Smart Portfolio Advisor

Group 11

Namrata Thakur	A0261619B
Ouyang Hui	A0261875U
See Jia Fong Grace	A0261797M
Wang Zhipeng	A0261980Y

Table of Contents

1.Introduction

There is a common saying that monkeys make better fund managers than humans. This was based on a series of experiments run, the most famous one being a Chimpanzee, Raven, who threw 10 darts to pick stocks from a basket of 133 internet companies. Her stock picks delivered a 213 per cent gain in 1999, outperforming more than 6,000 professional brokers on Wall Street.

How can the average investor make quality investment decisions? Other than cherry picking stocks, there are many options in the market today, from mutual funds, to index investing and roboadvisories. One way to assess a portfolio's health is by measuring its returns relative to its risk, to understand if the portfolio is generating a reasonable return for the risk it is assuming. The Sharpe Ratio, proposed by William Sharpe, is one of the most popular methods that is used to measure risk adjusted relative returns in Modern Portfolio Theory.

In this project, we demonstrate the use of Sharpe Ratio as a measure of portfolio fitness to evolve a portfolio of stocks using Genetic Algorithm. We also apply LSTM modeling to forecast future stock prices.

1.1. Business Value

Modern fund managers rely on a variety of finance tools, most of which are expensive products. Hence our offering is targeted at people who are interested in investing in an optimal way but cannot afford the tools and time to do so.

Through the use of Genetic Algorithms, our system is able to pick out good portfolio allocations with high Sharpe Ratios every day, which investors can compare against their own portfolio holdings, to see if anything can be adjusted for improvement, for example, by addition of a particular stock that performed well in the recommended portfolio.

They can also understand how optimal their current allocation is compared to other possibilities. For example, if their current portfolio has a Sharpe Ratio of 1.5, but the genetic algorithm is able to produce a few portfolio alternatives that have Sharpe Ratio above 1.5, that may signal that their current allocation can be further optimized under current market conditions.

In addition to providing personalized and optimized portfolio, we also provide a rolling 2-week stock price forecast, based on the latest daily closing prices. Stock price

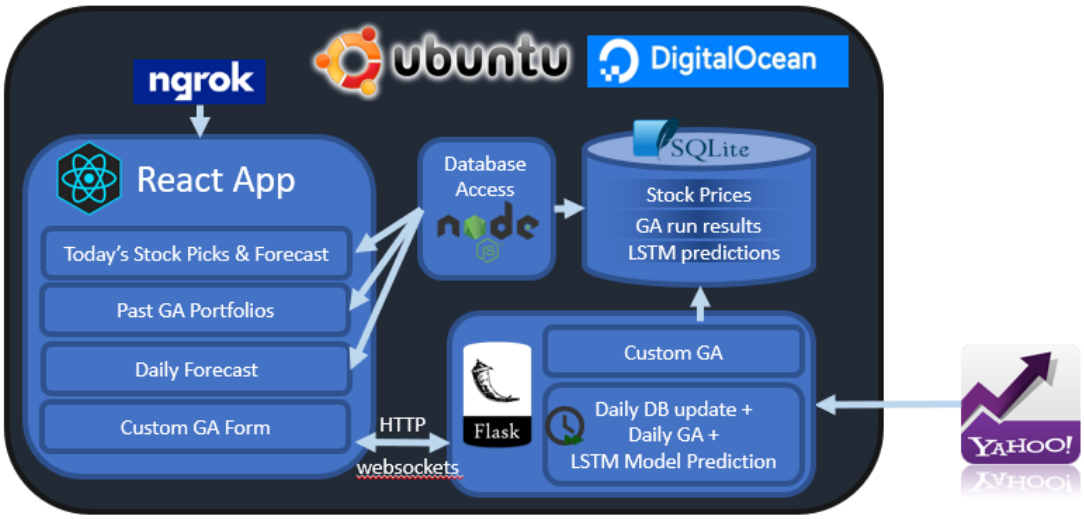
prediction is often a complex and time-consuming activity, involving the collection and analysis of related company/industry updates, government policies and broad market sentiment. We aggregated key macroeconomic and technical indicators and applied deep learning techniques to offer our users exclusive insights into future market performance. For the purpose of this project, a long short-term memory (LSTM) was used for predicting the future two weeks stock price trend by considering past 40 days data. Users are able to save time from performing their own forecasting, and rest easy knowing that we provide highly accurate forecasts.

1.2. Project Aim

The aim of this system is to

(a) approximate the best return to risk portfolio allocation possible at any point in time based on historical returns.
(b) forecast the stock prices for the next 2 weeks

2.System Architecture



The Smart Portfolio Advisor is split into 5 parts. n-grok, a react app, a flask server, a database server, and an SQLite database packaged into an Ubuntu Virtual Machine hosted in DigitalOcean, a cloud based infrastructure provider.

In the react application, all web pages pull preprocessed data from the SQLite database, except for the custom GA runs, which triggers a background thread on the server to run the GA for a basket of stocks picked by the user. As GA runs, the results of each epoch are published to the frontend via websocket communication after each epoch.

Within the flask server, there are 2 main services. Firstly, there is a service which responds to customized GA run requests from the frontend. Secondly, there is a cron job that does 3 functions. Firstly, it pulls the latest stock price information from Yahoo Finance on a daily basis. Secondly, it runs the daily GA to get the best portfolio for the day. Thirdly, it trains 1 LSTM model per ticker on the updated stock price information. It writes the results of all these 3 functions to the database, for the frontend to quickly pull out the desired information.

So, upon arriving at the Homepage, the user will get to see the Portfolio Recommendation of the day along with the Sharpe Ratio, percentage return of the investment and the risk percentage. These values are generated by pulling the daily stock price from Yahoo Financials and pushing those to the database (Daily DB Update) followed by running the daily GA and the LSTM Model Training on the updated dataset.

3. System Features

3.1. Stock Data Retrieval

We grabbed the data of 20 largest stocks in the S&P 500 in the past year from the Yahoo Finance API to form a dataset of historical stock data, and stored this in the database. 20 stocks were used for the run as a proof of concept. The daily stock prices are fetched from the Yahoo Financial API and the database is being updated using a CRON job scheduler on a daily basis as well.

3.2. Portfolio Recommendation of the day (based on Daily GA Run)

Daily GA Run refers to a series of runs which were done offline taking into consideration the latest daily data pulled from online finance APIs such as Yahoo Finance.

As the GA runs took a long time to run, and the time increasing with the number tickers, the team decided to:

- limit the number of tickers to 20, and population size to 10 as a proof of concept,
- run the GA offline during non trade hours and write the results to database for easy retrieval
- parallelize certain operations (see runtime optimizations)

The “daily GA Runs” for a time period stretching back to 6 Sep 2021 was performed to simulate the algorithm performance on historical data. Each run uses 12 months of historical data to compute annualized return and annualized volatility (standard deviation).

3.3. Custom Portfolio Recommendation (based on Custom GA Run)

Custom GA run refers to a run which the user may trigger on the fly, for a custom basket of stocks of his choice, to observe the portfolio optimization process

This is meant to allow the users to optimize across a basket of stocks already in his portfolio. As the algorithm runs, the console output will be pushed to the web frontend after every epoch, so that the user can receive updates on his portfolio optimization process

3.4. Stock Price Forecasting

Using the stored stock historical data, we trained the LSTM model to predict the stock price trend in the next 10 days based on the last 40 days of data.

After having trained the model, we save the predicted stock price for the next 10 days to the database and the trained LSTM model as a .hdf5 file.

Since the stock database is updated everyday with new addition of daily stock price, we are re-training the LSTM models everyday too. Re-training the model everyday makes sure the latest changes in the stock prices are captured in the model. Hence, we can give sharper predictions for the future dates .

On the frontend, the stock forecasts for tickers that are being recommended by the GA algorithm are featured on the HomePage. There is also a separate page called Forecast where users can input which stocks they are interested in, and our backend will pull the necessary forecasts from the database. These forecasts will be the output of the daily offline LSTM model training process that are stored in the database.

3.5. Historical Portfolio Recommendations

Our server can return the previous recommended portfolio in the History Portfolio page. Once they pick a specific day, they can get our Portfolio recommendation of that day and compare it with the present day. This will help the user understand how the portfolio recommendations have evolved over the time.

3.6. Financial News

Users can also decide their own investment decisions in response to the latest financial news.

We provide our users with the latest and most comprehensive financial news to help them decide their investment strategies based on what happens every day.

4. GA Algorithm for Portfolio Recommendation

4.1. Knowledge Identification

4.1.1. Fitness Function Definition

The Sharpe Ratio is computed using the following formula:

$$Sharpe = (\mu - rf) / \sigma$$

Where μ is the return of the portfolio,

rf is the risk free rate,

σ is the standard deviation of the portfolio

From the formula, it can be understood that the aim of the Sharpe ratio is to maximise the excess returns on the portfolio compared to holding a risk-free asset such as Treasury bonds over a certain time period, in relation to the risk of that same portfolio, which is measured as volatility of the prices over that same time period.

The interpretation for Sharpe ratio is as follows:

Sharpe Value	Interpretation
<1	Not good, taking more risk than is proportionate for the return
1-1.99	OK
2-2.99	Really good
>3	Exceptional

4.1.2. Data specification

To compute the Sharpe ratio, the following were needed:

(a) Historical Adjusted Closing Stock prices for up to 2 years per ticker (for the 20 identified subset of tickers), since the historical GA were being backtested for a period of 1 year
(b) Risk Free rate benchmarked by the 10 year US Treasury bills for up to 2 years.

Both pieces of data could be obtained from Yahoo Finance API.

4.1.3. Hyperparameter specification

Each GA Run is initialized with the hyperparameters in the table below. These were arbitrarily set, and the frontend supports customization of some of these parameters.

Hyperparameter	Description	Frontend setting
Run date	Date which GA is run for. The system will pull 1 year of historical data based on this date	
Stock tickers	The basket of stocks which the GA should optimize over	Y
Max Stocks	Max number of stocks which should receive allocations within the stock tickers	
Max Epochs	Number of epochs which the GA should run for, if it did not converge prematurely	Y
Max Depth	Max number of recursions the GA should run for. Each recursion is performed on a freshly generated random population. Max Depth is 5 for Daily GA offline run and it is kept 0 for the Custom GA online run.	
Selection Rate	The top N% of chromosomes which should form the next generation	Y
Crossover rate	The rate at which crossover should be permitted	Y
Mutation Rate	The rate at which mutation should be permitted	Y
Risk Free Rate	The returns which the investor would have achieved if he had invested in a relatively risk free asset instead	

4.2. Knowledge specification

4.2.1. Chromosome

A chromosome is a vector containing the proportions allocated to each ticker.

For example, if there were 3 tickers, stock A, stock B, stock C, the chromosome could look like [0.2, 0.3, 0.5]. This means that 20% of the portfolio should be allocated to stock A, 30% to stock B, and 50% to stock C. As such, the sum of all the ticker allocations in the chromosome needs to be equal to 1, and this could be achieved by dividing each number in the vector by the sum of all the numbers in the vector.

In the sample github code, every stock had an allocation. But as we were considering to scale up the basket of available stocks to 100 or 500, we decided to implement a maxStocks hyperparameter to limit the number of stocks which would receive allocations, or the allocation would be spread too thinly over a large basket of stocks.

Hence the code to generate the chromosome was modified to only choose [maxStocks] stocks out of X number of possible tickers.

```
# Chromosome Definition
def chromosome(n, totalstocks):
    ''' Generates set of random numbers whose sum is equal to 1
        Input: n = number of stocks we want to invest out of
         totalstocks. totalstocks = universe of investible stocks
        Output: Array of random numbers'''

    ch = np.random.rand(n)
    ch = ch/sum(ch)
    #disperse ch across the available stocks
    portfolio = [0] * totalstocks
    i = 0
    while (i != n):
        #look for portfolio[index] == 0 to replace with chromosome
        index = np.random.randint(0, totalstocks)
        if (portfolio[index] == 0):
            portfolio[index] = ch[i]
            i = i + 1

    return portfolio
```

4.2.2. Fitness Value (Sharpe) Computation

The computation was done in accordance with a few online tutorials [4,5]

1. **Computation of Daily Return**

   Firstly, the daily return was computed for each stock using the adjusted closing price.

$$\text{Daily Return (\%)} = \frac{\text{Today's price} - \text{Yesterday's price}}{\text{Yesterday's price}}$$

This computation was done as the data is pulled from the source and written into the database to shorten GA computation time.

2. **Computation of Annualized Daily Return**

Next, the annualized daily return was computed based on 1 year of daily return data.

The average of all the daily returns in 1 year was taken and multiplied by the number of trade days, which is commonly assumed to be 252 in most tutorials

$$\text{Annualized Daily Return (\%)} = \text{Mean (Daily Return)} \times \text{Number of Trade Days}$$

3. **Computation of Annualized Portfolio Variance**

The computation of daily portfolio standard deviation was slightly more complex, because most Sharpe Ratio computation tutorials focused on the standard deviation for a single stock ticker, whereas we needed a weighted computation of portfolio standard deviation

For an N-stock portfolio, portfolio variance is computed with the following formula:

$$\textbf{Portfolio variance} = \sum_{i=1}^{N} W_i \sigma_i + \left[\sum_{j=1}^{N} \sum_{k=1}^{N} W_j W_k \, COV_{j,k}\right]$$

Then, the annualized Portfolio Standard Deviation was computed using the following formula:

$$\text{Annualized Standard Dev(\%)} = \sqrt{Portfolio\ Variance} \times \sqrt{\text{Number of Trade Days}}$$

4. **Computation of Risk Free Rate**

The risk free rate was computed by taking the mean of the 10Y US Treasury bill yield (ticker: ^TNX) over the last 12 months.

5. **Computation of Annual Sharpe Ratio**

   The excess return on the portfolio was computed by subtracting the risk free rate from the annualized daily return.

   Finally the Sharpe ratio was computed by dividing the excess return by the volatility, which was measured by the annualized standard deviation of the portfolio.

$$\text{Sharpe Ratio} = \frac{\text{Annualized Daily Return} - \text{Risk Free Rate}}{\text{Annualized Std Dev of Portfolio}}$$

4.2.3. Population Selection and Generation

Each GA run had a default population size of 10. We opted to keep the population size smaller for faster runs, as we realized that computing the fitness function seemed to be a computation bottleneck (see runtime optimization).

At each epoch, the fitness values of all the population members would be computed. Then, we would **rank** them according to their fitness values, and take the top X portfolios according to [selectionRate]. For example, if [selectionRate] was 0.4, we would take the top 4 chromosomes (out of 10) for mutation and crossover.

With these 4 chromosomes, we would then replicate them until the desired population size (i.e. 10) is reached. So the new population would look like this before mutation and crossover:

[Top1, Top2, Top3, Top4, Top1, Top2, Top3, Top4, Top1, Top2]

We ran a few GA batches where we subjected all the chromosomes in this population to the crossover/mutation rate, and some succeeding generations got potentially worse results! Hence, to preserve the performance to be on par or better only, the first chromosome, which represented the best portfolio for the current generation, was excluded from crossover and mutation, in order to preserve a baseline Sharpe for the next generation that was at least as good as the current generation.

4.2.4. Crossover

In literature, there are many possible forms of crossover operations, e.g. single point crossover, two-point crossover, uniform crossover, arithmetic crossover and heuristic crossover.

We implemented **single point crossover** for simplicity. However, because we were intending to scale up the number of tickers, it was possible for the number of ticker allocations to increase until every possible ticker had a tiny allocation. Consider a worst case scenario where out of 20 stocks, parent 1 had all its 10 allocations in stock 1-10, and parent 2 had all its 10 allocations in stock 11-20. If crossover point is set at the midpoint of both stocks, the child will end up having 20 stock ticker allocations in the portfolio! The problem will get worse if not rectified in a larger basket of potential stocks, and as more crossovers are performed. [1]

To prevent the number of ticker allocations from increasing too much, we randomly dropped allocations if the number of ticker allocation exceeded [maxStocks]. We did not check if we picked the same random index, but the function does a reasonable job to keep the number of ticker allocations in check.

Then we had to re-normalize the final array to ensure that the sum of allocations was 1.

Below is the code for the crossover. We only performed the crossover on the chromosome that was selected for it, thus the function only returns 1 child instead of 2.

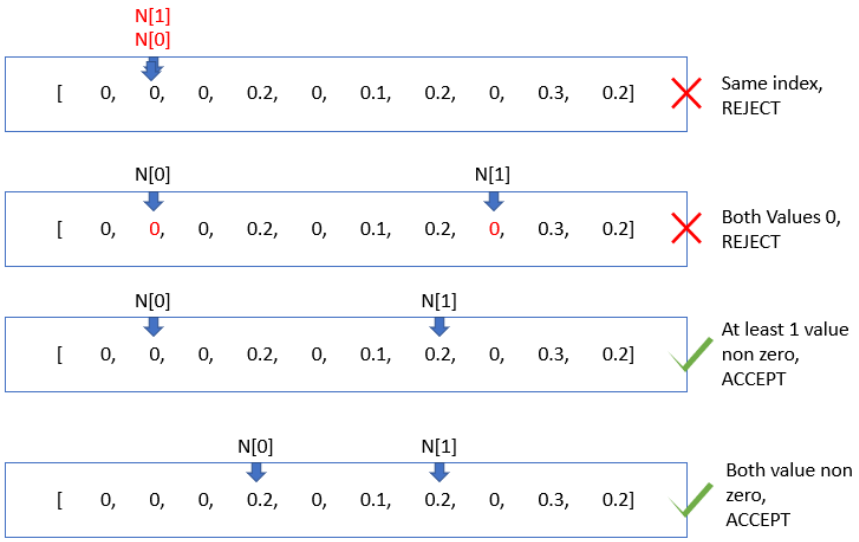
```
def crossover(parent1, parent2, maxStocks=10):
    length = len(parent1)
    crossoverPt = np.random.randint(0, length)
    child = [0] * length
    numStocks = 0
    stockLoc = []
    for i in range(length):
        if (i<crossoverPt):
            child[i] =  parent1[i]
        else:
            child[i] =  parent2[i]
        if (child[i] > 0.0):
            numStocks+=1;
            stockLoc.append(i)

    if (numStocks > maxStocks):
        toDrop = numStocks - maxStocks
        for j in range(toDrop):
            dropInd = stockLoc[np.random.randint(0, len(stockLoc))]
            child[dropInd] = 0.0
    sumOfChild = sum(child)
    if (sumOfChild > 0.0):
        child = [float(i)/sumOfChild for i in child]
    return child
```

[1] The reverse, where the stock ends up with zero allocations is also possible but not likely. In the event that there is zero allocations, the stock fitness will be low, hence it is likely to get excluded for the next epoch.

4.2.5. Mutation

For mutation, we swapped 2 indices within a chromosome. However, as we expected that our chromosome could be potentially "sparse", i.e. with only [maxStocks] ticker allocations out of a big basket, to make the mutation useful, we avoided swapping ticker allocations for tickers with 0 allocations, as that would have no impact on the chromosome allocation.



Below is the code for the mutation. The highlighted condition in the while loop will stay true until the 2 selected indexes are different, and either or both indexes are pointing to non-zero ticker allocations.

```
def mutation(parent):
    child=parent.copy()
    n=np.random.choice(range(len(parent)),2)
    while (n[0]==n[1] or (child[n[0]]==0.0 and child[[n[1]]==0.0])):
        n=np.random.choice(range(len(parent)),2)
    child[n[0]],child[n[1]]=child[n[1]],child[n[0]]
    return child
```

4.2.6. Termination Criteria

4.2.6.1. Single run termination criteria (convergence or max epoch)

There are 2 termination criteria for each GA run, the run will terminate when either is fulfilled.

The first criteria is convergence. At every epoch, the population is sorted according to their fitness, and the elite chromosomes are compared against the elite chromosomes from the previous epoch. In this case, with a selection rate of 0.4, the number of elite chromosomes per epoch was 4 (out of 10). The pairwise Euclidean distance is computed, and if the Euclidean distance does not exceed 0.0001, a convergence counter is incremented. If the convergence counter is incremented 3 times, we terminate the run. By examining the portfolios, it can be seen that the best portfolios of the last 3 epochs of any GA run may not be exact duplicates, but close enough.

The second criteria is max epoch. When the GA fails to converge, we cut it off when maxEpochs exceed 40 as usually by this time, the increment in Sharpe is not significant (due to population homogenity) and running with another seed population can potentially introduce more improvements compared to continuing along the current epoch.

4.2.6.2. Multi-run termination criteria (Sharpe Ratio 2, recursion depth of 5)

A GA Run starts off with a randomly initialized population of pop_size, typically 10 members. Usually after 1 run (up to 40 epochs), the Sharpe ratio may not be ideal because the run converges at a local maxima that does not yield a good Sharpe ratio. For daily GA runs, a portfolio allocation with Sharpe Ratio > 2 was desired, as it signaled a good return to risk ratio. Since the daily GA run was done offline, the program was made to rerun recursively for a maximum of 5 times (recursion depth <= 5), in the hopes of hitting a good start search point (initial population) which can eventually find a good local maxima with a minimum Sharpe Ratio of 2. While recursing 5 times did not always result in the target Sharpe Ratio of 2, it almost always resulted in an improved Sharpe Ratio, compared to increasing the number of epochs, selection rate, mutation rate, or crossover rate.

We found that the data influenced how easily the desired Sharpe ratio could be achieved. In particular, dates that were more recent yielded lower Sharpe ratios compared to dates in 2021. Before we enforced a maximum depth of 5, the algorithm would recurse for many times for portfolios in Jun to Oct 2022, being unable to reach the target Sharpe ratio of 2.

This could be due to the fact that the 1 year return (e.g. Oct 22 compared with Oct 21) started to turn negative on May 22, making it harder to find a decent performing portfolio in the limited basket of stocks that would give a high return.

Below is a summary of Daily GA Runs for 06-Sep-21 to 25-Oct-22, capped at 5 runs per day

Month	Av. Sharpe	Runs	S&P500 1-yr %change
Oct-22	0.847959	90	-15.531
Sep-22	1.123859	110	-15.5536
Aug-22	1.261385	115	-11.2369
Jul-22	1.192506	105	-4.68113
Jun-22	1.31393	95	-10.6377
May-22	1.700427	73	-0.31593
Apr-22	2.360509	26	0.090374
Mar-22	2.540174	23	15.55188
Feb-22	2.585688	21	16.40865
Jan-21	2.972397	21	23.30379
Dec-21	3.307817	23	28.84886
Nov-21	3.61518	22	27.78956
Oct-21	3.033292	21	42.82695
Sep-21	2.897593	19	30.16647

4.3. Knowledge Refinement

4.3.1. Algorithm Tuning

Some tweaks were made to the algorithm to run more efficiently. Some of these included:

(a) At first we employed Generational GA, where we attempted to perform crossover/mutation on all chromosomes in the next generation according to the crossover rate and mutation rate. However, upon examining the results, we quickly realised that subsequent epochs could yield less optimal portfolios, hence we opted to do Steady State GA instead. We excluded the fittest individual from crossover/mutation possibility as we did not want to lose our best portfolio, the rest of the portfolio was subject to high crossover/mutation rate as we only had 9 other members, and it was critical to try as many permutations as possible to figure out possible portfolio improvement.

(b) We introduced recursion to restart the GA runs when the desired Sharpe ratio was not met, and also capped the GA runs when they failed to achieve a good result after 5 runs. The reason it was done this way was because we realised that given our population size of 10, the GA runs tend to get stuck if local maxima very easily, yielding suboptimal sharpe portfolios. Hence it is better to restart the GA, as starting from a different start point can be more helpful to improve the GA rather than continuing along the current epoch, as the GA could have been evolving along a path with many suboptimal local maximas.

(c) From(a), we also considered if getting a better quality initial population would be helpful and experimented with increasing the population size. The hypothesis was that more random chromosomes present would increase the quality of the fittest individual from the first epoch, and hence set the stage for better subsequent Sharpe ratio. However, after a few runs, we realized that increasing the population size came at a cost of fitness computation, ranking, additional crossover/mutation computation, and these added significantly to the runtime overhead. What was worse was that the random allocation of stock tickers often resulted in the generation of many low quality chromosomes, and hence did not contribute to the betterment of the Sharpe ratio significantly. We ended up expending a lot of computation on examining and tweaking low quality chromosomes. Hence, the conclusion was that expending extra effort to get a better initial population may not benefit much as due to significant performance issues.

(d) Instead, we found that choosing a smaller population size rather than larger one allows for faster convergence, even at high mutation and crossover rates. This was the preferred strategy because it was “fail-fast”. A convergence counter was implemented to detect early convergence so that computation time of subsequent epochs could be saved when the improvement in sharpe was minimal. Early run termination, coupled with restarting at different initial populations, helped to bring up the portfolio Sharpe in an acceptable timeframe.

(e) Last but not least, we also experimented with changing selection rates to between 0.3 to 0.6 and increasing mutation/crossover rates to as much as 0.9, but no significant improvement in Sharpe was observed.

4.3.2. Key Performance Indicators

Some performance measures are described below

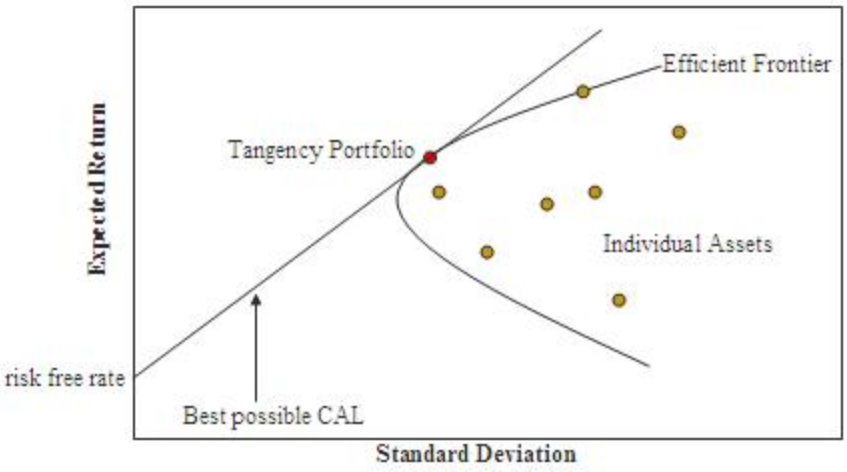
Measure	Sharpe	Remark

	Improvement	
Average Sharpe Improvement per Epoch	+ 0.588	Evaluated by taking the fittest individual at each epoch (popid=0) and comparing from epoch 0 to maxEpoch, for each run in a day. The average across all runs in a year from 02-Sep-2021 to 06-Sep-2022 was computed
Average Sharpe Improvement due to restart	+ 0.913	For each day, if there was more than 1 GA run, the min and max final Sharpes were extracted and their difference computed. The average range (divided by number of days with GA runs > 1) was then taken to be an indicator of the Sharpe improvement
Average Difference versus 1000 Randomly Generated Portfolios	- 0.099	This was a surprising result. Visually the GA generated portfolio appeared to lie near the Efficient Frontier, indicating a relatively efficient allocation (see below). However, 91/260 days, GA generated portfolio underperformed randomly generated portfolios. More discussion below

Efficient Frontier Hypothesis

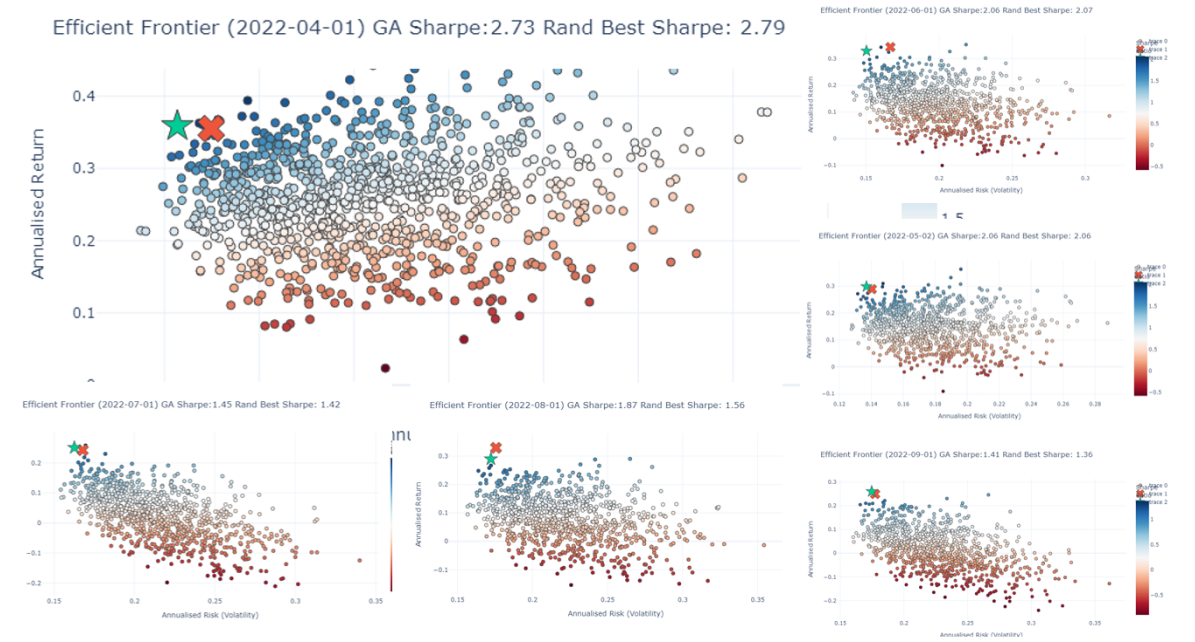
The efficient frontier is the set of optimal portfolios that offer the highest expected return for a defined level of risk or the lowest risk for a given level of expected return. [6]

It is typically plotted as a risk versus return graph. Each point in the graph represents a portfolio allocation. Portfolios below the curve are sub optimal, so only the upper part of the curve is considered. Portfolios to the right are high risk and also not recommended



We adapted a python notebook to generate out the efficient frontier for the days that the daily GA was run. The notebook generated 1000 random portfolios with the 20 tickers we defined, and calculated the excess risk and returns for these portfolios to plot on the graph. Most of the days, the GA generated portfolio consistently appeared to sit near the Efficient Frontier, indicating that it was a near optimal portfolio for the risk assumed. However, in terms of Sharpe ratio, it couldn't beat the best of the randomly generated portfolios.

The Efficient Frontier diagrams below are a sampling from the set of 260 images generated. The green star represents the best random portfolio, and the red cross represents the portfolio recommendation by the GA Run. The rest of the datapoints are the randomly generated portfolios (999) and the colour scheme of the datapoints indicates the desirability of the Sharpe Ratio (red=poor, bl `ue= good)



We try to explain why this happens. One possibility was that GA uses less random portfolios, hence unable to approximate the global optimum better. The minimum number of portfolios generated by GA runs is 50, and the maximum is less than 2000. However, in general, the number of portfolios generated would not hit 2000 due to constraints such as the fittest chromosome in each generation being omitted from crossover/mutation, and portfolios generated by crossover/mutation have more limitations than a truly random distribution as they are constrained by the crossover/mutation operations. Hence GA is limited in its ability to extend over the search space compared to random portfolio picking.

The other theory is that the stock markets have been quite volatile for the past 1 year. There are many fluctuations in the stock prices and risk free rates, and these have caused there to be many local maxima that the GA got stuck at and 5 permitted restarts were not sufficient to locate better maximas.

You may have recalled that we decided not to increase the population size because it resulted in more computation (e.g. ranking, crossover, mutation, low convergence). Increasing population size would mean that more random portfolios would be generated at the first epoch, mirroring the random portfolio picking strategy. However, the subsequent runtime cost in terms of performing all those operations to evolve the

portfolio also increases. Since random portfolio picking seems to work better and run faster, it seems to suggest perhaps GA is not quite suited for this task after all.

4.3.3. Run time optimizations

We observed that computing the fitness values for each population was taking up a significant amount of time. The console printout for each population member's sharpe ratio took about 2 seconds and the original GA run took about 20 minutes to complete 1 run. Hence, we decided to parallelize the fitness value computation across the population using Python's multiprocessing library. This cut down the runtime for a full GA run (40 epochs) significantly from 20 minutes to less than 4. Parallelization was also attempted for the crossover and mutation functions for each chromosome, but the time savings was not significant.

Parallelization for python worked differently on Unix systems versus Windows systems. On Unix, a fork was made with the parent's thread context duplicated to the child thread, whereas in Windows a new python interpreter was "spawned", which was lacking in the parent thread context. This meant that certain functions were missing in memory, and the only way around it was to refactor all the code into a python file and read it back into the child thread. For ease of deployment, we decided to field the backend into a Unix based OS (Ubuntu) instead.

To produce the database of GA runs for a year since Sep 2021, the python notebook was uploaded to a 4 core VM in DigitalOcean and 3 notebook duplicates[2] were made to run GAs for different time frames (e.g. Jun - Sep 22, Mar - Jun 22, Sep 21 - Mar 22) while writing to the same sqlite database file. This way, the entire database could be generated fairly quickly (< 12 hours). This mode of working also facilitated bug fixes in the code, which would otherwise have been rather cumbersome to implement.

4.4. Limitations and Improvements

4.4.1. Too Slow!

The greatest limitation of the Genetic Algorithm is speed, or the lack of it. The algorithm takes very long to run. It requires further optimization which we were not in time to do. Our lack of familiarity with python APIs led us to code the algorithm to perform in relatively inefficient ways. Towards the tailend of the project, when running the Efficient hypothesis notebooks, we saw that it is possible to generate many portfolios and

[2] 3 Notebooks were observed to bring CPU utilization to 100%. 1 notebook (60%) and 2 notebooks (80%). The workload was split arbitrarily according to observation that earlier timeframes (Sep 2021 to Mar 2022) seemed to run faster due to lower recurse depths

evaluate them quickly with certain optimizations in place. However we were not in time to replicate these changes to our main system setup.

4.4.2. Overly-dynamic portfolio recommendations

There is a cost to buy and sell stocks to be in line with the portfolio recommendation. Depending on the data, the allocation could look vastly different from day to day, confusing investors. Investors are not day traders so their strategy might not involve buying/selling on a daily basis and in fact, excessive buying and selling could eat into their returns due to trading fees imposed by any trading platform of their choice.

Some things we could try to give stability to the recommendation would be firstly, to introduce a penalty to the fitness function that penalizes it for any buy/sell decisions. We could reference an existing trading platform's fee structure (e.g. SGX's) to compute a penalty based on the buy/sell volume.

Secondly, we could also try to use yesterday's recommended portfolio as a starting point for today's GA run. This approach may be limited in reaching the efficient frontier, but given a good "yesterday" portfolio that sits relatively near the efficient frontier, and further, computing today's "best random portfolio" as a guide to an achievable Sharpe, it might be possible to find an intermediate portfolio between the "today's best random" and "yesterday's GA recommendation" that has a buy/sell penalty of less than some penalty threshold

4.4.3. Only Linux

As mentioned, the code can only run in Linux, it would be good to refactor the code so that we can run it on Windows too

5. LSTM Model for Stock Price Forecasting

5.1. Knowledge Identification

Stock price can be influenced by many interconnected factors, it is not only directly impacted by the business financial status, reputation and future planning of an enterprise, it will also be impacted by numerous external factors such as natural disasters, global economic situation, interest rate and so on. To make a reliable prediction on stock future trends, multivariate information has to be gathered to train the model. For the purpose of this project, we collected the data in three aspects: the historical data that show the basic information of the stocks; the macroeconomic indicators which reflect overall health of the economy and have long-term impacts of stock market; and the technical indicators which are the mathematical pattern derived from historical data widely used by professional investors to make trading decisions.

Category	Indicator	Description	Source/ Formula
Historical Data	Volume	Stock volume is counted as the total number of shares that are actually traded (bought and sold) during the trading day. Volume can indicate market activity. For example, stocks with high volume are usually viewed to have less Volatility and higher liquidity (Mitchell, 2022).	Yahoo Finance
	Adjusted Close Price (Adjclose)	The adjusted closing price amends a stock's closing price to reflect that stock's value after accounting for any corporate actions, which is often used when examining historical returns or doing a detailed analysis of past performance (Ganti, 2021).	Yahoo Finance
Macroeconomic Indicators	CBOE Volatility Index (VIX)	The Chicago Board Options Exchange (CBOE) Volatility Index measures stock market volatility by tracking trading in S&P 500 options. For example, if the VIX value declines, the S&P 500 is predicted to be stable in the near future, while if the VIX value increases, it may indicate that S&P 500 is falling (Kuepper, 2022).	Yahoo Finance
	U.S. Dollar Index (USDX)	The U.S. dollar index (USDX) is a measure of the value of the U.S. dollar relative to six of America's most significant trading partners (the Euro, Swiss franc, Japanese yen, Canadian dollar, British pound, and Swedish krona). The USDX value can indicate the U.S. dollar's value in global markets (Lee, 2022).	Yahoo Finance

Technical Indicators	Moving average convergence divergence (MACD)	Moving average convergence divergence (MACD) is a trend-following momentum indicator that is calculated by subtracting the 26-period exponential moving average (EMA) from the 12-period EMA. And EMA is a type of moving average (MA) that places a greater weight and significance on the most recent data points. MACD's histogram is often used to identify potential failure and reversal of stock price. For instance, MACD crossing above zero is considered bullish, while crossing below zero is bearish, and the further from the zero line the stronger the signal (Dolan, 2022).	MACD = 12_Period_EMA - 26_Period_EMA
	Relative strength indicator (RSI)	The Relative Strength Index (RSI) is one of the widely used momentum oscillators on a scale of 0 to 100 which measures both the speed and rate of change in price movements within the market. Usually, RSI above 70 means the market condition is overbought, and RSI below 30 means market condition is oversold (Fernando, 2022).	RSI = 100 – [100 / (1 + (Average of Upward Price Change / Average of Downward Price Change)]
	Average True Range (ATR)	The average true range (ATR) is an indicator for price volatility showing the average price variation of a stock within a given time period, generally using 14 days. ATR is used by investors to determine when to trade(Hayes, 2022).	TR = max [(high - low), abs(high - $close_{prev}$), abs(low - $close_{prev}$)] $ATR = \frac{1}{n}\sum_{i=1}^{n} TR_i$

To get the stock information and the value of macroeconomic indicators, a python module called yahoofinancials is used to fetch the relevant data from Yahoo Finance. And the definition and formula about technical indicators are based on financial websites such as Investopedia.

5.2. Knowledge Specification

5.2.1. Data preprocessing

It is necessary to standardize data before training the model as values of input features may vary from one to another. For example, the AAPL stock volume is much higher than its relative strength indicator which has values from 0 to 100. MinMaxScaler was implemented to ensure the value of each feature is normalized to a scale of 0 to 1.

Moreover, calculating the technical indicators needs a certain amount of data. For instance, the average true range (ATR) evaluates the average price variation of a stock within a 14 days period, therefore, at least 14 days data is needed and the first 13 rows

would not have values. Enough data should be obtained for calculating the technical indicators and NaN data should be removed in the data preprocessing stage. Following is a snapshot of the dataset:

date	volume	VIX	USDX	MACD	RSI	ATR	adjclose
2017-12-05	0.167850	0.029644	0.336613	0.482733	0.273053	0.039151	0.000000
2017-12-06	0.167945	0.025428	0.352776	0.482518	0.379945	0.041202	0.004436
2017-12-07	0.138825	0.013734	0.366128	0.481350	0.352007	0.043078	0.003355
2017-12-08	0.151584	0.005847	0.373156	0.487047	0.481156	0.049701	0.009581
2017-12-11	0.135631	0.002584	0.371047	0.495740	0.519986	0.055034	0.013570

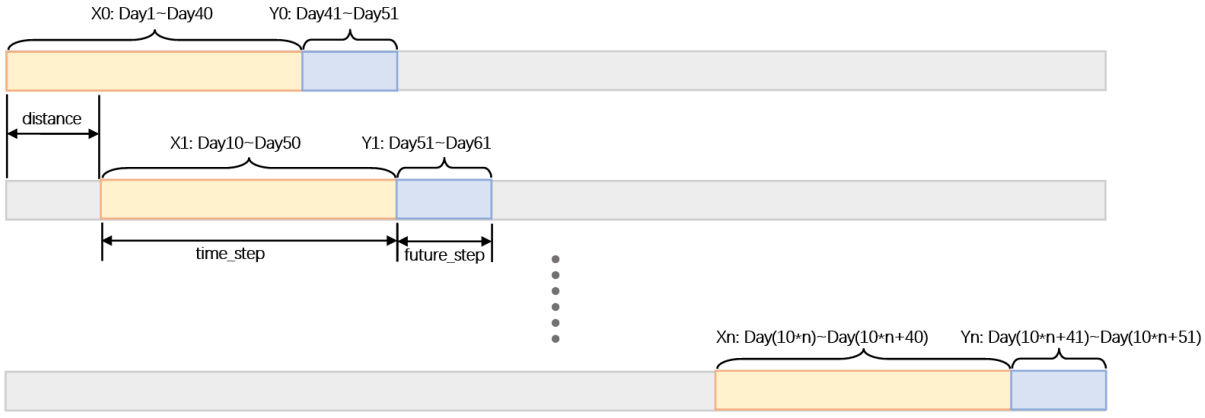
5.2.2. Dataset creation

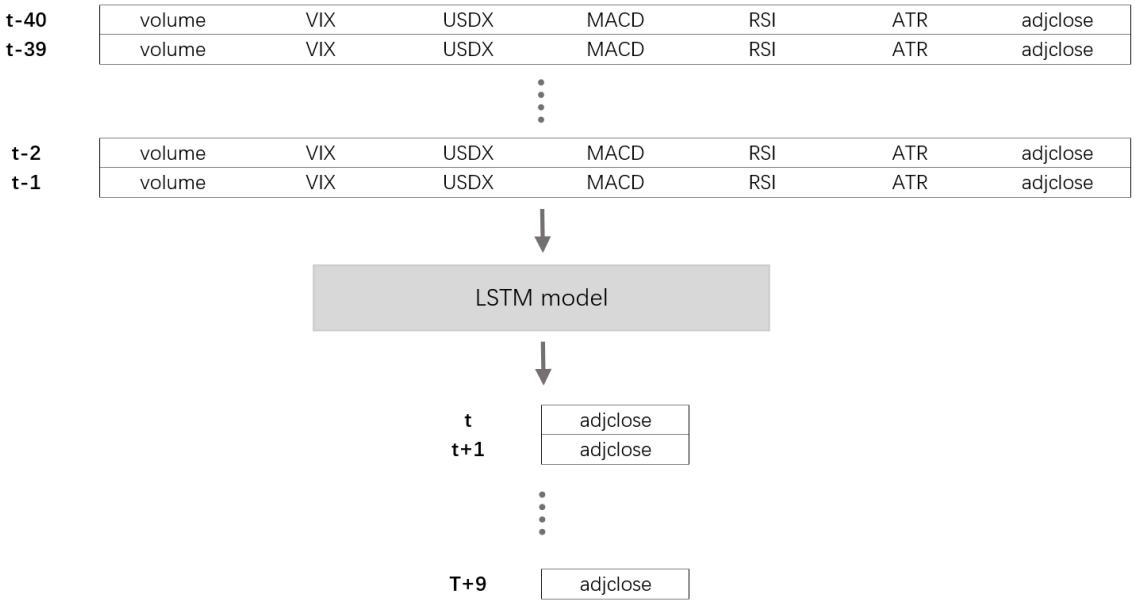
Our input data would be 40 days values and output data is the further 10 days adjclose value. The dataset is created in the following way using a sliding window with a step of 10. 10 days corresponds to roughly 2 weeks of data (weekends being non trade days)

time_step = 40

future_step = 10

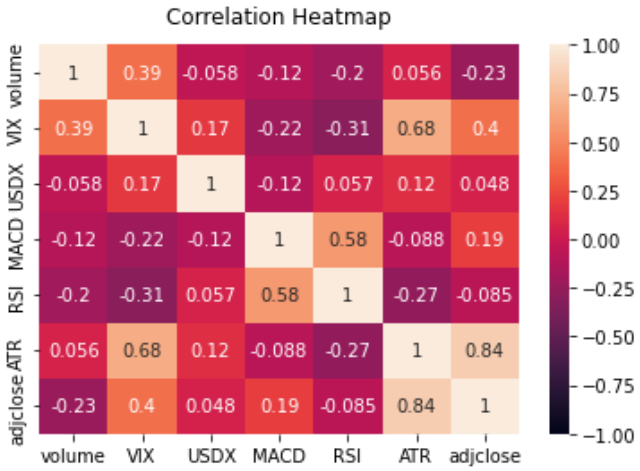
distance = 10





5.3. Knowledge Refinement

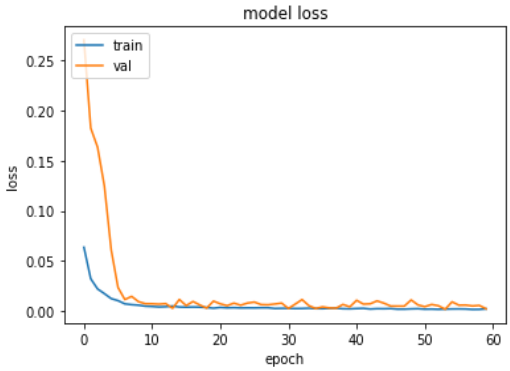
To select an appropriate set of data, we want the factors that contribute to the stock price fluctuation but do not include duplicates. Therefore, to identify the features that have potential influence and remove duplicate features, a correlation heatmap of all input features is generated. In this project, features that have high correlation coefficient with our target “Adjusted Close Price” are considered as influential features and features that have high correlation coefficient above 0.7 is considered as duplicate features to be removed. It is validated that none of the pairwise correlations between the chosen features exceeds the threshold.



To identify optimal model architecture and hyperparameters, we trained the models with different combinations of number of layers, learning rates and batch sizes, and compared the mean squared error (MSE) on both training and validation sets to evaluate their performance. Optimizer Adam and epoch size of 60 was used, and each combination of hyperparameters was executed 5 times to get the average test loss and average validation loss. The following table shows the results:

Neuron number	Batch size	Learning Rate	Average test loss	Average validation loss
(64)	5	0.1	0.005010980	0.0171092
		0.01	0.000838538	0.006069266
		0.001	0.001719581	0.005473279
	10	0.1	0.010183932	0.026804345
		0.01	0.00088972	0.00523274
		0.001	0.00151668	0.00367822
(128, 64)	5	0.1	0.08221204	0.1908787
		0.01	0.0009692	0.017170
		0.001	0.00156414	0.00922018
	10	0.1	0.06777188	0.31198183
		0.01	0.0010267	0.01713104
		0.001	0.00138937	0.004644903
(128, 64, 30)	5	0.1	0.0346339	0.268277
		0.01	0.00149486	0.051282045
		0.001	0.0022146	0.012507
	10	0.1	0.03024959	0.214945
		0.01	0.0012523	0.054831
		0.001	0.00226974	0.01361636

The results indicate that the model with batch size 10, learning rate 0.001 and single layer LSTM shows the best performance compared to all other models.



5.4. Limitations and improvements

Due to the time limitation of this project, we only collected and analyzed the historical data and macroeconomic indicators that can be obtained easily and freely from platforms such as Yahoo Finance, and the technical indicators that can be calculated from historical data. But the enterprise's internal factors such as business financial status was not analyzed in this project, as such information is not open to the public generally. And also, other potential influencers such as an enterprise's reputation and future planning, international events, natural disasters which are difficult to be quantified are also not included in this project.

Another issue is that in practice, the factors that impact stock price trends are continually changing, which means that using past 5 years data would not be suitable to train a model to predict today's stock market. For example, USDX considers six currencies of most significant trading partners of America, but these six currencies are not always static: the German mark and many European currencies had been replaced by the euro. It is possible that in the near future other currencies will supplant the currencies in the index as the global economics is continuously changing. Consequently, using the data many years ago may not accurately reflect today's stock market situation.

Future improvements could be done in the following aspects:

(a) Denoising using wavelet transform

In practice, stock price fluctuation is full of uncertainty, and the existence of the high noise will lead to the poor performance of the prediction model. It is suggested that Haar wavelet transform can be used to denoise the time-series data such as stock price data, and usually models can perform better with denoised data (Ortega & Khashanah, 2013).

(b) Further hyperparameter tuning

For this project, we evaluated the LSTM models by using MSE as loss function. In the future, more indexes can be used to measure the accuracy and reliability of the prediction such as Root Mean Squared Error(RMSE) or R-Squared. Moreover, more could be explored about optimization methods and neural network architectures and different neuron numbers.

(c) More data

Stock price forecasts could be a sophisticated task in reality when considering daily events and news. Additional techniques such as topic modeling can be applied to

quantify the news gathered from various platforms such as newspapers, blogs and social media. With multifaceted data, our system can give our user more valuable guidance on stock investment.

6. Conclusion

In conclusion, the team has developed and delivered a prototype portfolio recommendation system that is able to advise the users how to invest their monies in an optimal manner.

The team learnt how to evaluate portfolio allocations to maximize rewards to risk ratios, and how to apply GA to solve optimization problems, as well as apply LSTM to do stock price forecasting. The team also gained hands-on experience in deploying applications to the web.

7. References

- (n.d.). Most successful chimpanzee on Wall Street. Guinness World Records. https://www.guinnessworldrecords.com/world-records/most-successful-chimpanzee-on-wall-street
- (2021). Portfolio Optimization using Genetic Algorithm. github. https://github.com/naresh-dscience/Portfolio-Optimization-using-Genetic-Algorithm
- (2021). Efficient Frontier Python. github. https://github.com/rian-dolphin/Efficient-Frontier-Python
- The Trader's Code. (2020, October 17). How to Calculate Daily & Annual Sharpe in Excel [Video]. YouTube. https://www.youtube.com/watch?v=_jGi7m8QlkM
- Tactile Trade. (2020, November 13). How To Use The Sharpe Ratio + Calculate In Excel [Video]. YouTube. https://www.youtube.com/watch?v=17Q4m0IUqsY
- (2022). Efficient Frontier: What It Is and How Investors Use It. Investopedia. https://www.investopedia.com/terms/e/efficientfrontier.asp
- Dolan, B. (2022, October 21). MACD indicator explained, with formula, examples, and limitations. Investopedia. Retrieved October 27, 2022, from https://www.investopedia.com/terms/m/macd.asp
- Fernando, J. (2022, October 21). Relative strength index (RSI) indicator explained with formula. Investopedia. Retrieved October 27, 2022, from https://www.investopedia.com/terms/r/rsi.asp
- Ganti, A. (2021, May 19). Adjusted closing price definition. Investopedia. Retrieved October 27, 2022, from https://www.investopedia.com/terms/a/adjusted_closing_price.asp
- Hayes, A. (2022, September 9). Average true range (ATR) formula, what it means, and how to use it. Investopedia. Retrieved October 27, 2022, from https://www.investopedia.com/terms/a/atr.asp
- Kuepper, J. (2022, September 22). Cboe volatility index (VIX): What does it measure in investing? Investopedia. Retrieved October 27, 2022, from https://www.investopedia.com/terms/v/vix.asp
- Lee, M. (2022, July 13). How U.S. stock prices correlate to the value of the U.S. dollar. Investopedia. Retrieved October 27, 2022, from https://www.investopedia.com/ask/answers/06/usdollarcorrelation.asp

- Mitchell, C. (2022, September 22). How to use stock volume to improve your trading. Investopedia. Retrieved October 27, 2022, from https://www.investopedia.com/articles/technical/02/010702.asp

- Ortega, L., & Khashanah, K. (2013). A neuro-wavelet model for the short-term forecasting of high-frequency time series of Stock returns. Journal of Forecasting, 33(2), 134–146. https://doi.org/10.1002/for.2270

Appendix I: Proposal

Date of proposal: Sep 2022

Project Title: Smart Portfolio Advisor

Group ID (As Enrolled in LumiNUS Class Groups): 5

Group Members (name , Student ID):

Group Member	Student ID
Namrata Thakur	A0261619B
Ouyang Hui	A0261875U
See Jia Fong Grace	A0261797M
Wang Zhipeng	A0261980Y

Background

Against a backdrop of high inflation and rising interest rates, many people today turn to investing to preserve the long term value of their money. There are many options in the market today, from government bonds to Investment Linked Policies, to roboadvisories, mutual funds, endowments and DIY investing. The investor is spoiled for choices. How can any investor understand where his chosen portfolio stands? Other than expending effort and time to research or buy into managed solutions that incur high fees, we attempt to create a low cost advisory service that will help the user to understand the potential of the offerings he chooses.

Scope

In our project proof of concept, we only take the top 20 stocks from the S&P 500 index as our initial basket of stocks to optimize. Any recommended portfolio will have a

maximum of 10 tickers, picked from the best performing stocks out of these 20 stocks. However, our solution could be in theory further expanded to any asset class, be it bonds, mutual funds or etfs.
In addition, we employ LSTM modeling to predict future stock prices. This would be useful for investors looking to get a one-glance understanding of any offering's near term potential value.

Aim

The aim of this system is to

- approximate the best return to risk portfolio allocation possible at any point in time based on historical returns.
- forecast the stock prices for the next 2 weeks

Objectives

The objective of this project is to develop and deliver a system that is able to deliver optimal portfolio recommendations and stock price forecasts to users in a user-friendly interface.

Project Description:

Dataset

For GA to find an optimized portfolio,daily adjusted closing prices are pulled from Yahoo Finance. For LSTM to predict stock price, stock historical daily data and macroeconomic indicators daily value are fetched from Yahoo Finance. And the technical indicators are calculated based on the definition and formula provided by financial websites such as Investopedia.

Model

Genetic Algorithm is used for finding an optimized portfolio. LSTM is used for predicting the future 10 workdays stock trend.

Output

The output of Genetic Algorithm is a list of proportions of the stocks in the optimized portfolio. The output of the LSTM model is the list of adjusted closing prices for the future 10 days.

Implementation Process

1. We will use the top 20 stocks from S&P in the past year from Yahoo Finance as our basic basket. The daily adjusted closing prices of stocks in the basket are fetched from the Yahoo Financial API and the database is being updated using a CRON job scheduler on a daily basis.
2. For portfolio recommendation of the day, we will let Genetic Algorithm run once a day using the latest daily data of 20 stocks and limiting the population size to 10. And the asset allocation of daily portfolio recommendation is stored in SQLite and updated daily.
3. For custom portfolio recommendation, we will let Genetic Algorithm run on the data of stocks selected by users.
4. For stock trend prediction, LSTM will be constructed to forecast future 10 days adjusted closing prices based on the data of the past 40 days. LSTM models are trained everyday based on the latest daily data and the predicted values are stored in SQLite and updated daily.
5. We will use React to develop the front end and Flask for the backend.
6. Finally, we will integrate all parts together and deploy the system to ngrok.

Proposed System Workflow

1. Users will see the home page showing the portfolio recommendation of the day when logging to our website.
2. Users can access the historical portfolio recommendation from the 'History' Page. Users can select the date from the calendar and the corresponding recommendation for stocks distribution for that day will be displayed.
3. Users can run the genetic algorithm on their choice of stocks from the 'Custom Portfolio' page. Users need to select the stocks interested. The server will output the sharpe ratio values and stock distribution for every epoch. The final best sharpe ratio and best stock distribution will be given when the GA algorithm terminates.
4. Users can also access the financial news from the 'News' webpage.

Appendix II: Mapped System Functionalities

Course	System Functionalities & Technique
Machine Reasoning	Data collection and storage: Stock prices and index values are fetched from Yahoo Finance. And the technical index calculating formulas are manually extracted from websites. The data were managed using SQL and stored using SQLite. Generate-Test approach: The system employs a generate-test approach. Portfolios are randomly generated and mutation/crossover applied to generate variants for testing. The testing is done against the fitness function (Sharpe ratio) and less fit portfolios are discarded. Data driven: The system is data driven as it makes use of past stock price data and other indicators to evaluate the Sharpe ratios and do price forecasting
Reasoning System	Genetic Algorithm: GA is used to find optimal portfolios
Cognitive System	Flask app and Ngrok for deployment.

Appendix III: Installation Guide

1. Getting Started

We offer 2 modes of access: a deployed version and an offline version. Do note that with the offline version, as the daily cron job is not run, the database will not be updated, and the latest stock price data is correct as of **28 Oct 2022**.

1.1 Access to the deployed version

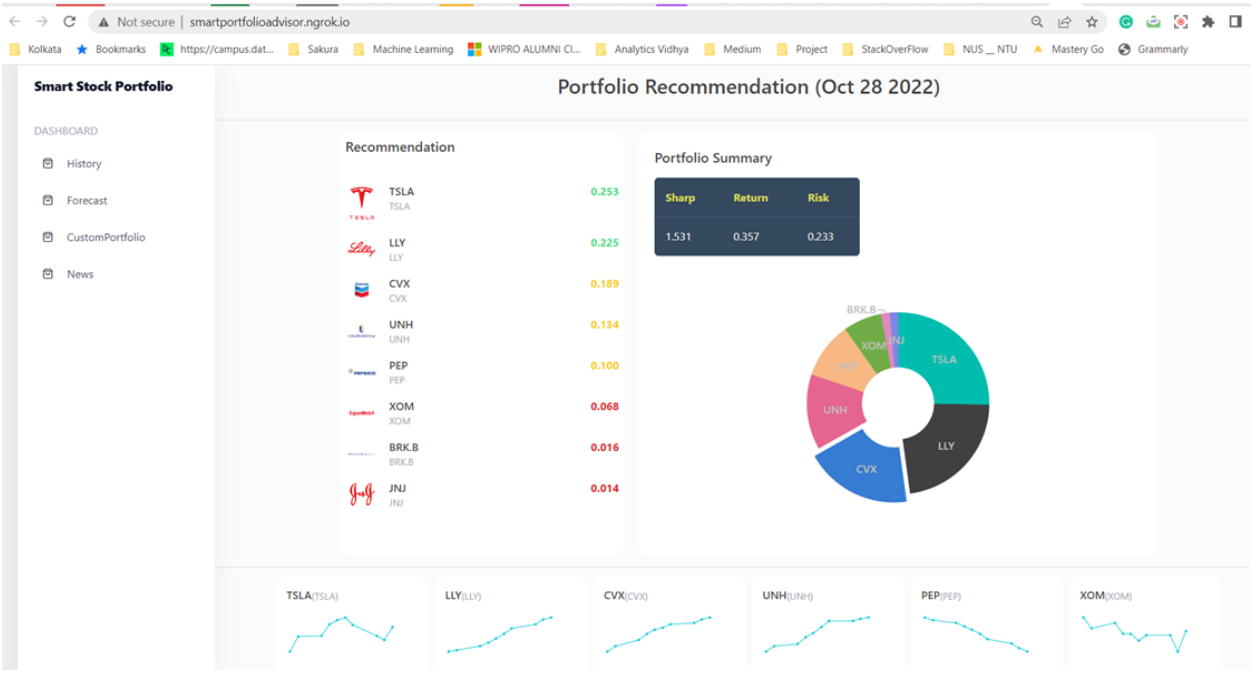
For a quick start, a version has been deployed to Digital Ocean. The application can be accessed from anywhere through the given URL.

Steps:

- Open any web browser (preferably chrome) and go to:

  http://smartportfolioadvisor.ngrok.io/

The application will be opened as shown in the image below



1.2 Full Installation

The user may want to deploy the application locally. The process and the pre-requisites are mentioned below.

1.2.1 System Requirements

- Python or Anaconda, and Python libraries as specified in requirements.txt
- A modern web browser. Recommended Google Chrome version 76 and above.
- NodeJS and NPM: Install using **Option 2** from https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-ubuntu-20-04
- Flask
- Git [Optional]: https://www.digitalocean.com/community/tutorials/how-to-install-git-on-ubuntu-20-04

DISCLAIMER: Smart Portfolio Adviser has only been tested on the following operating systems below

· Ubuntu 20.4: We need to use Linux based system for Flask backend as we parallelized the code and it will not run well on Windows.

1.2.2 Getting the code

- Clone the GitHub code:

  git clone https://github.com/NamrataThakur/SmartPortfolioAdvisor.git

- Alternatively, download the source code from the GitHub link directly if GIT is not installed

1.2.3 Installing Frontend

Once the code is downloaded in the system, please open the terminal, and enter the folder where the extracted project is present. After that, follow the below steps:

i. cd System Code/frontend

ii. sudo apt install npm

iii. npm install –legacy-peer-deps

1.2.4 Starting the frontend server

To start the frontend server, please follow the below steps:

i. npm start

Open another terminal:

i. cd System Code/frontend

ii. cd src/server

iii. npm run start

1.2.5 Installing Backend

Once the frontend is up and running, we need to install and start the backend.

i. sudo apt install python3-pip (If an error rises saying ‘pip’ is not present)

ii. pip install -r requirements.txt (To install all the packages needed)

iii. sudo apt install python3-flask (If flask installation from requirements.txt failed)

1.2.6 Starting the backend server

i. cd System Code/backend

ii. export FLASK_APP=app.py

iii. export FLASK_ENVIRONMENT=development

iv. flask run

With this the flask is started on **http://127.0.0.1:5000/** and the frontend will be started on **https://localhost:3000** and **https://localhost:3001**

1.2.7 System configuration

Deployment config for the web app needs to know where to access the database access program, and the flask server. This configuration is stored at SmartPortfolioAdvisor/System Code/frontend/src/config.js

Configuration Parameter	Meaning	Default (local hosting)
database.url	The ip:port address which the database access program is hosted at	localhost:3001
flask. url	The ip:port address which the database access program is hosted at	localhost: 3000

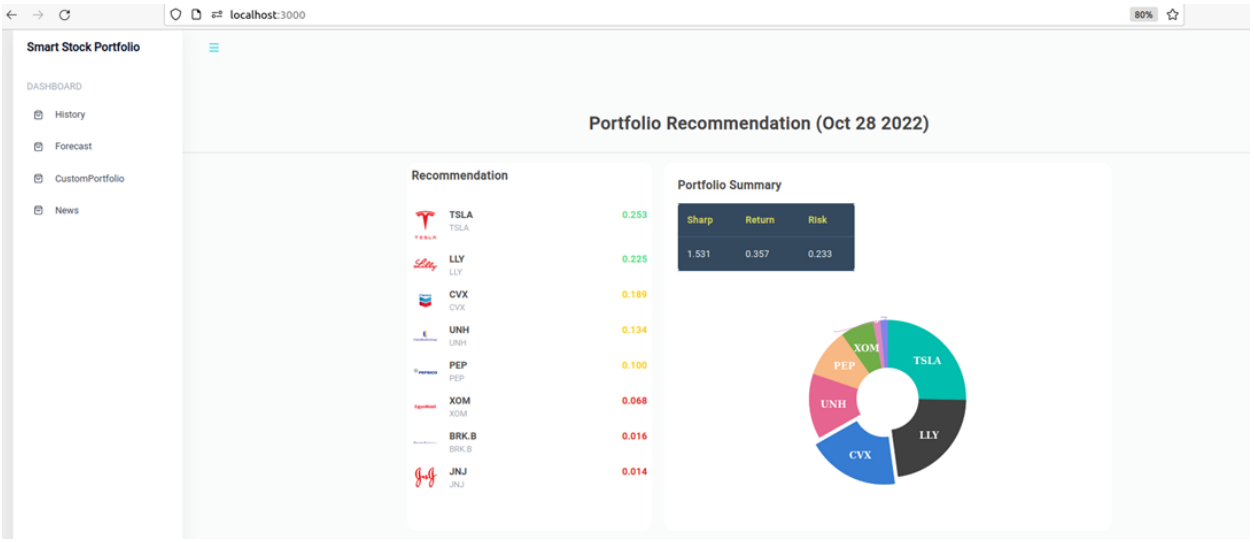
Appendix IV: User Guide

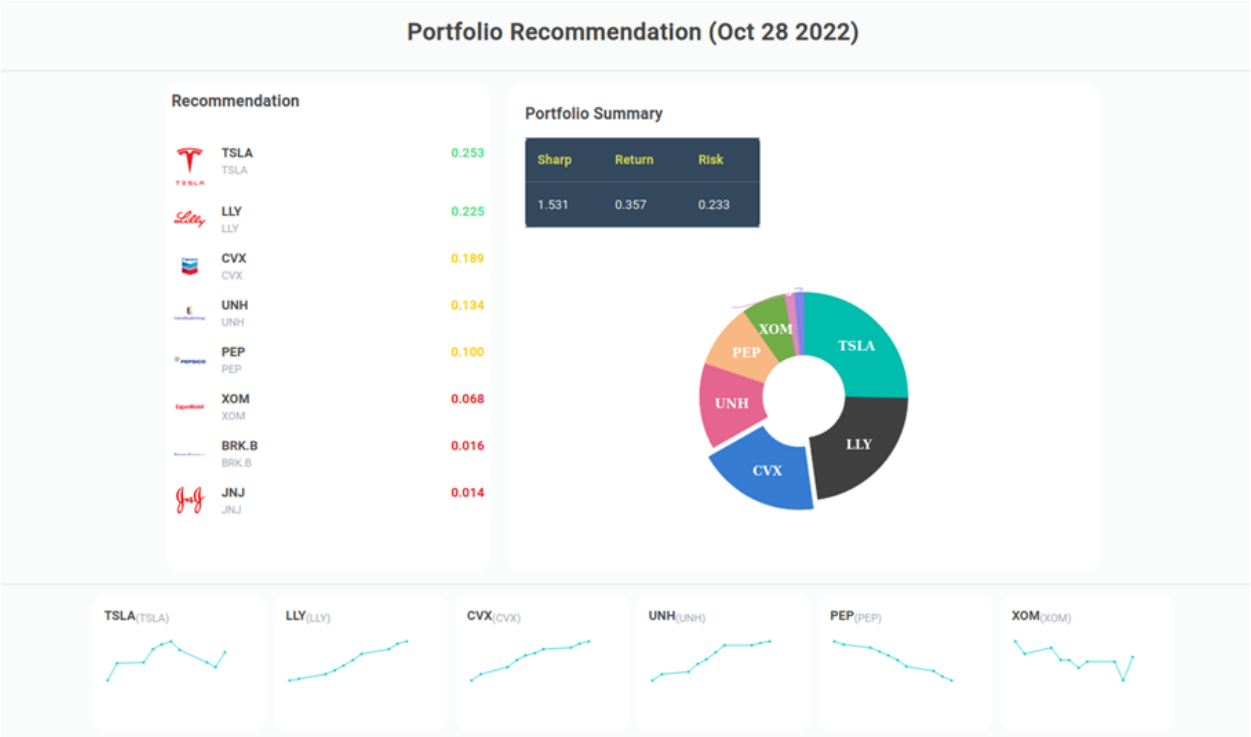
Follow the steps below to access and navigate the webpage:

- Go to **https://localhost:3000/**
- You would reach the Homepage

HomePage:

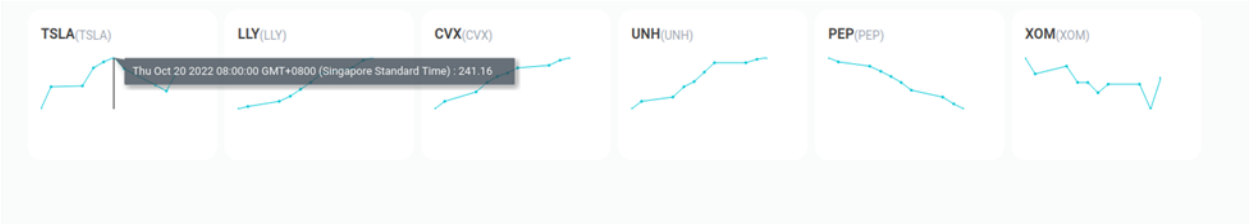
The best performing stocks are recommended. A portfolio summary is given that mentioned the best Sharpe Ratio value, Return Percentage and Risk Percentage.





The LSTM predictions for the best performing stocks are showed in the below section.

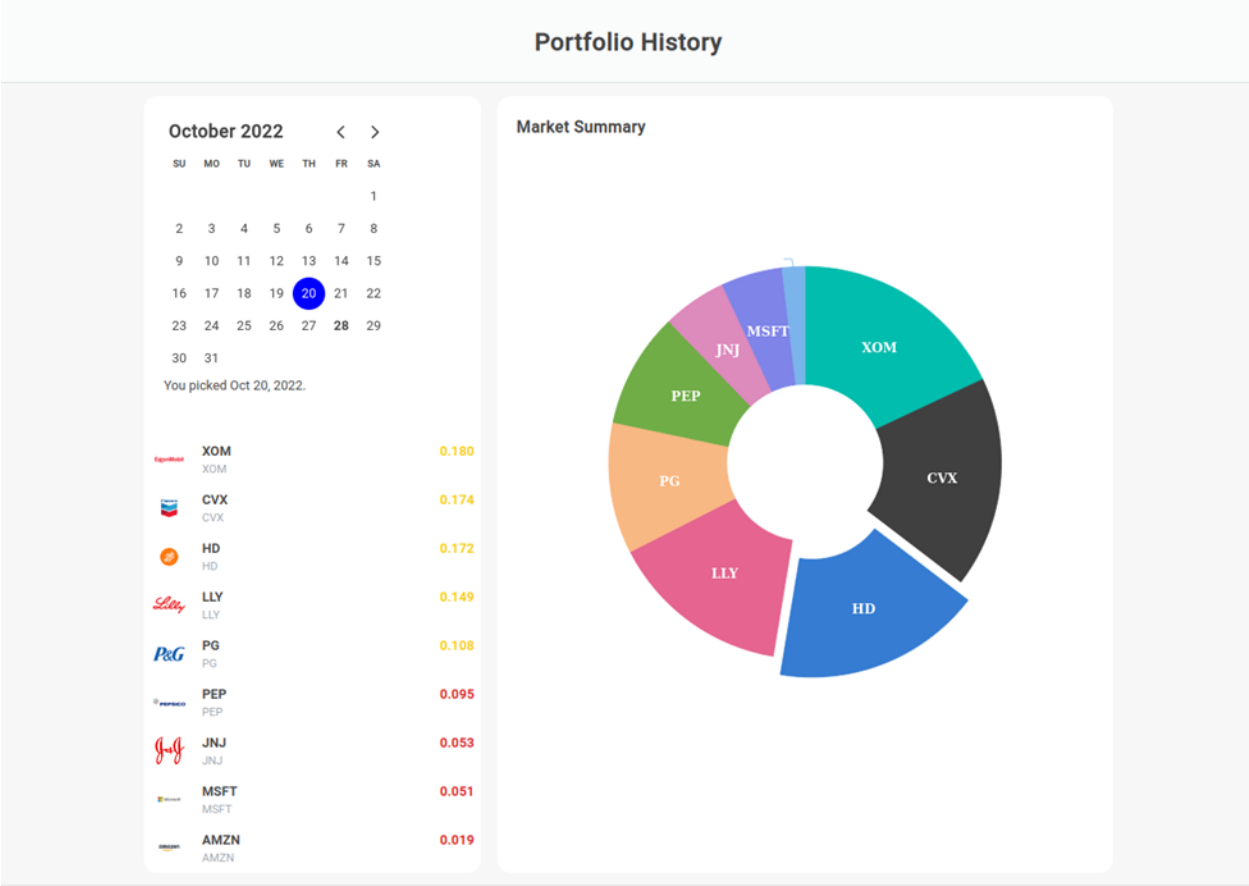
By hovering over the charts, you would see how the performance of the stocks changed day-over-day for the time period.



Hovering over the Pie chart shows the recommended distribution of the stocks.

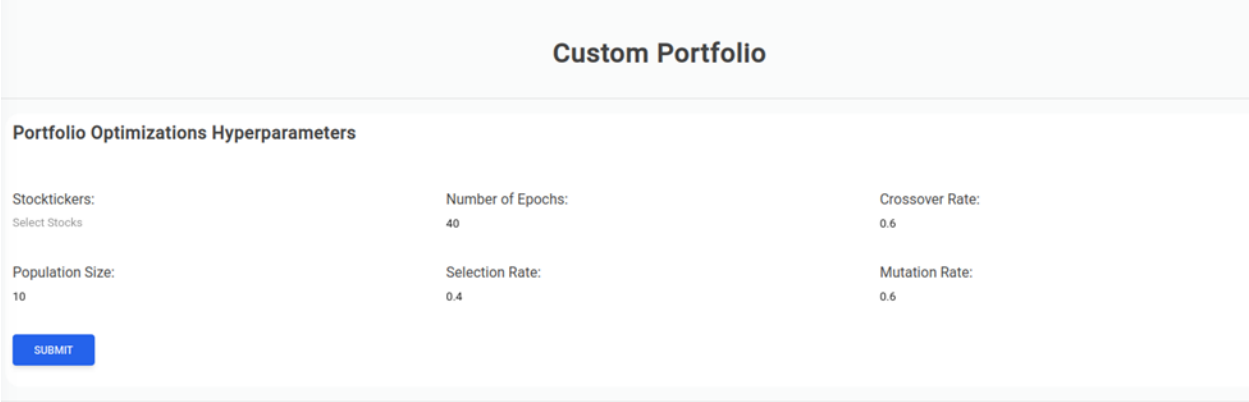
Portfolio History

You can access all the previous portfolio recommendation from the HISTORY tab of the left navigation bar. It shows a calendar and the recommended stock allocation distribution for the date selected.

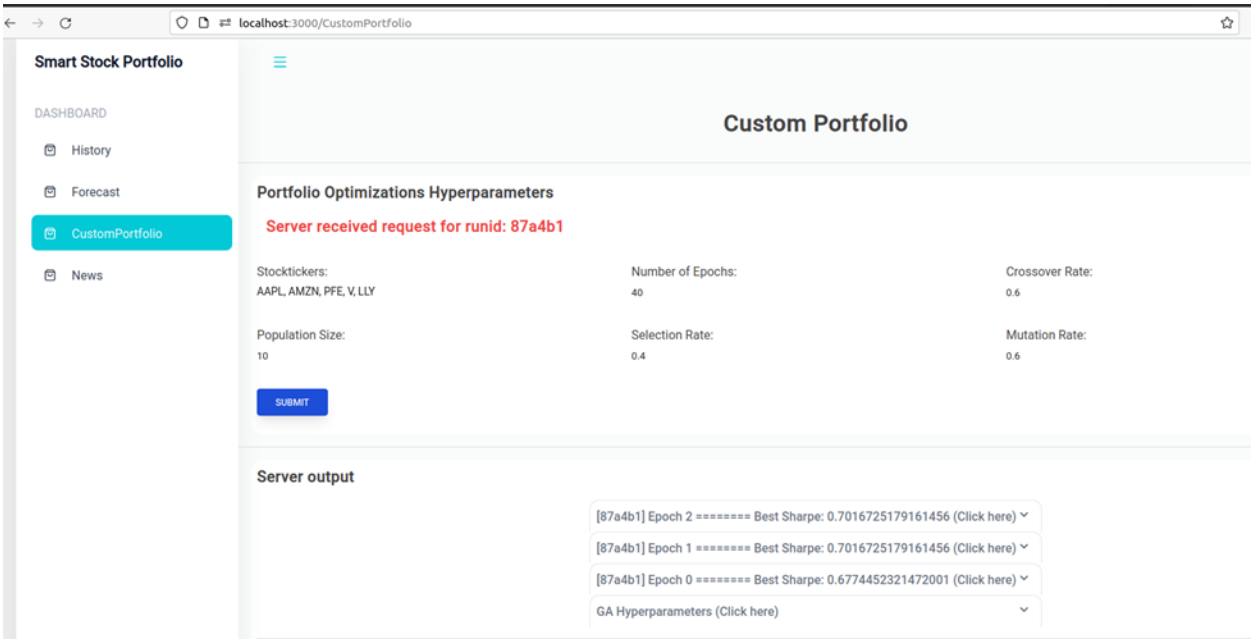


Custom Portfolio

The custom portfolio gives the user access to create a basket of stocks according to one's preference. The user also has the choice to update any of the training hyper-parameters to tweak the resultant Sharpe ratio.



Upon selecting the stocks and updating the hyper-parameters (if needed), user needs to press **SUBMIT**. A random **runId** will be generated for every request and the outputs for every epoch will start showing up in the page.



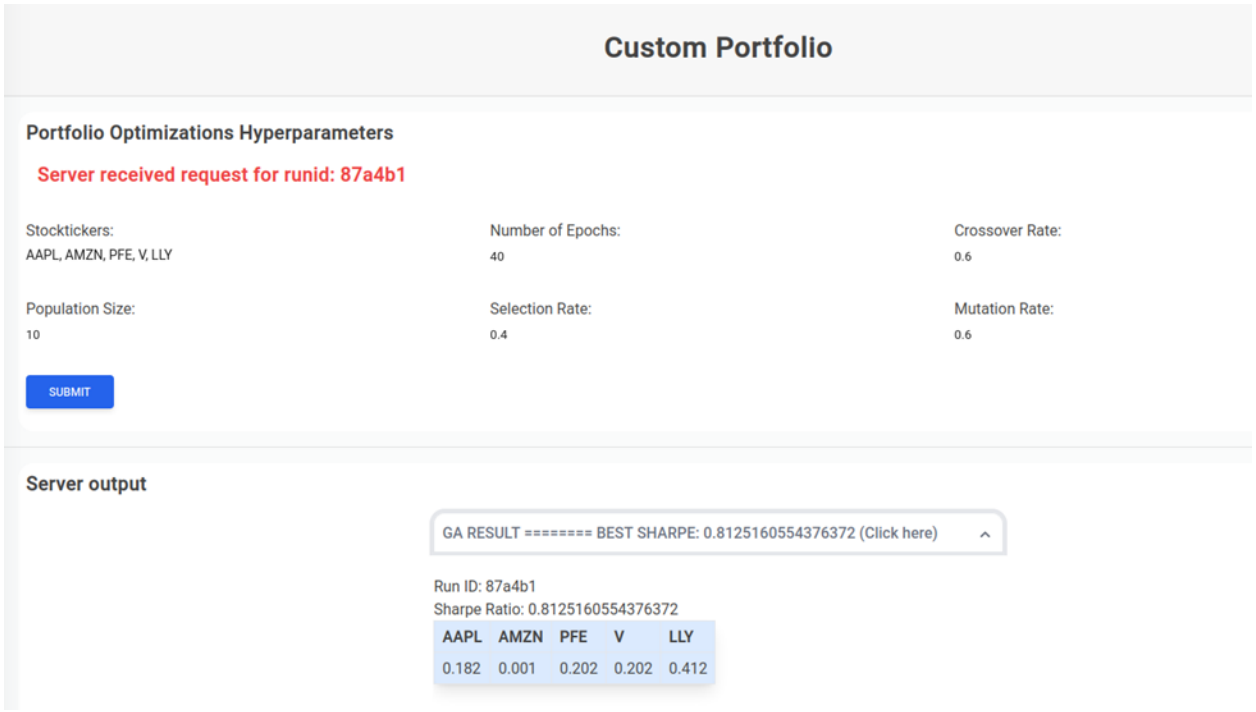
User can click on the '**CLICK HERE**' part of every epoch to see how for every epoch the Genetic Algorithm is performing and how the recommended stock distribution is changing.

[87a4b1] Epoch 19 ======== Best Sharpe: 0.7664180746581707 (Click here)

Run ID: 87a4b1
Epoch: 19

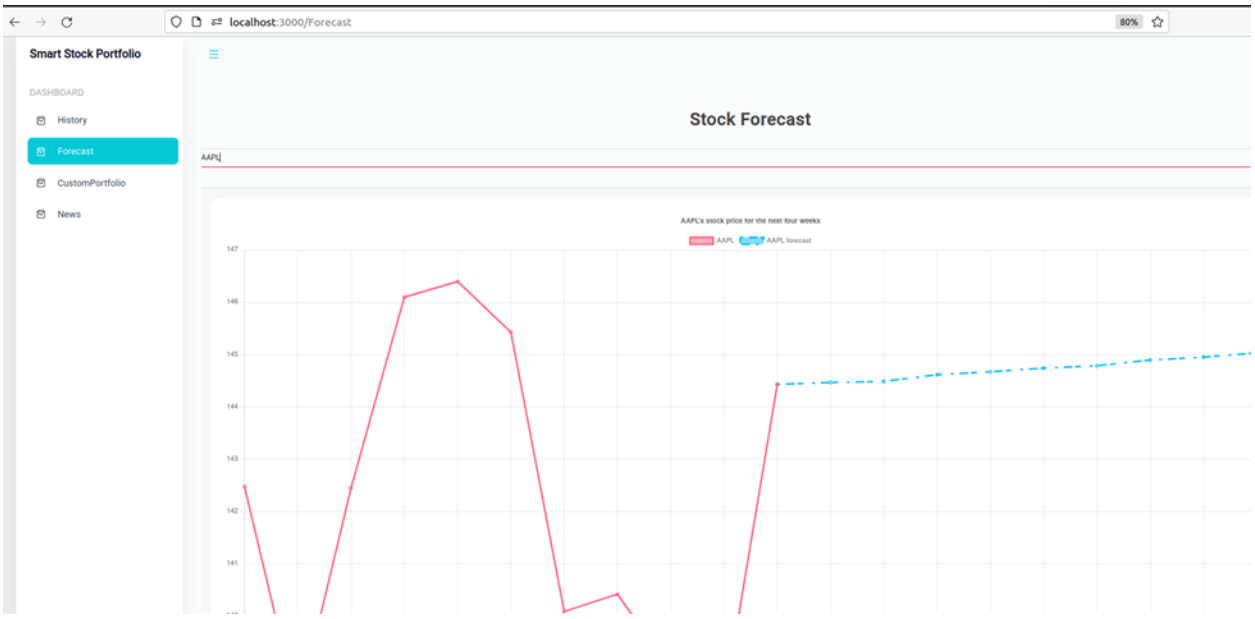
popid	sharpe	AAPL	AMZN	PFE	V	LLY
0	0.77	0.19	0.00	0.27	0.18	0.36
1	0.76	0.18	0.00	0.27	0.19	0.36
2	0.76	0.18	0.00	0.27	0.19	0.36
3	0.72	0.27	0.00	0.18	0.19	0.36
4	0.71	0.18	0.00	0.19	0.27	0.36
5	0.50	0.36	0.00	0.27	0.18	0.19
6	0.50	0.18	0.19	0.27	0.00	0.36
7	0.45	0.18	0.00	0.27	0.36	0.19
8	0.22	0.18	0.27	0.00	0.19	0.36
9	-0.26	0.19	0.36	0.27	0.18	0.00

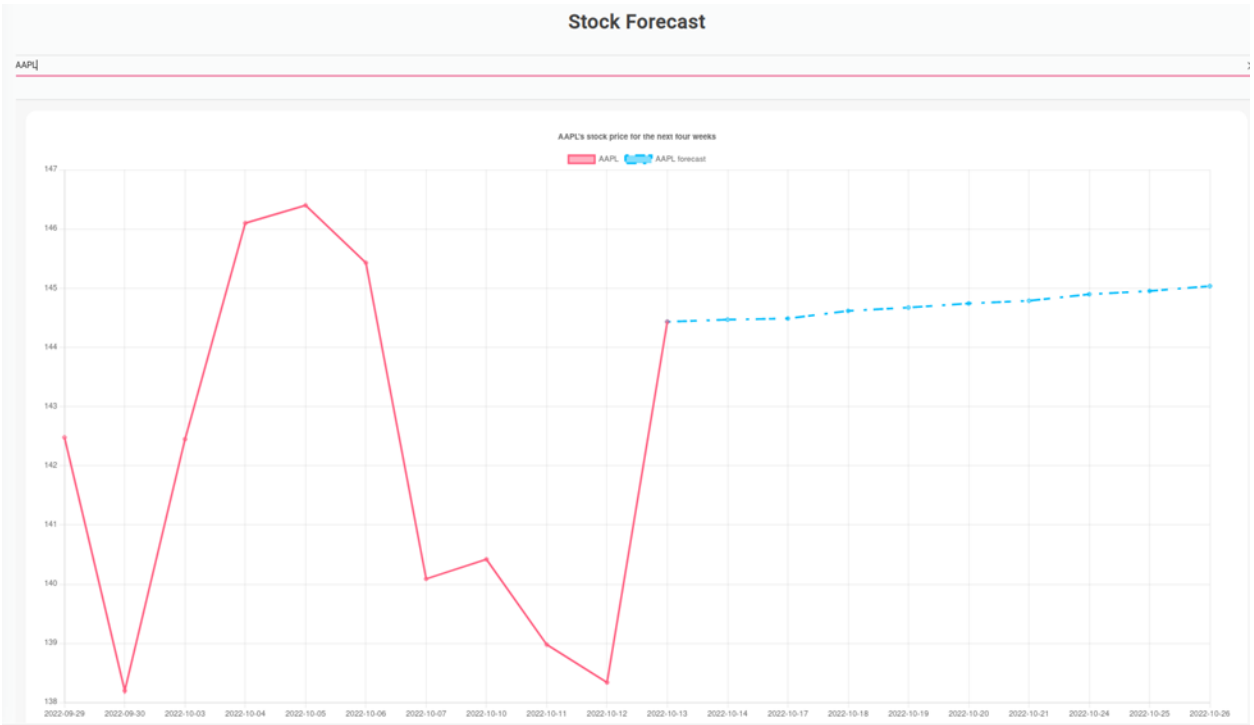
When the Genetic Algorithm reaches its termination stage, the **final GA Result** will be shown that will contain the overall **Best Sharpe ratio** and the **final stock allocation percentage**.


Custom Portfolio
Portfolio Optimizations Hyperparameters
Server received request for runid: 87a4b1
Stocktickers:
AAPL, AMZN, PFE, V, LLY
Number of Epochs:
40
Crossover Rate:
0.6
Population Size:
10
Selection Rate:
0.4
Mutation Rate:
0.6
SUBMIT
Server output
GA RESULT ======== BEST SHARPE: 0.8125160554376372 (Click here)
Run ID: 87a4b1
Sharpe Ratio: 0.8125160554376372
AAPL AMZN PFE V LLY
0.182 0.001 0.202 0.202 0.412

Forecast

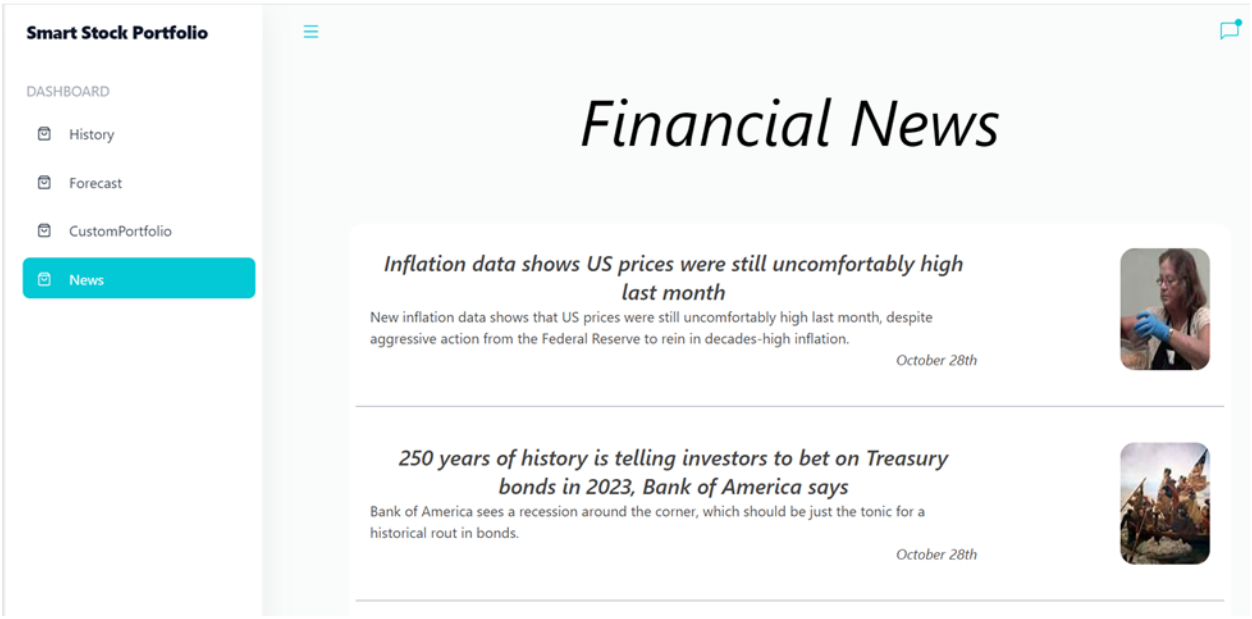
The forecast of the stocks can be accessed from the **'FORECAST'** tab of the left navigation bar. The user needs to mention the stock name for which the forecast is needed.





News

The user can access the financial news for the stocks present. The news will help the user to quickly assess the decision for choosing a particular stock in the basket.



Appendix V: Individual Report

1. Namrata Thakur

a. Personal Contribution

In this project, I worked as a backend developer. I designed the database update pipeline where the details of 20 stocks are fetched from the Yahoo Financials API daily. The latest date present in the Yahoo Financials is (today_date – 1). The column 'dailyReturn' is then calculated using the column 'close' for the (today_date – 1) and (today_date – 2). With the column value computed, the final dataset is then written to the database. Thus, the daily database update is happening. Once the database is updated, the daily GA run and daily LSTM model training begins. I created a Flask command to run the daily GA and LSTM. This command is designed to be fired from the command-line interface so that it doesn't mess with the Flask App running on 5000 port. I created a CRON job scheduler which will run this flask command at a particular time of the day. I also worked on creating the flask backend. I tried to deploy the application to Heroku for productionising it but faced a slug size error in Heroku. Due to time constraints, the team decided to move the deployment part to DigitalOcean where I contributed in rigorous testing and fixing the integration related issues. I also contributed in writing the project report.

b. Learning outcome

The biggest take-away from this project is to gain experience in building an application from end to end. I got familiar with development tools such ngrok, npm, Flask that would definitely help me to design and implement such pipelines in my future course of work.

Remote job scheduling is another aspect of my learning from the project. I now know how to create standalone flask commands and run them independently of the frontend. I also got quite extensive idea of deployment in the cloud. Genetic Algorithm is a concept that I learnt from the course in theory, but the project allowed me to gain an in-depth idea about how each part of GA (mutation, crossover, selection of elite population) works out.

Another skill I developed is how to manage a project and work with team members. The learning from them helped me to strengthen my knowledge in a variety of areas. Alongside, I also learnt how to use GitHub for version control.

c. Knowledge and Skill Application

I am confident that the skills that I gained from the project will help me to perform better at my next job. I learnt the ins and outs of Genetic Algorithm that is so vastly different from the supervised/unsupervised learning tasks that I used to perform in my previous job. I also gained a better understanding of the end-to-end cloud deployment, a skill that is highly needed in solving real world problems. Lastly, the experience of working in a team made me improve my inter-personal and communication skills.

2. Ouyang Hui

a. Personal Contribution

In this project, my main contribution was implementing the LSTM model to predict the stock future trend and writing the corresponding report. The tasks mainly included preliminary research, data collection, data preprocessing, LSTM model construction, hyperparameter tuning, model training and evaluation and so on.

Preliminary research included finding the business value and current solutions of predicting stock price using machine learning techniques as well as investigating the data we needed and data resources. Data collection involved using API to fetch data and manually extracting from websites. For the purpose of this project, a python module called yahoofinancials is used to fetch the stock price and relevant index value from Yahoo Finance. And also, the definition and formula about technical indicators are based on financial websites such as Investopedia. At the data preprocessing stage, I had done tasks such as technical index calculation, data normalization, feature selection, data cleaning and so on. After LSTM model construction and reshaping the data in order to fit the model, I tried to identify the optimal combination of different model architectures, learning rates and batch sizes by comparing their performance on training and validation data set. Then the final step I had done was to build a pipeline to train models for each stock in the stock basket and to create a database storing the forecasted stock price.

In addition to my main role, I was also responsible for documenting the details about LSTM for stock trend prediction and writing the project report. Moreover, I also contributed to fixing some front-end bugs and making the marking video.

b. Learning outcome

The most important thing I learnt from this project is the skills to build a system that utilizes artificial intelligence knowledge and techniques to solve real-world problems. During the project, I gained hands-on experience on designing and developing a pipeline to bring machine learning and deep learning use cases to production. For example, in order to build a LSTM for stock prediction, I broke the task down into several steps including data collection and preparation, training of the model, evaluation of the model, optimization, retraining and final deployment. From these experiences, I learnt how to map the complex objective to a sequence of steps and therefore improve the efficiency.

Moreover, I practiced how to use tools such as TensorFlow, NumPy, Pandas and so on for building machine learning and deep learning models. And also, I got familiar with web application development tools such ngrok, React, npm, Flask and so on.

Another valuable skill I developed during the project is about how to manage a project and work with team members. For example, I practiced how to use GitHub for version control and how to merge changes from others remotely. In addition, I realized that investigating the business case and market research before developing is also an important key to success of a project. Being clear about what problems we are facing can help in clearly defining the target and designing a suitable solution to address the problem.

c. Knowledge and Skill Application

Firstly, the skills I practiced in this project will be useful when I am doing software development jobs and AI related jobs in the future. The tools for developing and deploying a web application we used in this project is also widely used by most internet companies. And the AI knowledge and techniques involved in this project can also be applied in various industries in many use cases such as supply chain optimization, dynamic retail pricing, financial investment strategies, risk management and so on.

Next, during the project, I practiced how to construct a pipeline to build a LSTM model to predict stock trends from scratch. I can use these skills in any other use cases to address a problem from business value survey to final deployment.

Lastly, the experience of working in a team will help me be more confident in the future workplace when cooperating and communicating with team members. For example, I am more experienced in splitting a big objective into stages and limiting each stage in scope so that teammates can cooperate efficiently and projects can be managed with clarity.

3. See Jia Fong Grace

a. Personal Contribution

I researched, wrote and ran the Genetic Algorithm, including optimizing, backtesting, database logging and validation against the Efficient Frontier Hypothesis. I did the frontend/backend integration to create the form for the Custom GA in the react app, post it to the backend, run the background thread, and return the run results to frontend for display. I deployed the setup to DigitalOcean and created the technical video and contributed to the project report.

b. Learning outcome

I gained some financial knowledge on how to evaluate portfolios for their desirability, including how to compute Sharpe ratio and benchmark against the Efficient Frontier for that time period.

As I was relatively new to web programming and python, having only picked up python before the course and deep dive into web programming in the course of this project, I learnt how to code more effectively in python and javascript. I managed to do things such as pull data from external APIs, parallelize functions, create background threads, deploy Node and Flask applications, resolve cross origin errors, code for socket io communications in both javascript and python, add a deployment configuration for our webapp etc.

I also learnt how to create an Ubuntu VM in the cloud, install a graphical desktop on it, and deploy everything to the cloud. I learnt to configure ngrok to forward http requests to my application, and also learnt how to deploy applications under a custom domain name (http://www.smartportfolioadvisor.online:3000), although I eventually opted to use the ngrok.io domain name because the port number at the back looked weird.

c. Knowledge and Skill Application

The financial knowledge I learnt can be applied to my own investing journey to evaluate my own portfolio.

The rest of the technical skills which I gained can be put to practical use building, optimizing and deploying react apps, Flask apps etc. This project was a great hands-on experience for me. I really was able to combine and use many skills which I came across only briefly in my working experience and in my self learning journeys and create and deploy applications from scratch to end like a full stack engineer. I look forward to carrying the knowledge I

gained, e.g. programming for React applications, using Genetic Algorithm to solve problems, into my next job.

4. Wang Zhipeng

a. Personal Contribution

Throughout the project, I worked as backend developer and frontend developer. In the backend, at first, I use express.js to connect the frontend and the database, but later in order to easy call other function, I changed my backend framework to flask.

Backend received front-end requests for historical investment advice and requests for stock forecast prices, then obtain the corresponding data from the database, and finally send it to the front-end, and call the Yahoo API at the same time to obtain the corresponding company's trademark and display it on the frontend. And in the financial news part, I send a request to the backend, the backend calls the Microsoft news API and returns me the latest news about finance.

In the frontend, I completed about 95% of the program, I use react to build the frontend, I create several pages include Home Page, History Page, Custom Portfolio Page, News Page, Calendar Page, Todo List Page, Community Page, and Login Page. I use the component Route to navigator the different pages, use context component to pass data in different web pages. I use Chart.js and syncfusion component to draw different kinds of charts, includes bar chart, pie chart, line chart. I also use google API build the Login page, to let users can use their google account to login our system, and use ChatEngine and firebase to build the community page to let users discuss stocks.

b. Learning outcome

Throughout the project, I have learnt so many new things. First, I learned how to use flask to build the backend, because I was a full stack programmer, and I only know backend framework like express, Django, but never used flask framework. I also learned some component in frontend that I have never used. Syncfusion component is a good commercial software development tool, it can easily build very complex commercial component.

I also learned how to better use Github to build project with others, like create branch, invite other people to join the repository. I also learned how to write readme document.

I also learned how to use GA algorithm and LSTM to make the portfolio of stocks and the forecast of the stock. with the discussing with our group members and reading their code, I have a great understand of these algorithm and better know how to use these algorithm to solve real world problem.

Last but not least, I learned how to handle complex requests to the server, like build a thread pool to send different company logo request to the backend, and the mount lock to make sure visit the database security, and use group in the SQL statement to simplify the return function of the backend.

c. Knowledge and Skill Application

Recommender systems are among the most popular applications of data science today. Developing web application is the most useful skill to find a job in todays tech company. Our project combined both two useful skills to make a very commercial project, which has a very complete function including login, portfolio recommendation, custom's portfolio recommendation, and future stock trend forecast. A community like Facebook to let our user share their ideas.

With the LSTM and GA algorithm knowledge that I have gained, we build a LSTM module, using the sequence history stock price as the training data, to forecast the future stock price. And we