Financial Data Compilation: Database & Pipeline Architecture

Big Data Technology
May 5th, 2024

BBA DBA - Group 8
José Tomás Burgillos, Ernests Jansons, Antonino Sistac, Fabián Vagnoni, Daniel Yebra

Table of Contents

1. Motivation and Rationale

With the immense amount of information that is accessible in the modern day, one can only feel overwhelmed when trying to confirm a hypothesis or when working on a data-driven project. While browsing through numerous articles and Statista pages to find the necessary information can be time consuming and, sometimes, quite bothersome, many times this overload of information is not even the main problem, but rather an assumed scenario. What then overtakes the first spot on the leaderboard of problems is finding out whether the information that has taken so long to find is even accurate and actual. In many cases, the article that seems to be so useful is accurate, but after some extended research, it is more likely than not that at least some of the information is out of date or worse, completely irrelevant. Similarly, a different article that has been published in the same week has to be actual, meaning the information contained within is up to date, but sooner or later it is discovered that the figures and facts presented are inaccurate. Thus, there is always a tradeoff between reliability and recency in information.

Possessing reliable data is crucial for making smart business decisions, both for individuals and companies of any sector of the economy. Professional investors are no exception to this rule. Thus, we created a method for harnessing comprehensive information about all the companies that compose the S&P 500 index. This data includes stock prices, trading volumes, and a series of financial ratios like earnings per share (EPS), price-to-earnings (P/E) ratios, and more. These are essential when assessing company performance. The project's main goal is to use this harnessed data to optimize investment portfolios. This involves collecting and analyzing the data to identify useful insights relevant to decision-making and investment strategies. Data will allow us to evaluate the financial health and potential growth of the companies that compose the famous index. For instance, financial ratios provide a deep look into the company's liquidity and operational efficiency. By combining this data, we seek to build a portfolio that balances risk and reward.

The project focuses on the S&P 500 index to create a representative dataset of U.S. leading market sectors and industries, which is crucial to apply portfolio diversification and be able to manage risk effectively. The index is not just a benchmark for U.S. equity performance, but also reflects the country's economic landscape, making it ideal to ensure a diversified portfolio. However, it is acknowledged that the inclusion of only U.S. stocks might seem arbitrary and counterproductive, since many other competitive and well-performing markets exist beyond the frontiers of the North American country. Nevertheless, the project aims to produce a minimum viable product, so simplicity and functionality are key. Once the software is sufficiently reliable, it could be easily scaled to include further stocks. Hence, this work is also focused on building a scalable product.

With this approach, we aim to show how a data-driven project can significantly enhance decision-making in financial decisions. The technical aspirations of the project are ambitious, driven by the wish to better our technological capabilities to improve our financial analysis. We have created a pipeline in python, in order to compile the companies' data directly from the Yahoo Finance website into a JSON, which we intend to pipeline into HDFS. We have chosen HDFS because of its ability to manage large datasets. It will allow us to efficiently manipulate the data, even as the project evolves and new types of data are incorporated. This process is instrumental in achieving our ultimate objective: create a portfolio optimizer.

Our decision to pursue this project was inspired by our passion for the financial markets, and the great potential of data analytics in the field of financial analysis. Our project aims to show how individual investors, who sometimes find themselves lost in the vast sea of financial data, can leverage programming tools to improve their analysis. Additionally, the creation of this tool has been very fulfilling to us, who have been able to apply the theoretical knowledge learned in different subjects and courses and apply it in a practical setting. This project, with its technical challenges, offers a unique point-of-start to contribute to the financial community, showing the impact of a data-centric approach on investment strategies and decision-making processes.

2. Value Creation

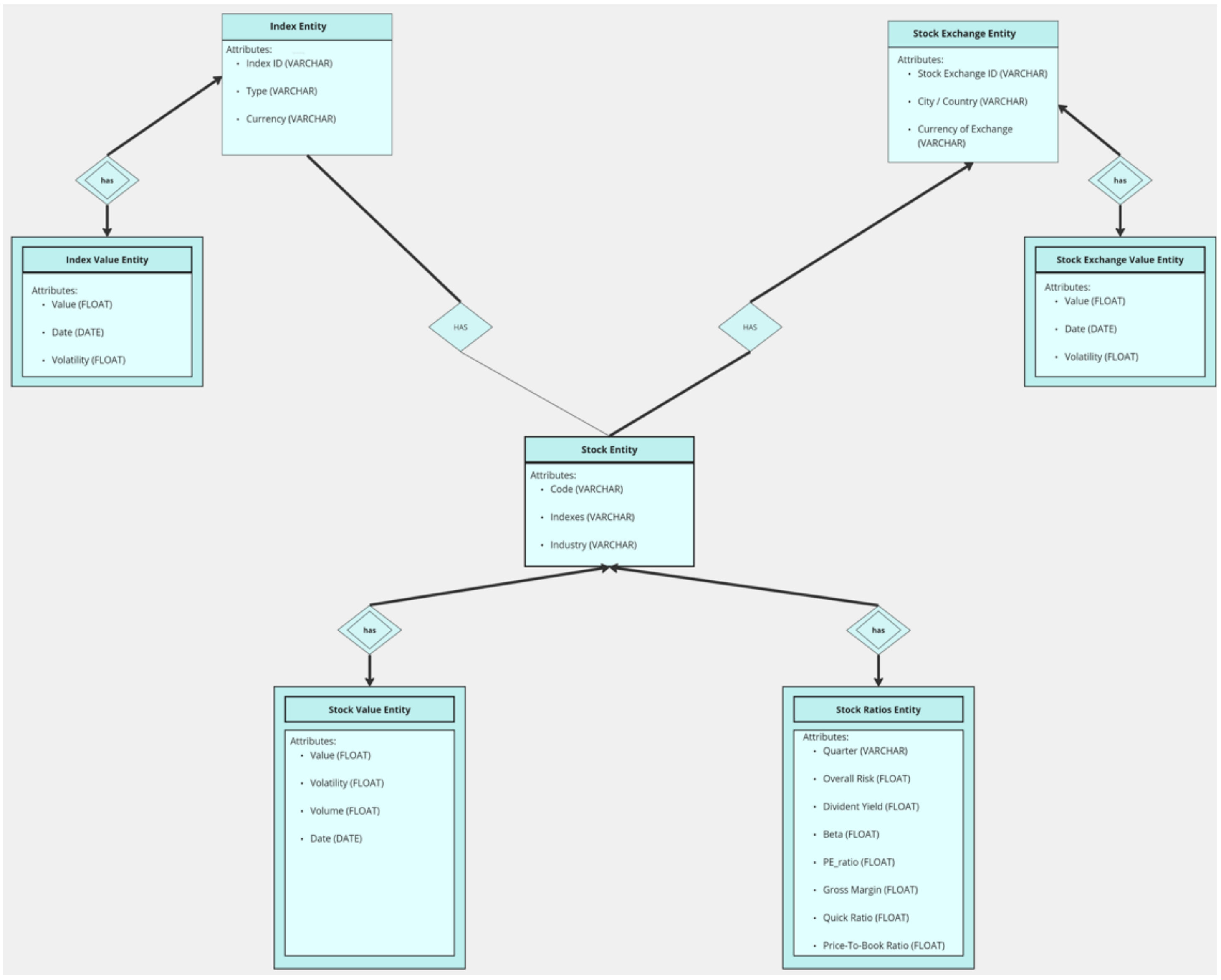
The amount, accuracy, and actuality of information plays an even larger role when money, or anything else whose loss would be unpleasant, is involved. The securities market is one such instance where information is abundant, much of it irrelevant, and the remaining part quite likely inaccurate. This presents certain problems in which basing decisions off of information that has any of those characteristics is likely to lead to negative returns in the long run. Firstly, if one was to try to predict the future price direction of a security, it is necessary to use the most recent information, and, given our previously mentioned dilemma, that information is likely to be incorrect. Secondly, the abundance of information likely means that some form of aggregates have to be used which are either behind stiff paywalls or non-existent. Therefore, while markets provide great profitable opportunities, the extent of that profitability, more often than not, is based on the kind of information one has access to, and mostly, that information lies in the hands of a select few individuals. The goal of this project, in terms of value creation, is to democratize that information to more people than before, so that they too can make more accurate and fact-based decisions.

While general information about individual securities can quite easily be found out by heading to one of the numerous finance websites, it is quite time consuming to find those figures for a basket of securities. In other words, to analyze 10-15 companies within the Healthcare sector, one would need to go through them one-by-one, note down the information on paper or an Excel sheet, and compare them afterwards, all of which would take quite a while. This is something that team members from this project have experience at a personal level while working and, while this is somewhat reasonable and doable for the aforementioned 10-15 companies, it becomes quite impossible to be done for a larger number of companies. Hence, the main way in which our project's end-result adds value is by providing relevant ratios and other sorts of information about all the companies within the S&P 500. This means that now the user can, without any advanced technical knowledge, upload it all onto excel and compute valuable aggregates from information that is both accurate and actual. Similarly, since we provide most of the base figures about the relevant securities, users are also free to construct their own ratios that they deem relevant to their use cases. All of this does not only save the user valuable time, but may also enable them to access figures that they previously couldn't or figures that were behind a paywall, in the end, leading them to make more fact-based, and hopefully, more profitable decisions.

To elaborate more on use cases and the various ways that the project would lead to value creation, consider an analyst who is evaluating the Healthcare sector. Previously, the analyst used to go through various companies, but because there are thousands of them, he only focused on those with the largest market capitalization, and while the strategy worked a few times, in many instances the truly profitable opportunities were missed solely due to the reason that medium market cap companies have a significantly larger prospect for growth than the large cap companies which the analyst considered. With our database, which is always kept up-to-date and, to the best of our knowledge, has always been accurate, the analyst need not limit themselves to just a few large cap companies. The same analysis methodology can now be applied to all relevant companies within any sector, without any meaningful lost time.

3. Database Design

In the database's ER Model (Figure 1.0 & available as an image file in the folder Design) there are seven entities and six relationships. The most relevant entity for our analysis is the Stock Entity and its weak entities, Stock Value and Stock Ratios (This will be called Value Entity and Ratios Entity from now on for simplicity's sake). The Stock Entity is a strong entity, since it does not need any extrinsic attribute from another entity to be uniquely identified. Conversely, the Value and Ratios entities are weak entities because for their unique identification they need the attributes of the Stock Entity. That is, if one was to retrieve some records from the Value Entity, for them to be actually useful first one must know the stock ticker or stock name to which these records belong. Since this information is only found in the Stock Entity, the Value Entity is a weak entity. Because of this, as it would be evident, the relationship Stock Entity - Value Entity and the relationship Stock Entity - Ratios Entity are relationships with cardinality constraints 1:1 or one to one, since one stock can only have a set of values and a set of values can only belong to a stock. Also, in both relationships there is total partition from each part of the relationship because a stock must have a set of values and a set of ratios, and vice versa.



It is noticeable that the Stock Entity could actually englobe all the attributes of the Value Entity and the Ratios Entity in a single conjuncted entity. Nevertheless, it was deemed better to have them split in these three different entities for a simple reason: updates frequency. The Stock Entity would hardly ever be updated, since the name, ticker and exchange of a stock practically never change and the indexes in which this stock is involved tend to change once every few years at maximum. On the contrary, the ratios of the same stock change every quarter, when the firm reports results, and, therefore, new information would have to be appended to this entity frequently. In an even more extreme case, the attributes found on the Value Entity change constantly and, therefore, this entity would have to be modified every day. Hence, the reason for these entities to be split is that they differ dramatically in their frequency of required modifications and, consequently, it is more efficient to have them divided so, when altering one, the others do not need to be brought to memory. All these explanations are parallel to the reasons why the Exchange and Index Entities have a relationship with a weak Value Entity. However, since these entities, Exchange and Index, are not firms, including a Ratios Entity would be senseless. Ratios are measurements based on the financial statements of a firm derived from its operations. Indexes and Exchanges do not have operations and, consequently, they have no ratios.

Regarding the relationship Stock Entity - Index Entity, no cardinality constraint is imposed to any of the parts. That means that this is a N:M or many to many relation. This is justified because a stock can be part of multiple indexes and, simultaneously, an index must engage multiple stocks. Additionally, it should be mentioned that the partition of the Index Entity in this relationship is total, since an Index must be related to stocks. However, the partition of the Stock Entity in this relationship is partial because a stock is not always related to an index. Nevertheless, it is worth noting that this hardly ever is the case, usually all stocks, at least all relevant ones, are related to one or more indexes.

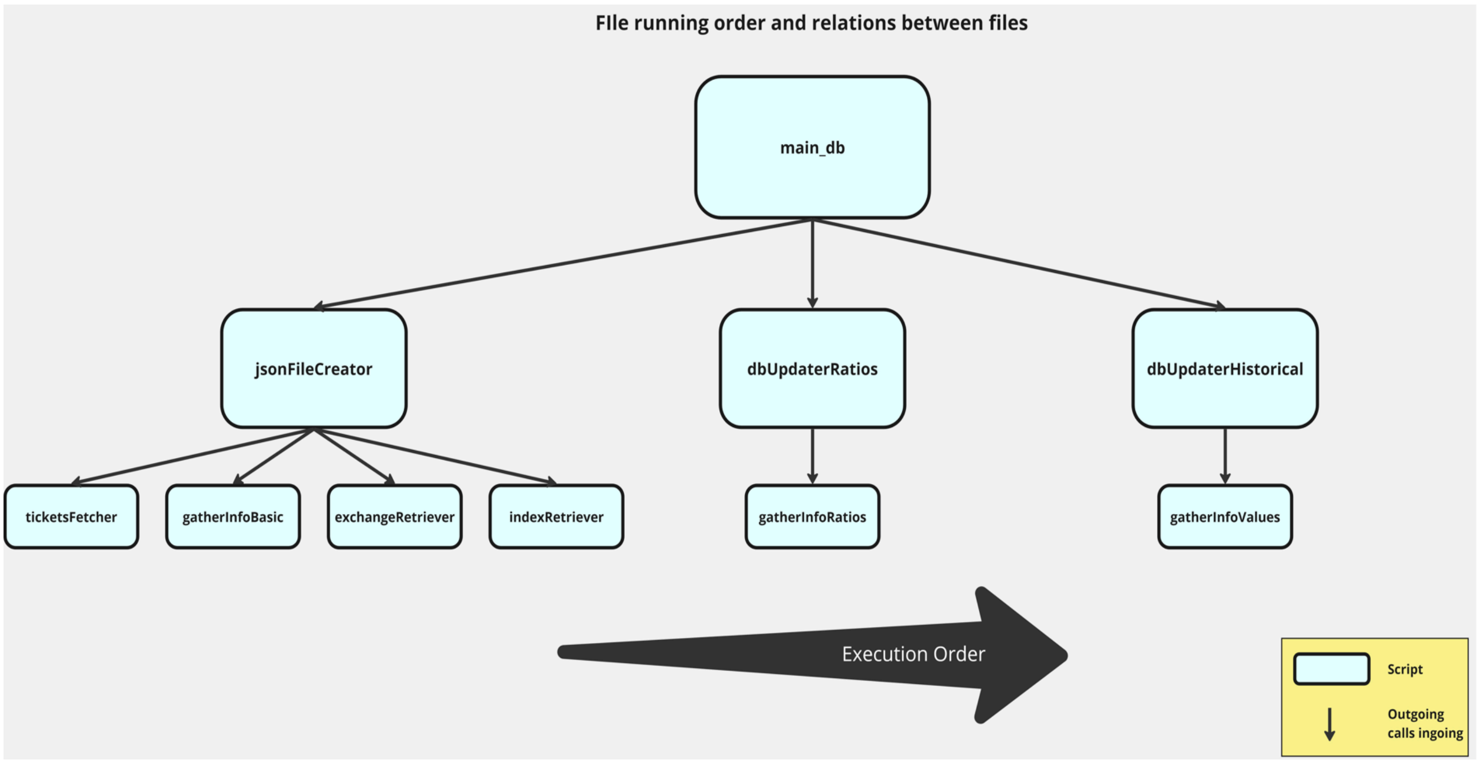
The Stock Entity - Exchange Entity relationship presents a cardinality constraint of 1:M or one to many. This finds its reason in the fact that a stock is generally enlisted in a single exchange, but an exchange enlists multiple stocks. Beyond this, the partition from both sides is total, since a stock must be traded in an exchange and an exchange must have stocks.

Finally, normalization in this database was not an aim, due to the NoSQL nature of it. Because of that, denormalization was intended to be optimized. Nevertheless, by reason of the memory constraints that the project has faced, total denormalization was not achieved.

4. Methodologies

4.1. Data Ingestion Pipeline

Once the architecture of the database has been established, the next step is to construct a pipeline to collect data and populate it. The pipeline consists of numerous python scripts that are integrated to fetch this information and correctly format it before feeding it into the database. There are three main files being executed, that require functionalities from different files: jsonFileCreator, dbUpdaterRatios and lastly, dbUpdaterHistorical. These three main files are compiled into the main_db file which, once it is executed, creates and populates the database. Before breaking down the functionalities of the scripts, it should be noted that all these are provided in the .py files, which hold, beyond the actual code, numerous comments explaining each action, as well as an execution diagram of the actual pipeline (Figure 2.0 & available in the Design folder). Finally, this section will be devoted to the exhaustive explanation of this pipeline process.



The "jsonFileCreator" script initiates the data pipeline by constructing the basic structure of the database in line with the Entity-Relationship (ER) Model and populating this basic skeleton with the qualitative, static information held in the Stock, Index and Exchange entities. This script leverages Python's "json" module and incorporates several other scripts —"tickersFetcher", "gatherInfoBasic", "exchangeRetriever", and "indexRetriever"— to compile a comprehensive dataset that serves as the material for the database. It starts by importing necessary modules and functions from other scripts within the pipeline. This includes functions for fetching tickers, gathering basic information about securities, retrieving exchange composite indexes, and gathering information about indexes. It then creates an empty dictionary named "database", which will serve as a container for all collected data. This approach is flexible and allows for the dynamic addition of data as the script runs. The "tickersFetcher.retrieveTickers()" function is called to fetch the tickers for S&P 500 shares from Wikipedia using this page API and the pandas library's function to read HTML code. The tickers serve as unique identifiers for each security and are crucial for subsequent data retrieval and organization. For each ticker fetched, the script uses "gatherInfoBasic.gather_info_basic(ticker)" to retrieve and return basic qualitative information such as the company name, sector, and exchange and save it into a dictionary 'Stocks' that will become a document containing all stocks' information.

This data is essential for understanding the context of each security within the database. The completed 'Stocks' document will now be added to the "database" dictionary, structuring it in a way that aligns with the ER Model. This structured approach facilitates efficient data retrieval and manipulation in later stages of the pipeline. This is important since this file may be updated with new fields, which would enable it to load any new desired content in the future, if the structure of the database were to change. By preparing the initial JSON file with the basic database structure and populating it with key information about the securities, exchanges, and indexes, the "jsonFileCreator" script sets the stage for more detailed data collection and updates. This script is crucial for ensuring that the database is organized, comprehensive, and ready for subsequent enrichment through the pipeline.

To gather information on indices and exchanges, the functions exchangeRetriever(security) and indexRetriever(security) are used. However, in this case the tickers for the indices and exchanges are not gathered automatically by the script, but inserted manually. This is an area of improvement that could be automated with the use of web scraping tools and libraries like beautifulSoup. From the aforementioned functions basic qualitative data is gathered about the exchanges and indices, all leveraging the Yahoo! Finance API, as done before for the collection of stocks' basic qualitative information.

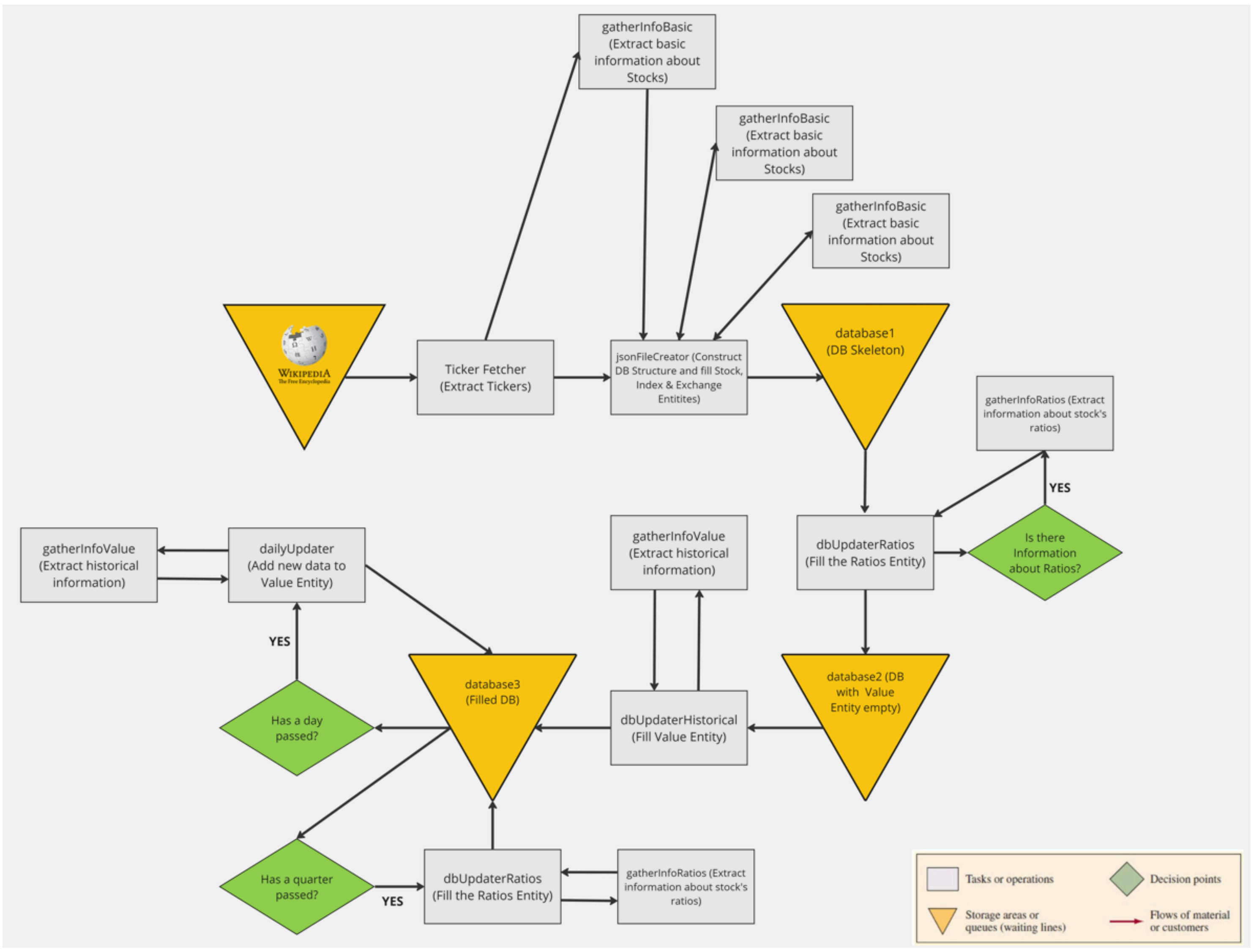
The following script is "dbUpdaterRatios". This file is designed to update the database with financial ratios for the stocks, an essential step in enriching the database with the financial insights into each company listed in the S&P 500. The script begins by importing necessary functionalities, including the "gatherInfoRatios" script for fetching financial ratios of securities and the "json" module for handling JSON data, which is the format of the database. The "gatherInfoRatios" is designed to periodically fetch the desired information from Yahoo! Finance. Consequently the "dbUpdaterRatios" script opens the newly created JSON database file for reading. This database contains the basic structure and initial data populated by the "jsonFileCreator" script, but the Value and Ratios Entities are still empty. The script will extract the keys (i.e., the tickers) from the "stocks" section of the loaded database, which will guide the retrieval of financial ratios for each stock. For each ticker, the script calls the "gatherInfoRatios.gather_info_ratios(k)" function to fetch various financial ratios. These ratios include overall risk, dividend yield, beta (a measure of volatility), PE ratio (price-to-earnings ratio), gross margin, quick ratio, and price-to-book ratio. Each of these metrics provides valuable insights into the financial health, performance, and risk profile of the companies. Additionally, a new value for the quarter attribute is added. In this way, if one wants to retrieve the dividend yield of a firm for the third quarter of 2023, one would go to the quarter attribute of the Ratios Entity, extract the index where '2023-Q3' is found and retrieve from the dividend yield attribute the observation found on that index. The fetched ratios for each stock are then appended to the respective stock's data in the database file. This process involves modifying the "stocks" document within the database, adding a new entry under each stock's "ratios" key, represented as the Stock Ratios Entity in the ER Model, with the fetched values.

Lastly, the "dbUpdaterHistorical" script is executed. This script is tasked with updating the database with historical information about the stocks to populate the Values Entity. This includes data points crucial for analysis, such as trading dates, volatility, trading volume, and closing prices. It begins by importing the "json" module for handling

JSON-formatted data and the "gatherInfoValue" script, which is responsible for extracting historical information from Yahoo! Finance. The script's main function, "dbUpdaterHistorical()", encapsulates the entire process. The script opens the database, retrieves the keys (stock identifiers) from the database and iterates over them. For each key, it further iterates over sub-keys within each stock's data, indicating a potentially nested structure designed to accommodate detailed historical data. For each stock identified by its ticker ("keySec"), the script fetches historical data by calling the "gatherInfoValue" function with the 'daily' parameter set to False, so all the possible historical information about the security is retrieved. The returned data includes trading dates, volatility (indicating price fluctuation over a specified period), trading volume (the number of shares traded), and closing prices (the final price at which a stock trades during a regular trading session). Upon successful retrieval of historical data, the script updates the corresponding stock's entry in the database with the new data. This includes adding information about trading dates, volatility, volume, and closing prices, enriching the database with a temporal dimension of stock performance.

The previously described process is the one that creates and initially populates the database. Nevertheless, one of the main aims of this work is to automate the process of updating the database. It was already mentioned that, for this, the Value Entity would have to be updated daily, since new information about securities' prices and volumes is generated everyday. Hence, the dailyUpdater function is defined to output to the database new information about the Values Entity. This function works very similarly to the dbUpdaterHistorical by leveraging the gatherInfoValues function to obtain new information about the securities. However, since the wanted information is the one of the last day, the parameter 'daily' in the gatherInfoValues function is set to True to ensure that only information about the last twenty four hours is gathered. Additionally, due to the current database implementation just holding information about closing prices, the volatility attribute is not calculated, since there is only one closing price each day for each security.

To better understand the process of constructing and updating the database through the data pipeline, a flowchart was modeled (Figure 3.0 & available in the Design folder). This diagram shows the execution of the functions with their functionalities, outputs and the underlying calling of other functions, as well as the updating of the database, which will be discussed in the following section.


gatherInfoBasic (Extract basic information about Stocks)
gatherInfoBasic (Extract basic information about Stocks)
gatherInfoBasic (Extract basic information about Stocks)
WIKIPEDIA The Free Encyclopedia
Ticker Fetcher (Extract Tickers)
jsonFileCreator (Construct DB Structure and fill Stock, Index & Exchange Entitites)
database1 (DB Skeleton)
gatherInfoRatios (Extract information about stock's ratios)
YES
gatherInfoValue (Extract historical information)
dailyUpdater (Add new data to Value Entity)
gatherInfoValue (Extract historical information)
dbUpdaterRatios (Fill the Ratios Entity)
Is there Information about Ratios?
YES
database3 (Filled DB)
database2 (DB with Value Entity empty)
Has a day passed?
dbUpdaterHistorical (Fill Value Entity)
Has a quarter passed?
YES
dbUpdaterRatios (Fill the Ratios Entity)
gatherInfoRatios (Extract information about stock's ratios)
Tasks or operations
Decision points
Storage areas or queues (waiting lines)
Flows of material or customers

4.2. Automation of the Data Ingestion Pipeline

In the world of data management and analytics, the freshness and reliability of data play pivotal roles in ensuring the accuracy of insights derived from it. The Data Ingestion Pipeline, crucial for populating our database with timely and relevant data, consists of a series of Python scripts: jsonFileCreator, dbUpdaterRatios, dbUpdaterHistorical and dailyUpdater, the first three being integrated into a master script main_db. Manually executing these scripts is not only labor-intensive, but it has a limited utility since it will populate the database with data up-to-date that will soon become obsolete. Hence, it is needed to continuously append the generated information to the already existing one. With that aim, the function dailyUpdate is created to add each day the newly generated data to the database's Values entity. The importance of this lies in the fact that market fluctuations are daily occurrences and that even a prediction with one day horizon potentially yields high benefits. Beyond this, the data about a firm's ratios also evolves, but less frequently. In general, ratios are updated each time a company releases financial results, which usually occurs quarterly. Therefore, the function dbUpdaterRatios is prepared to update the database each quarter seamlessly while saving the quarter for which the new ratios belong, as explained previously..

Nonetheless, having to open and run a file daily or having to remember to run a file every three months is unfeasible for nowadays users. So, to mitigate this issue, we propose the automation of this pipeline using Windows Task Scheduler. This approach ensures that the pipeline is executed consistently at predetermined intervals, thereby maintaining the database's accuracy and relevance without manual intervention.

The automation process involves configuring the Windows Task Scheduler to execute the dailyUpdater script daily and the dbUpdaterRatios quarterly. Task Scheduler, a component of Microsoft Windows, offers a robust framework for scheduling tasks to run automatically at predefined times or intervals. By leveraging this tool, we can ensure that our database is updated consistently with the latest data, adhering to the established update frequency. The setup begins by creating the task in the Task Scheduler. Then, under the 'Triggers' tab, we configure the task to start daily at a specified time, ensuring the pipeline's regular execution. The next step consists in selecting the script we intend to run, and lastly verify everything works.

This automation brings forth several clear advantages. Firstly, automated execution ensures that the database is updated at regular intervals, enhancing the reliability of the data stored within. Also, it reduces the manual effort required to maintain the database, allowing us to focus on more strategic tasks, and making the process more efficient. Lastly, it minimizes the likelihood of human error associated with manual execution, such as forgetting to run the scripts or executing them at incorrect times.

4.3. Basic App

Beyond the ingestion phase, it was considered to show a possible application of the collected data. For this, a basic app was built using the Kivy library from Python. This software is aimed to provide some visualizations and forecasts without the need for technical knowledge. Therefore, the application presents a simple and user-friendly graphical interface that allows the user to input a security ticker and some other information, so the corresponding data is retrieved from the JSON file and analyzed.

With this aim, instead of directly coding in Python language, a .kv file was written with the structure of the application. This file contains multiple screens, a screen manager and multiple possible pop-up windows. Firstly, at opening, the shown screen is the initial menu of the application where, besides our original logo, the three functionalities of the application are available as buttons. The first one of these is for plotting the closing prices of a desired security during a specified time span. After selecting this functionality, a screen is presented with labels reading basic instructions about what the user is expected to enter and input text boxes. These inputs are error-proofed by some mechanisms that will be explained later. After the user inserts the desired stock, the timespan and clicks the submit button, a function is executed and, if everything is correct, the user is shown the following screen with the plot in it. In this screen there is also a return button so, if the user desires to plot another security, turning back is a convenient option.

The submit function serves the main logic behind the first functionality. This function first checks with string comprehension techniques for the patterns in the inputted text. The expected input for the ticker has a shape ‘MMMM’, ‘MMM’, ‘MMM.M’, ‘^MMMM’ or ‘^MMM’, where M is any letter, and the one for the timespan has a shape ‘yyyy-mm-dd,yyyy-mm-dd’, where m is any number so that (1<=m<=12) and d is one so that (1<=d<=31). If any of the two inputs do not match the corresponding possible shapes, an error is returned in the form of a pop-up window and the submit function returns an empty variable that, once checked, does not allow for the second screen to be shown, so the user stays in the first one. If this does not happen, both inputted texts are sent to a plotter function defined in another file. This function is in charge of the logic behind retrieving the data and plotting it. With that objective, it first reads the json file and saves all tickers so it can be checked if the inputted security is a Stock, an Index or an Exchange. If it is None, it returns an error that will be checked by the submit function and it will cause a pop-up window to be shown and the submit function to return None. If the ticker is found in one of the three possible entities, a variable category is set to that entity's name. After this, the logic of the dates is checked by, first, ensuring that they were inserted in chronological order. Subsequently, the available dates of data for the security are scanned looking for the desired dates. If they are not found, it is assumed that the user might have inserted a holiday or weekend day, when data is not available, so it is checked if there is any data point at least three days away from the one desired by the user. If this is not the case, an error is returned; but if it is the case, the date is changed to the closest point that is no more than three days apart. Finally, lists containing the desired days of the timespan and their corresponding values are created and plotted. This plot is saved as a PNG image, so the app can retrieve it.

Beyond this, in the initial menu, there is also the button for selecting the second functionality. This one accepts from the user a ticker and a horizon to then calculate a prediction for that stock during the horizon, in days. For this, the screen utilizes its own submit function, which is highly similar to the previously explained one, so it checks that the imputed ticker follows the expected patterns and, also, checks that the imputed horizon is a non-zero integer. Were a negative integer to be imputed, the processing functionalities will be carried out by turning the integer into positive. However, any input that deviates from the previous specifications will return an error in the form of pop up windows, which are the same ones that were utilized for the first functionality.

After the inputs to the function are checked, a function from another Python file is called to make and plot the prediction. For this two models were assessed: an automatically selected ARIMA model and a Prophet model. For the reader who does not know how ARIMA models work, basically they require certain very important hyperparameters that will determine the efficacy of the predictions. These can be obtained through analysis of the series to be modeled, but this analysis is not possible to do if one wants to automate the process. Hence an auto-ARIMA model was preferred, which would not perform the analysis, but would determine the hyperparameters in terms of AIC. Nevertheless, this model was not the one implemented finally, because, for non-seasonal series, auto-ARIMA returns a linear prediction. Consequently, Prophet was preferred, since it automatically breaks the trend of the series into a piecewise function.

Finally, the third functionality of the application is the simplest one: it accepts from the user a ticker and a timespan and shows on the screen the percentual return of the security during that time span. Hence, it is highly based on the first functionality, which has the same inputs, and all its error-proving properties are rooted in the ones of the first functionality.

The application, however, has a great drawback: since its functionalities are currently based on JSON files, it has to load into memory the entirety of the file to retrieve information from it. This slows down the different processes making them have a latency of around half a minute, which is intolerable for a user-friendly application. Nevertheless, this drawback might find its solution in the further development of the application to utilize HDFS and MapReduce. In this way, the JSON database could be broken into blocks and distributed across a cluster of machines. Afterwards, through MapReduce, the different functionalities of the application could be parallelized to make them more efficient and reduce their latency.

This is just the main area of improvement for the application's current minimum viable product (MVP). However, more can be done to improve our data intensive application. Among these other options, the most relevant one would be to add further functionalities that would help our users understand better the data they have. These could include further plots, more variables to be plotted, or, even, a more friendly and aesthetic design. Also, functionalities more directly related to portfolio optimization, which would deal with the integration of the app with other portfolio optimization softwares, as further sections discuss.

Additionally, since Kivy is a library thought to create both computer and smartphone applications, creating a version of the application to work on smartphones is a plausible scenario. For this, however, the main bottleneck is the current need of the user to have the data locally on their machines. Hence, the use of cloud services could enlarge both the variety of devices on which the application could be used and the reliability of the data access and security.

4.4. Parallelization & Distributed File System

In the previous section it was shown that the volume of data handled both by the pipeline and the application is the main bottleneck that affects its latency. Currently, the application takes around half a minute to carry out any of its three functionalities. This high latency is mainly found in the processing time, as previously mentioned, and can be improved by changes in the storage and the computation system.

Parallelization is the main tool to be used when handling and processing data of large volumes. In consequence, the best option now appears to be to implement a HDFS cluster, as stated in the application section. This file system would allow splitting the current database into different blocks, which would be stored in separate nodes. To ensure fault tolerance, redundancy would be also implemented in this splitting, so recovery time in case of failure is faster and, in general, downtime is minimized.

To achieve this, since the best practice is to work on Linux and use Ubuntu as the Linux distribution, a Virtual Machine has to be installed. In this case, VirtualBox from Oracle was chosen and an environment was created with 4GB of memory and 20GB of disk space. Afterwards, Apache Hadoop has to be installed, for which Java has to be installed too.

Once the Virtual Machine is prepared, Hadoop installed and the environmental variables are set, the data directories have to be created and the distributed file system started. Finally, the JSON database can be moved to the HDFS cluster and operations can be performed in it. For this, additionally, other Apache softwares like Hive or Pig could be used to implement SQL-like queries in our JSON DB.

It should be understood that the potential to improve the latency of the application based on parallelization protocols is not unlimited nor perfect. The constraint of the parallelization speed-up, as stated by Amdahl's law, is based on the quantity of tasks that must be performed sequentially. In the application, however, there are many parallelizable tasks, the main one being the retrieval and checking of information. For example, when using functionality one or two of the application, the user inputs a ticker, which might be intercepted immediately by the application as an invalid one due to its format or it might pass on. If it passes, the software will have to search over the database to find the corresponding data to that ticker or to observe if it is not present. This is a very inefficient task that could be highly parallelized. In the same way, the application could parallelize the retrieval of the specific dates to be considered for the first functionality. For these two examples, implementing MapReduce would be very beneficial.

Nonetheless, not all tasks enjoy such a high degree of potential parallelization. Since the data with which the application deals is time series data, almost all analysis on it must be sequential. To exemplify this one can observe the process of predicting the prices of a security for the next 100 days: the used model, Prophet, requires to know all the previous prices up to the following prediction; hence if one wants to predict day 100, first one must have predicted day 99 and all previous, and for day 99 one must have predicted day 98 and all previous… Definitively, this is not a parallelizable task.

Another bottleneck for parallelization would be the fitting of the Prophet model, that is developed using Stan modeling language, which does not inherently support parallelization. However, some of the hyperparameter tuning of the model could be parallelized. Currently the application uses an 'Auto-Prophet', which automatically searches and selects the best hyperparameters for the model. The main one of these is the knots of the series. Since we are dealing with non-linear data, Prophet fits a piecewise linear regression on it to adapt to non-linearity. For that many knots are tried and the best are selected based on different goodness of fit metrics, like AICc and general residuals. Nevertheless, Prophet does not parallelize this process of trying different knots, which significantly increases the latency of the application's third functionality. However, one could implement a parallelized grid search of knots using concurrent threading. This would be done by defining a set of potential knots and fitting one Prophet model on each combination of these, then saving the results and comparing. All of this could be done in parallel, significantly reducing the latency of the application. But, still, this raises the challenge of how to define the best set of potential knots to be tried.

Finally, it must be restated that there is a high amount of value to be gained from the parallelization of the process. This is mainly because the main bottleneck is the retrieval of information, not the analysis in itself. Hence, the application could enjoy significantly lower latency and the user experience could be notably boosted.

5. Potential use cases

Throughout the entire report, it has been uncovered that the value created through our constant data retrieval process is highly considered by players in the financial market. Actually, the value generated by our data extraction can be quantitatively measured by benchmarking competitors offering a standardized service of real-time financial information such as Bloomberg Terminal or Refinitiv, in terms of pricing. In the case of the first instance, which is the main player globally, it offers a subscription model based on an annual cost of around $27,660 per subscription, which although the range of products and services offered is wide, the essential service is the provision of real-time data to users. Definitely, the information and practicability of our project purpose has a huge potential if scale and real-time updates are achieved, which would be the most challenging functionalities of the software. Therefore, it is essential to consider research on possible solutions and use cases to which our automatized data ingestion pipeline and simple but scalable graphs generator might be applied.

The first case to deal with will be an investment portfolio optimizer, which was our initial motivation and functional idea. A portfolio optimizer is a tool which provides the user with guidance by inputting infinite diverse constraints including a desired return and the assumption of a specific risk. The algorithm would offer an optimal portfolio recommendation based on economic indicators and based on the user inputs. In order to accomplish this purpose, the portfolio optimization theory emphasizes the need for the highest precision and quality of the stocks and economics ratios to provide the best selection of assets with a proper risk adjustment offering the highest expected return. All this comprehensive data coverage in real time is extracted with our process and the preprocessing is perfectly matched by making a feature engineering process through the ratios computation is done. It can be adapted to include all data needed in order to consider more types of investments and make the optimizer algorithm more complex to customizable inputs. In addition, Monte Carlo simulations could be run based on the historical updated data in order to test the proposed portfolio growth forecasted and survival, providing more reliable results.

The second use case is related to the previous potential solution, since it can be a complement to a portfolio optimizer. It consists in the creation of an investment strategy backtesting process, which is essential for funds managers and investors in order to evaluate their previous performance. It would leverage a data extraction process similar to ours, ensuring the extraction of numerous market cycles. Afterwards, numerous trades would be simulated according to the strategy proposed based on this historical data, generating diverse hypothetical ratios so a tracking of the performance can be accomplished. Some backtesting tools available in the market include QuantConnect, Quantopian, Backtrader and MetaTrader.

The third case will cover financial modeling. Financial modeling is a laborious task which requires a deep expertise and, of course, the presence of reliable and continuously updated data, that is crucial to ensure consistency and accuracy from the first step. The data required might mean an extension of our current retrieval process in terms of information, since it needs exhaustive data regarding all revenues and expenses coming from the income statements of the companies, but this information could be obtain through the use of web scraping tools or LLMs that could read, interpret and return to the script the desired values from the financial statements of companies. Hence, it would not mean a significant concern for our scalable process. Moreover, once the data is obtained, there are two main ways of building financial models adopted in the sector: using excel spreadsheets or specific financial modeling softwares. Focusing on the second one, since it offers numerous advantages over spreadsheets, the main reason why a financial modeling software is more useful is mainly because of the automatization of the model, including the data integration and reducing substantially the need of data entry, reducing the possible errors and increasing the efficiency of the whole process. On the other hand, our data ingestion pipeline could be seamlessly integrated to the software, offering the ability to keep every model up-to-date and making possible an extensive and accurate interpretation of the model. As a consequence of this data integration, the capacity to create a sensitive analysis varying numerous inputs and assumptions, assessing and valuing all diverse hypothetical scenarios, providing a comprehensive view to all stakeholders surrounding the investment decision-making process. The development of such a complicated software is relevant, but the potential it has is definitely important to consider. Some of the main financial modeling tools available in the market are Cube and Finmark.

The following use case to treat is algorithmic trading, which involves the combination of three main topics: algorithm execution, black-box algorithms and high-frequency trading (HFT) algorithms. Basically, algorithmic trading involves the use of computational formulas to execute the trades, such as arrival price algorithms, basket algorithms or percentage of volume algorithms among many other options. On the other hand, it exists other kind of algorithms that are surging given the recent application of artificial intelligence to the field and the main difference with the previous ones is their lack of explainability, due to the fact that they are not following any execution rules previously defined as it is the case of the algorithms previously mentioned. These called black-box algorithms are opaque even for the engineers designing them because understanding how decisions were taken means a significant challenge. The point with this kind of algorithms that might present extraordinary outcomes is that they are widely used for high frequency trading given their outperforming over other basic algorithms. And this is the point in which our automated data pipeline adds value. Algorithmic trading profits are based on taking decisions fast, it is a question of milliseconds, so having all data continuously updated is extremely important to get powerful results, which can be done with our data extraction. This kind of operation leverages heavily on this data extraction process, which allows the algorithms to execute trades faster than humans, avoiding manual errors and human sentiment concerns. Of course, the implementation of our second use case, a backtesting process, is highly enriching since it would help to test diverse scenarios in order to train the algorithms and fine tune its parameters, increasing their effectiveness and minimizing potential losses.

Finally, an extra use case to mention corresponds to the optimal utilization of this data for the creation of visualization tools. As data analytics is gaining a considerable importance in all the sectors but especially in the financial one, the real time fetching of data is useful for the generation of diverse types of graphs and charts which might help the decision making process for investments or management process of the companies. The creation of a general dashboard is a good option to visualize the evolution and prospects of data. As shown above, the development of a basic graphs application is relatively simple and its potential upsides are highly significant and the integration with the previous use cases is possible and highly beneficial.

6. Data Governance

Investment decisions have powerful, real life implications. That is why it is necessary to ensure that financial data is accurate and consistent. Additionally, customer data gathered through our application must be kept private, secure and its treatment compliant with regulations. Data governance provides an answer to these needs.

Specifically, there are four main objectives that are relevant to our application: data quality, data security, compliance, and data usage. Data quality is central to the project, as it ensures accuracy, completeness, and timeliness that are necessary when staying informed about the financial markets. Given the project's reliance on real-time financial data to optimize investment portfolios, it is essential that the data extracted through the pipeline remains reliable and valid for financial analysis. As for data security and compliance, it must incorporate advanced security measures such as encryption and secure data handling protocols to safeguard its information. Lastly, for data usage, establishing clear guidelines on how the data can be used within the analytics tools and who has access to this data is crucial. These policies help prevent misuse of the data and ensure that all stakeholders, such as team members as well as customers, understand their roles and responsibilities in data handling. We will now analyze specific implementation methodologies in more depth.

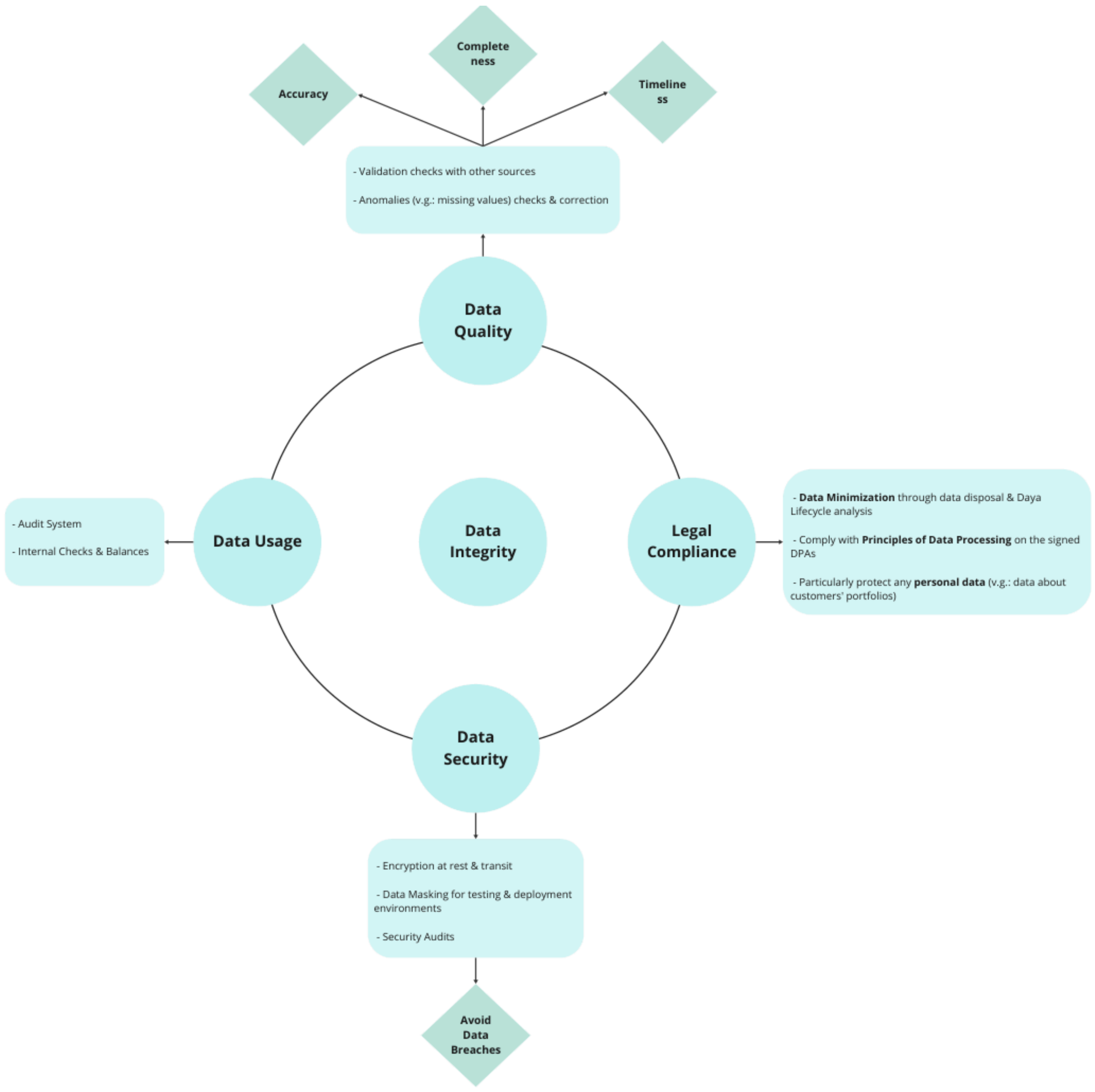
Our governance framework is designed to maintain the integrity and security of the financial data through a well-defined set of practices. Central to this framework is the role of data stewardship, where a designated data steward is accountable for overseeing the accuracy and integrity of data across different segments of the pipeline. Stewardship is relevant from the initial data collection to the final storage of data in a Hadoop Distributed File System (HDFS). Their responsibilities include not only the maintenance of data quality throughout its lifecycle but also decision making to correct the data, and update it, when necessary. To safeguard sensitive financial data, the project implements data access controls within the database stored in the HDFS cluster. This includes robust authentication mechanisms and role-based access controls. Such measures ensure that only authorized members of the team are able to access critical customer data, minimizing the risk of unauthorized data exposure and protecting privacy to retain customer trust.

Moreover, the framework would include an audit trail system that meticulously records all data access and modifications. This system is used to maintain internal checks and balances, and it is equally crucial for external audits and compliance verifications. Given the project's reliance on automated scripts for data updates, it is important that each script execution is traceable. This way, operational transparency is ensured, and there is a method for monitoring and reviewing the effectiveness of data governance practices. Lastly, the data governance framework demands regular reviews and updates. Thus, it is able to quickly adapt to emergencies or technological advancements. This adaptive approach involves periodic evaluations of data handling and security procedures to address potential vulnerabilities and to integrate new security technologies as they become available. Continual improvement ensures that the governance framework remains robust and capable of supporting the project's ongoing and future needs.

Once a comprehensive data governance framework is established, it is possible to be more precise in our data quality management methods. Given that the project relies on data sourced from Yahoo Finance, which may not always be totally reliable, there is a need for data validation. Automated validation checks may be implemented as part of the data ingestion pipeline to verify data accuracy as soon as it is fetched. This involves checking for anomalies, inconsistencies, and missing values that could affect data reliability. To do so, the application might compare it to other sources, or see whether it is consistent with previous trends and formats. Furthermore, regular data cleansing routines are crucial for the project. These routines help in correcting detectable errors, removing duplicates, and filling missing values. To objectively assess and monitor the quality of data managed within the system, the project employs three quality metrics: accuracy, completeness, and timeliness. These metrics are continuously monitored, providing the project team with real-time feedback on the state of the data.

Data security is another critical aspect of the project, especially considering the sensitivity of the customer data involved, which might provide insight into investment decisions. To protect this data against unauthorized access and potential breaches, the framework incorporates several robust security measures. Encryption of data at rest and in transit represents a foundational security practice, ensuring that all data stored in the HDFS database and transferred across networks is encrypted using strong encryption standards. This prevents potential interceptors from accessing readable data. Regular security audits and vulnerability assessments form another pillar of the project's security protocols. These audits help identify and mitigate risks associated with data breaches, unauthorized access, and other cyber threats. Additionally, data masking techniques are employed when handling data in development and testing environments. This practice ensures that actual data cannot be exposed even if non-production environments are compromised. Obscuring sensitive information provides an additional layer of security, which is useful during the development phases of the project's analytical tools.

Regarding data disposal, the project currently relies on manual reviews to identify and remove outdated or irrelevant data, adhering to compliance and data protection regulations. However, to enhance this aspect, the project could benefit from implementing automated data lifecycle management policies. These policies would use criteria such as data age, relevance, and compliance requirements to automate the purging process. For instance, implementing a system that tags data based on its entry date and a predefined retention schedule could automate the deletion process, ensuring data is not held beyond its useful or legal lifespan. This method would decrease storage costs by eliminating unnecessary data retention. It is important, however, to be able to accurately identify which data is outdated, since often investors base their decisions on historical, rather than current data insights. Furthermore, incorporating more advanced data anonymization techniques before disposal could add an extra layer of security, ensuring that disposed data cannot be traced back to individuals. This practice would be particularly beneficial in adhering to the principles of data minimization and privacy by design, as stipulated by data protection laws such as the GDPR.


Accuracy
Complete ness
Timeline ss
- Validation checks with other sources
- Anomalies (v.g.: missing values) checks & correction
Data Quality
Data Usage
Data Integrity
Legal Compliance
- Audit System
- Internal Checks & Balances
- **Data Minimization** through data disposal & Daya Lifecycle analysis
- Comply with **Principles of Data Processing** on the signed DPAs
- Particularly protect any **personal data** (v.g.: data about customers' portfolios)
Data Security
- Encryption at rest & transit
- Data Masking for testing & deployment environments
- Security Audits
Avoid Data Breaches

Looking forward, our data governance framework is well positioned to evolve with technological advancements and regulatory changes. Currently, the project operates at a proactive stage of governance maturity, where processes and policies are established, but there is room for improvement before the effective stage is reached. Particularly, in areas such as automated compliance checks and advanced data analytics integration. As the project matures, it could move towards a more automated approach to data governance. It is important that team members recognize data governance as a core layer of using financial analysis to power decision-making.

6.1. GDPR Compliance

Firms that collect any sort of data, specifically personal data, need to ensure that they abide by the laws and regulations of that region or country where the data is stored or whose customers are being served. In modern times, since data plays an ever more valuable role in economies as well as on individual firms' bottom lines, most firms and applications usually collect at least some user data. Thus, given its increasing importance, more and more laws are enacted which makes it difficult for small firms with small budgets to stay updated. Therefore, knowing that data regulations will become increasingly demanding, the developers of data-intensive applications need to take that into account at the very early stages of app development. Optfolio, too, will ensure that personal data that can be linked to user's portfolios complies with the GDPR regulations by following some of the steps outlined in the proceeding paragraphs.

Since Optfolio would store its data in Europe and serve European customers, GDPR is seen as one of the more significant legal frameworks that have to be taken into account when dealing with personal data. General Data Protection Regulation (GDPR), enacted in 2016 in the EU area, mainly deals with personal data and its processing, its subjects, controllers, as well as processors, and requires that a Data Processing Agreement is signed with any third-party processors. It focuses on five main pillars of data privacy: Data Controllers, Compliance for US companies, Data Privacy, Consent, and Services.

To begin with the first pillar of GDPR, Data Controllers, it is of utmost importance that it is clearly defined who are the true data controllers. In our case, it would be Optfolio, the company that operates the application, who would be considered as the main data controller, therefore taking up the responsibility of determining the purposes of processing the data. Moreover, not only we must ensure that it is us who comply with the data processing standards of GDPR, but also that any third-parties comply with it too, in which case clear terms & agreements must be signed by us and the corresponding third-parties, which would be mainly cloud servers providers to whom Optfolio would outsource the storing part of the data processing. Finally, as Optfolio would grow and expand its reach and budget, it should also select certain individuals or teams who would oversee the data control and various other GDPR requirements, given its always changing any dynamic nature.

Second, with regards to data privacy & security, a great deal of it has been covered in the previous section (see Section 7: Data Governance). The only addition that would have to be made to respect the GDPR regulations would involve creating strong access controls and other authentication mechanisms. This would help to ensure that access is granted solely to authorized parties, and that no one without the access rights can not access any sensitive information. All in all, in combination with the factors mentioned in the previous section, our app would grant data confidentiality, integrity, and availability, and therefore would be compliant with the second pillar of the GDPR.

Third, Optfolio must gain explicit consent from the users for any data collection procedures. The agreement must be written in clear and plain language, it must outline the specific purposes, it should make the user aware of any possible risks, rules, safeguards, and rights, as well as it should ensure appropriate security and confidentiality of the personal data. Of course, the agreement must cover and include some other important notes but those are outside the scope of this report. It must also be mentioned that users should have the ability to withdraw their consent when it is deemed necessary, and the procedure to do so should be simple and not time consuming.

Finally, the Services pillar of the GDPR must also be covered. This, naturally, includes the fact that the data processing is fair, transparent, and lawful, and that all users have the right to access the data that is stored, request corrections or deletions, as well as request information about the different ways this personal data is being used. Additionally, Optfolio should implement data minimization practices and only collect and process data that is absolutely necessary to provide its services, and once it has fulfilled the purposes for which the data was collected, it should be deleted or anonymized.

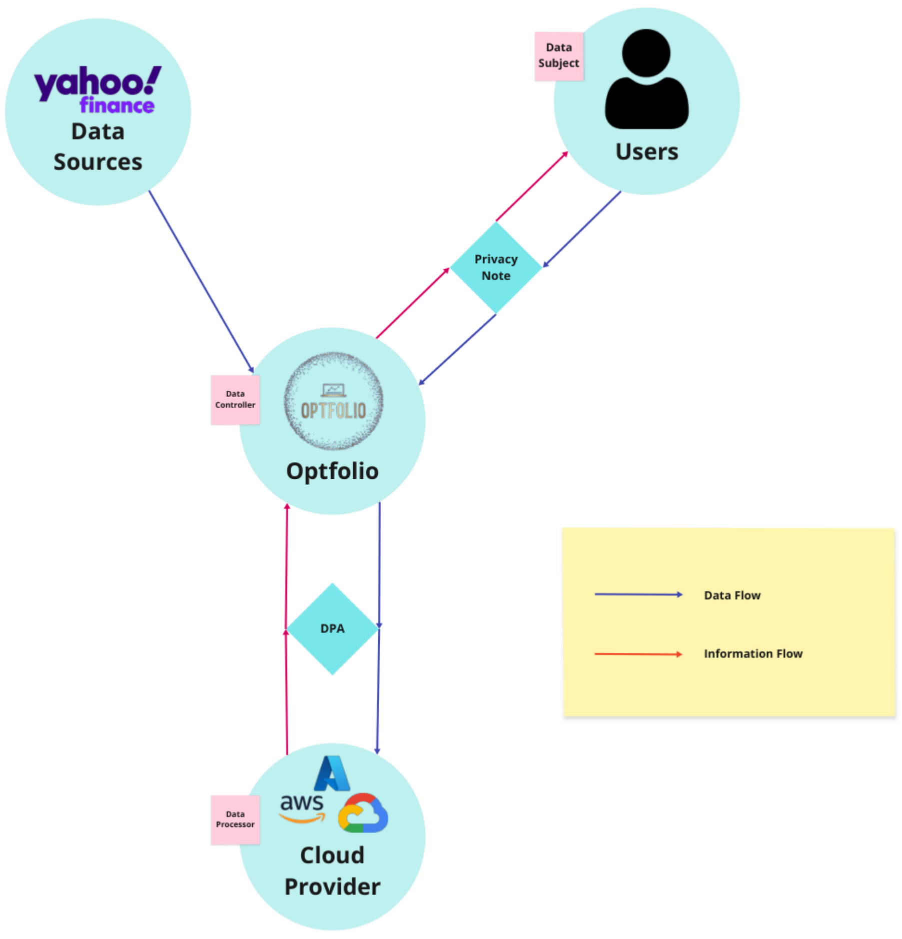
The following is an example of a Privacy Notice that could be further developed to become the one a client would sign to give Optfolio the capacity to process their data. This agreement follows the main points of the Principles of Data Processing, GDPR's 39th Recital:

> "*We strive to give all our users the best possible optimization on their investments at an adequate price*[1].
>
> *This purpose, however, involves a strong personal component. We want to give you the best portfolio in accordance with who you are*[2]*. Hence, we need to collect data to know you, your needs, resources and ambitions better*[2]*. Among the data we need to collect there is information about your income, tax rate, annual expendings, desired returns, etc*[3]*. As you may observe, we only require financial data about you and your situation, but nothing else, since we limit ourselves to what is required*[3]*. Additionally, all data will be disposed of after a certain period to ensure that information about yourself held by us is minimized*[4]*. Furthermore, we want to clarify that this information is not used for any other purpose than for tailoring your portfolio*[3].
>
> *To achieve the excellence of service we aim for, in many cases we need to process the data with the help of a third party. These are highly respected companies that will only provide us the capacity to store your data. Despite this sounding worse, it is actually needed because we cannot scale to give you a fairly priced service without the help of these storage providers*[5]*. Additionally, they are more experienced than us in data security, so your data is better safeguarded on their data centers*[6].

Please, click the 'I accept' button if you agree to allow us to collect these personal characteristics. Be assured that you can later change your mind[7]."

1. This sentence expresses the lawfulness and fairness of the process. It also shows the overall purpose of the data collection.
2. These sentences further delve into the specific purpose of the data collection and explain why the purposes could not be fulfilled by any other means.
3. These sentences make clear that data collected is limited to what is necessary and adequate for the previously stated purpose.
4. This sentence shows that data collected is time-limited.
5. This sentence explains the need for this kind of data processing and why there are no better alternatives.
6. This sentence shows some of the security and safeguard benefits or measures.
7. This sentence states to the customers their rights to withdraw data from the collection process.

Additionally, through the text only clear and plain language was used.



7. Portfolio Optimizer Design and Implementation

We have created a tool which allows the end user to have access to quite reliable predictions of individual stocks, almost instantaneously. This has added them a lot of value, but gives us a lot of space for potential development. What if we did not limit ourselves to these predictions, but take advantage of this potential to create even more value for them?

Here is where the idea of portfolio optimization comes in. Portfolio optimization is one of the most important features that a finance related project should have implemented. The concept consists in adjusting a portfolio to maintain the level of risk that a user is willing to carry, while yielding the maximum possible returns. In this way, creating a balanced portfolio, to spread the investment capital across several asset classes, in a more "correct" way. This is what investors call creating an "efficient portfolio", because you are generating the highest possible return at an established risk tolerance level.

This tool could add a lot of value since, apart from helping users to adjust their portfolios depending on the forecasts, they would have the possibility to take other measures of the fitness of their investments into consideration, directly from our platform.

It would consist in the user adding a set of stocks that they are interested in or have in their portfolio, and the system would set a percentage for each of the different investments, taking into consideration the ratios and data of the different companies. Different stocks have different volatility, which refers to the likelihood an asset's price will change significantly. Users might use this feature to adjust their investment strategies in real-time, accomodating for shifts in market conditions. Integrating a system like this one, it could dynamically align investment allocation, ensuring that the portfolio maintains the optimal allocation.

Furthermore, the application could incorporate a learning algorithm that adapts to the user's investment behavior over time. This approach could add the option of giving suggestions based on the user's past decisions and apparent risk tolerance. It would effectively create a tailored investing experience that could finally make the tool an indispensable part of the user's financial planning process.

7.1. Application Functionality for Portfolio Optimization

7.1. 1. Data Input Specifications

In order to adjust properly to the user needs and demands to provide a proper allocation of assets, it is required to ask for the proper information to assess the task. After benchmarking among other portfolio optimizers available in the market, we came across diverse input frames that might be useful to make an adequate portfolio optimization.

- Portfolio Type: It would allow you to choose how to construct its portfolio afterwards to be analyzed and optimized. It might be:
    - Asset Classes: inserting the type of asset in the asset allocation part, ranging from type of equity by market capitalization to fixed income and other derivative products.
    - Tickers: inserting directly the Stocks ticket in the asset allocation part. At the moment, we are only retrieving S&P 500 stocks data so only this option would be available.
- Time Period: it serves to specify the time period. The two options available would be:
    - Month-to-month.
    - Year-to-year.
- Start Year: Year of beginning of the portfolio optimization period.
- End Year: End year for the portfolio optimization period.
- Optimization Goal: defines what is the actual performance aim of the user depending on his preference.
    - Mean Variance - Maximize Sharpe Ratio: By measuring the risk-adjusted return, the maximization of the sharpe ratio tries to get the highest return for a given risk.
    - Mean Variance - Minimize Volatility subject to a given targeted annual return: given a target return, it would balance the portfolio to minimize the volatility of the returns.
    - Mean Variance - Maximize Return subject to a given targeted annual volatility: given a targeted annual volatility, it aims to maximize returns.

- Mean Variance - Minimize Variance: aims to build a predictable and stable portfolio by reducing the dispersion of the returns around the mean return of the portfolio.

The Conditional Value at Risk (CVaR) quantifies the average value of the worst case scenario given a level of confidence.

- CVaR - Minimize Conditional Value-at-Risk: Minimizes the potential losses due to difficult market conditions.
- CVaR - Minimize Conditional Value-at-Risk subject to a targeted annual return: Similar to the previous option but constraining to a desired level of return.
- CVaR - Maximize Return subject to a targeted monthly CVaR: given a monthly CVaR, it would try to maximize the return.
- Risk Parity: it would allocate assets according to their risk level addition to the portfolio instead of its expected return, with the objective of making a similar risk contribution of each asset to the portfolio in order to balance it.

Tracking Error is the divergence between the price of the asset and the price of the benchmark index, which as known, in our case, at the moment is the S&P 500.

- Tracking Error - Minimize Tracking Error: it minimizes the error to perform similarly to the benchmark as possible.
- Tracking Error - Maximize Information Ratio: given a level of tracking error, its objective is to get higher returns compared to the benchmark.

The Kelly Criterion is a formula used to get the best size of the position in the portfolio in order to improve the long term performance rate.

- Maximize Kelly Criterion: with this option, positions would be readjusted to improve the returns over time.

- Asset Constraints: This option would give the user the possibility of including a minimum weight and a maximum weight in the asset allocation section.
    - Yes.
    - No.

- Benchmark: to what the performance of the portfolio would be compared to. As known, the most sensible option would be to benchmark with the S&P 500 index but more options can be available, such as benchmarking with a specific stock.
    - INDEXSP: S&P 500, index out stocks data is available at the moment.
    - Ticker: specify a ticket stock to benchmark.

- Asset Allocation: this section is crucial for the user to input the positions that set up the portfolio. It would be required to input to complete a whole record:
    - Ticker symbol: stock ticker of the position in the portfolio.
    - Allocation: current size of the position in percentage.
    - Minimum Weight (If Asset Constraint Input was Yes): minimum weight of the position in the output portfolio recommendation.
    - Maximum Weight (If Asset Constraint Input was Yes): maximum weight of the position in the output portfolio recommendation.

7.1.2. Output

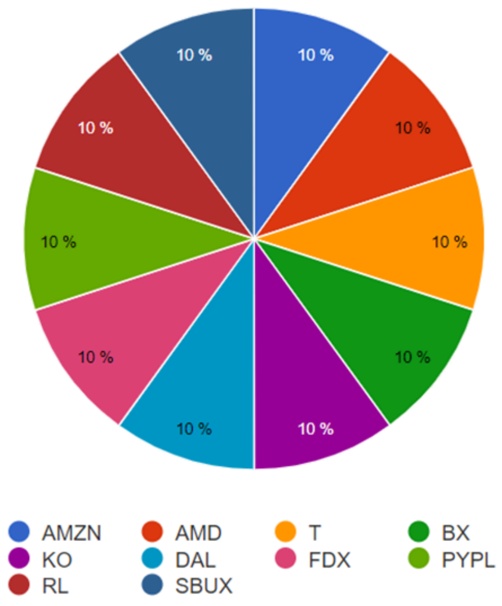
The desired result of these imputations previously specified would be a rebalanced portfolio graph with the weights specifications according to the optimization goal. In addition, to ensure some kind of reliability on financial terms, a security 95% range of potential performance would be computed to assess robustness depending on the market conditions.

7.2. Use cases

7.2.1. Scenario-Based Optimization

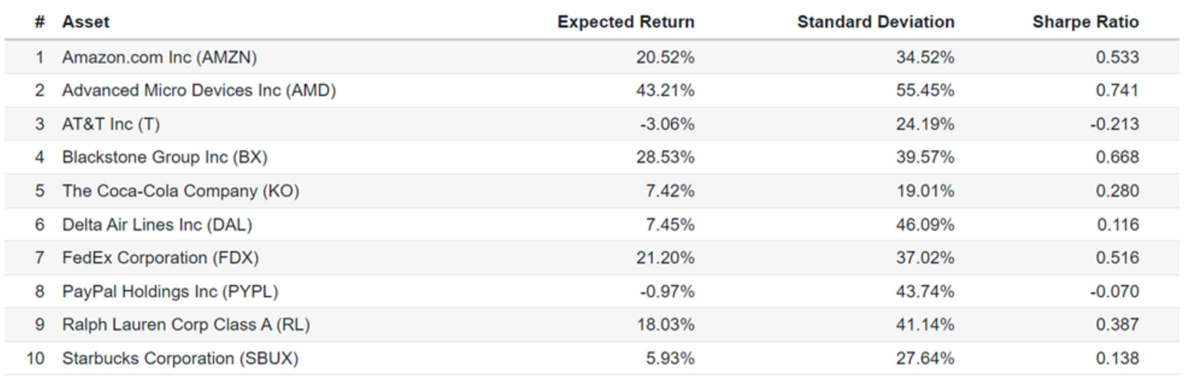
To understand deeply how this portfolio optimizer works, we are going to retrieve different examples applying a kind of optimization to a specific portfolio. Let's consider the following initial portfolio created in 2020 with 10 different stocks of different sectors weighted equally all of them, 10% each one:

- AMAZON ("AMZN").
- Advanced Micro Devices Inc. ("AMD").
- AT&T Inc. ("T").
- Blackstone ("BX").
- Coca-Cola Co. ("KO").
- Delta Air Lines Inc. ("DAL").
- Fedex Corp. ("FDX").
- Paypal Inc. ("PYPL").
- Ralph Lauren Corp. ("RL").
- Starbucks Corp. ("SBUX")



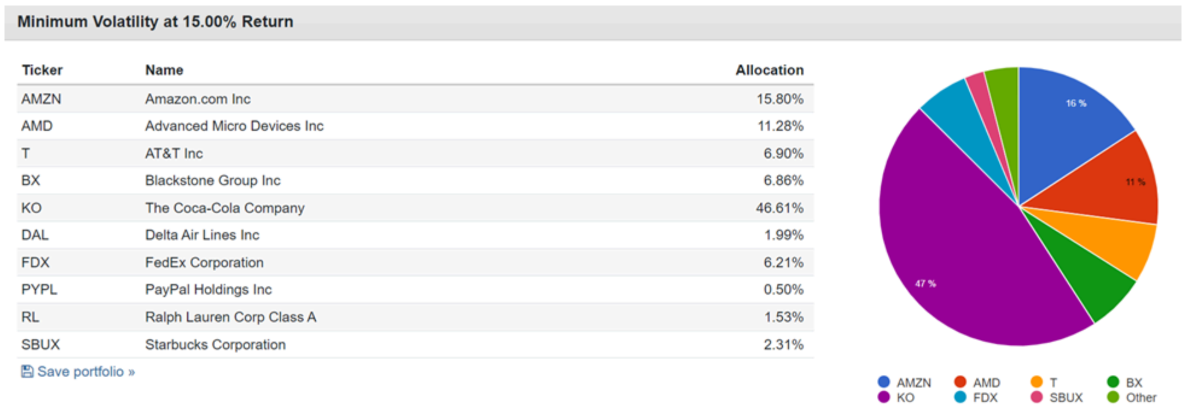
So this will be the input portfolio which we would have to choose the goal of the optimization. For this specific case, we want for instance to minimize the volatility subject to a targeted annual return of 15% using the mean variance method. In addition, we will use a robust optimization by applying the Monte Carlo method in order to resample the efficient frontier inputs.

With these inputs, different predictions are performed with the data extracted including prices and numerous ratios on each individual stock and the function would be applied depending on the specific optimization goal.



#	Asset	Expected Return	Standard Deviation	Sharpe Ratio
1	Amazon.com Inc (AMZN)	20.52%	34.52%	0.533
2	Advanced Micro Devices Inc (AMD)	43.21%	55.45%	0.741
3	AT&T Inc (T)	-3.06%	24.19%	-0.213
4	Blackstone Group Inc (BX)	28.53%	39.57%	0.668
5	The Coca-Cola Company (KO)	7.42%	19.01%	0.280
6	Delta Air Lines Inc (DAL)	7.45%	46.09%	0.116
7	FedEx Corporation (FDX)	21.20%	37.02%	0.516
8	PayPal Holdings Inc (PYPL)	-0.97%	43.74%	-0.070
9	Ralph Lauren Corp Class A (RL)	18.03%	41.14%	0.387
10	Starbucks Corporation (SBUX)	5.93%	27.64%	0.138

As a result, the optimization output offers a range of expected annual returns of 7.66% to 35.43%. Consequently, the output would construct a portfolio with the weights updated to minimize volatility at 15% annual expected return, being Amazon and AMD the stocks with higher allocation, with 15.80% and 11.28% respectively.



Minimum Volatility at 15.00% Return

Ticker	Name	Allocation
AMZN	Amazon.com Inc	15.80%
AMD	Advanced Micro Devices Inc	11.28%
T	AT&T Inc	6.90%
BX	Blackstone Group Inc	6.86%
KO	The Coca-Cola Company	46.61%
DAL	Delta Air Lines Inc	1.99%
FDX	FedEx Corporation	6.21%
PYPL	PayPal Holdings Inc	0.50%
RL	Ralph Lauren Corp Class A	1.53%
SBUX	Starbucks Corporation	2.31%

Save portfolio »

This would be the interaction process between the service and the client, which integrates the data pipeline including the data extraction through the Yahoo API to seamlessly integrate model predictions to forecast stock prices and confidence, in order to reallocate properly the different assets.

8. Summary & Future Improvements

The report has thoroughly examined the rationale and methodologies behind the creation of a database model, alongside an automated data ingestion pipeline and a demonstration of a graphical interface. Although it presents a robust foundation, the product is still in a developmental phase, illustrating its potential rather than its completion. The current functionalities serve practical use cases, yet there is substantial scope for expansion and refinement to meet broader investment needs. Particularly, extending the data ingestion process to encompass stocks outside the S&P 500 index would cater to investors seeking more diverse market opportunities, thus broadening the application's appeal and utility. Additionally, integrating real-time data processing capabilities would substantially enhance the product's effectiveness, positioning it as a competitive alternative to existing market applications. The challenge lies in designing a system capable of maintaining ACID properties while ensuring the consistent, real-time delivery of varied data to end-users. Moreover, the incorporation of advanced AI and machine learning technologies within the user interface could expand the usefulness of the application through portfolio optimization. The introduction of a portfolio optimizer would mark a significant enhancement in the application's capabilities. This tool will allow users to receive stock predictions and adjust their investment portfolios based on individual risk tolerances and market conditions.

Addressing data governance is crucial due to the significant real-life implications of Optfolio. Ensuring data accuracy, security, and compliance is paramount, as financial data drives critical investment portfolios and strategies. Our governance framework emphasizes data stewardship, where a designated steward ensures the integrity and accuracy of data throughout its lifecycle. Additionally, compliance with stringent regulations such as the GDPR is essential. The application adheres to these regulations by implementing rigorous data protection and privacy measures, including strong access controls and explicit user consent for data collection. While the current version of the application demonstrates potential, its continuous development guided by data governance and the integration of data technologies will lead to sizable advancements. This development strategy ensures that the application not only meets the current demands of users but also adapts to future technological and regulatory changes, securing its relevance and efficacy.

Bibliography

Kivy 1.11.1 documentation. (n.d.). Kivy.org. *https://kivy.org/doc/stable/*

Silberschatz, A. (2020). *Database system concepts*. New York Mcgraw-Hill Education.

*Cong, F., & Oosterle*e, C. W. (2016). Multi-period mean–variance portfolio optimization based on Monte-Carlo simulation. Journal Of Economic Dynamics *And Control, 64, 23-38. https://doi.org/10.1016/j.jedc.2016.01.001*

WallStreetMojo Team, & Vaidya, D. (2024). Portfolio Optimization. Wall Street Mojo. https://www.wallstreetmojo.com/portfolio-optimization/

Kenton, W. (2022, 29 julio). What Is a Bloomberg Terminal? Functions, Costs, and Alternatives. Investopedia. https://www.investopedia.com/terms/b/bloomberg_terminal.asp

Seth, S. (2023). Basics of Algorithmic Trading: Concepts and Examples. Investopedia. https://www.investopedia.com/articles/active-trading/101014/basics-algorithmic-trading-concepts-and-examples.asp

Recitals of the GDPR (General Data Protection Regulation). (2021, February 15). General Data Protection Regulation (GDPR). https://gdpr-info.eu/recitals/