect the profitability of our program. An example of this would be the COVID-19 pandemic. Our program saw a significant decrease in profit and sometimes even losses during the years between 2019 - 2021 as it was not able to predict the most viable stock to invest in the pandemic era by just using previous trend data.

## 5.2 Limitations and Obstacles Encountered

One obstacle we encountered was our data was shown originally. When extracting data from the Macrotrends website, we found out there was unnecessary data being shown that would slow down our program. We overcame this with our 'obtain_factor_data' and 'clean_and_merge' functions. After extracting our data and adding it into a Pandas dataframe, we isolated which parts of data we wanted by using indexing and try and except for errors. While working on finding the correlation by using the merge function, it would return an empty dataframe. This occurrence happened because some of the data was blank or NaN. Our solution was to use the function 'dropna'. We also had issues with pandas' read_html() function. The try and except function that we wrote for our function did not work. An additional import 'from lxml import etree' had to be imported to catch pandas' read_html() errors.

At one point, we considered using only Plotly for our plotting, but we wanted to have the user be able to add or remove inputs and the graph would update in real-time. We realized that Plotly by itself does not have the capabilities to do what we wanted. Ultimately, we figured out Streamlit was a good option for our plotting but we later realized that the user has to run commands into cmd when running Streamlit.

Another problem we encountered was the python-ta's check_contracts. Before adding check_contracts to our RecommendationTree class, we had no issues. However, while testing RecommendationTree after adding check_contracts, we encountered AssertionErrors. The error claimed that our return value did not match the return type annotatiton even when it did. After some enquiries, our Professor advised that the typeguard library is the issue. To fix our problem, we had to specify the typeguard library version to 2.13.3, not 3.0.x.
Some stocks were not supported by the Macrotrends website and the Yahoo Finance library which caused issues. So we had to filter out the stocks that are not supported by either of them. Now everything is fixed withour errors when we run with check_contracts.

Python-ta caused many issues in which it would return an error message saying the 'ClassDef' object has no attribute 'value' and contact the instructor with code. We realized that using the iloc function from pandas was causing this error so we decided to use the filter function instead.

## 5.3 Further Exploration Options

After testing and running our program, the project has shown promising results. One method to expand our project is to run our program to actually buy stocks from the stock exchange instead of a simulation. We can have the AI buy, hold and sell stocks every year and see the long-term results. However, before implementing this, we would need to extensively test out our program and ensure that losses are not made. Additional inspections must also be made to ensure that our program is secure from malicious actors.

Additionally, when selecting companies to determine the most influential factors, concentrating solely on stock price as the lone factor can result in a limited understanding of what makes a "top" company. Numerous profitable businesses are respected and highly valued in their various sectors even though they may not have the highest stock prices. While the stock price is an important indicator, it is not the only one that impacts a company's performance or worth.

We can also make improvements with using APIs. APIs we used like Yahoo Finance and Macrotrends did not support some stocks and we had to handle error cases and drop NaN. Instead, we could look for better supportive APIs that will far better suit the project's purpose.

Lastly, we can make improvements with statistics results. If we're changing the program with AI deciding to buy, hold, and sell every year, we could've calculated the win/loss ratio, max drawdown, and more in-depth analysis of factors' correlation values.

## 6 Appendix

### 6.1 Regarding PER and PBR

If PER means "how much this company earns and the stock price is this much", PBR is the concept of "how much this company owns and the stock price is like this".

Some companies have a high PBR due to the nature of the industry. Ex) Bio companies; since they do not need much land due to the nature of the research company. In addition, AAPL (Apple) has a high PER and PBR, not because it's an indication of a bad company, but it's an indication of AAPL being an expensive stock. So PER and PBR don't always indicate whether the company is bad or not.

### 6.2 Regarding ROE

ROE is what Warren Buffet—world's greatest investor—enjoys using. He advises investing in companies with a ROE of 15% or higher and steadily increasing within the last three years [Tea23].

ROE or the return on equity can be calculated by net income / shareholder's equity.

### 6.3 Additional Factors

Some other factors that might be important are dividend yield, debt-to-equity (D/E), profit margins, etc. Dividend yield is also one of the important factors. It measures the amount of cash dividends distributed to shareholders in relation to market value per share. This is used to evaluate the income potential of a stock investment. Higher is better for investors.

### 6.4 Regarding Mathematical Analysis

For the correlation, we have decided to use the Pearson correlation as we are analyzing the linear correlations instead of the monotonic relation [al23] [Bro23].

## References

[23a] *Pandas Documentation.* Accessed: March 5, 2023. URL: `https://pandas.pydata.org/docs/#pandas-documentation`.

[23b] *Pandas.DataFrame.corr.* Accessed: March 5, 2023. URL: `https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html`.

[23c] *The Long Term Perspective on Markets.* Accessed: March 5, 2023. URL: `https://www.macrotrends.net/`.

[23d] *Yahoo Finance.* Accessed: March 5, 2023. URL: `https://finance.yahoo.com/`.

[al23] user3636user3636 et al. *How to Choose between Pearson and Spearman Correlation?* Accessed: March 5, 2023. URL: `https://stats.stackexchange.com/questions/8071/how-to-choose-between-pearson-and-spearman-correlation%5C#:%5C~:text=The%5C%20difference%5C%20between%5C%20the%5C%20Pearson,measurements%5C%20taken%5C%20from%5C%20ordinal%5C%20scales.`.

[Bro23] Jason Brownlee. *How to Calculate Correlation Between Variables in Python.* Accessed: March 5, 2023. URL: `https://machinelearningmastery.com/how-to-use-correlation-to-understand-the-relationship-between-variables/`.

[Fin22] T-Ballz Finance. *Retrieving Historical P/E Data with Python.* Medium. 2022. URL: `https://medium.com/@tballz/retrieving-historical-p-e-data-with-python-d09198335984`.

[JEC23] JECSand. *Jecsand/Yahoofinancials: A Powerful Financial Data Module Used for Pulling Data from Yahoo Finance. This Module Can Pull Fundamental and Technical Data for Stocks, Indexes, Currencies, Cryptos, Etfs, Mutual Funds, U.S. Treasuries, and Commodity Futures.* `https://github.com/JECSand/yahoofinancials`. Accessed: March 5, 2023.

[Tea23] The Investopedia Team. *What Is Warren Buffett's Investment Startegy?* Accessed: March 5, 2023. URL: `https://www.investopedia.com/articles/01/071801.asp`.