Individual Module Assignment 2

Matteo Sammut 0206002L

February 13, 2025

This documentation addresses the written portion of Assignment 2 for the Financial Engineering (ARI5122) submission. This document should be accessed alongside the code portion *A2Q1.ipynb*, *A2Q2.ipynb*, *A2Q3.ipynb* and *A2Q4.ipynb* for the respective questions.

1 Question 1

1.1 Part 1.1

The relevant tickers were downloaded consistently as they were for previous sections. The daily returns are obtained by utilising a function that provides the value difference from one entry to the previous one. Due to the first date not having a previous entry this returns a null value and is dropped. This methodology for retrieving data and calculating returns was carried over to subsequent sections.

1.2 Part 1.3

Stock Beta is a risk measure used to compare to volatility of a stock in relation to the market, in our case we are using the S&P 500 as our market returns. We can calculate the Beta volatility using the formula:

$$R_i = \beta R_m + \alpha \tag{1}$$

This is identical to the line formula y=mx+c, hence we can find beta by performing a best fit indicative of linear regression models. With a score of one indicating the same level of volatility as the market, if a value is higher than one this indicates increased volatility and hence excess risk and vice versa. Investors should make use of stock Betas so the associated risk in a stock is compensated by their perceived expected return.

As a practical example, we may look to the values obtained. Amazon has a beta of 1.09 (2d.p.) this signifies that amazon stocks are 109% as volatile as the s&P 500, so if our base market (S&P 500) has a return of +10% then the returns on Amazon's stock will rise by 10.9% and vice versa for a negative return. Boeing has a slightly lower beta indicating less volatility with General Electric having the highest. Since we are utilising a very large dataset with 10 years or data returns we obtain a very small p-value indicating our Beta is statistically significant. The confidence interval provides us with a range containing a lower or upper bound with a confidence of 95%, this signifies that we are 95% confidant that beta lies between these two figures. In our case no interval was below or equal to 1 and so all of our tickers are always more volatile than the market most of the time.

1.3 Part 2.1

Firstly we cleaned up and defined the constant vairables in line with the explicit expression for Geometric Brownian Motion:

$$S_t = S_0 \mathrm{e}^{(\mu - \frac{\sigma^2}{2})t + \sigma W_t} \tag{2}$$

μ and σ are the mean and standard deviation of the returns respectfully and were hence calculated the same as in previous questions based on the historical data. Our t is 1 as we are generating daily price movements hence require a daily time step. S_0 represents the initial stock price and will be our divergance point, in our case this would be the last observed price as we want to simulate from that

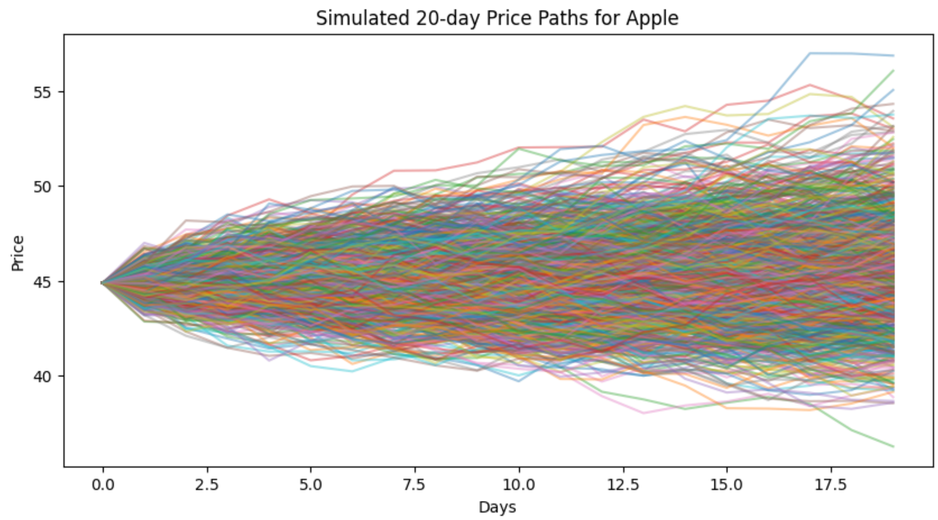

Figure 1: Range of potential future Apple price paths

point onwards, so we use the last closing price in our array. Finally we randomly sample from the daily normal distribution also known as a Brownian Motion, this introduces randomness into our model.

Next, we need a structure to store our simulated paths. To achieve this, we create a blank 2D array where each row represents a single simulation. Since each simulation spans 20 days, each row contains 20 elements with each element based on the previous one. We then generate 1,000 of these simulation arrays, resulting in a final shape of (20 × 1000).

Our price realizations were graphed in figure 1 naturally it is very dense as there are 1000 lines plotted. Each line representing a trajectory that the Apple price may go down.

1.4 Part 2.2

We calculate this using the following formula:

$$returns = \frac{final\ simulated\ price - initial\ price}{initial\ price} \tag{3}$$

The significance of this value represents the worst expected loss within the given confidence interval, based on our simulations there exists a 5% change by which Apple's stock will drop by 8.51% over the next 20 days.

This helps investors to prepare for a worst-case loss given a market downturn. Apple investors could opt to compare this figure to other stocks to observe its volatility.

2 Question 2

2.1 Part 3.1

To compare the performance of the portfolio optimizations we tabulated their portfolio weights and performance metrics in table 1, it should be noted that for our neural network portfolio optimization we opted to implement the alternate implementation and directly optimize for the portfolio weights by maximizing the Sharpe ratio.

2.2 Part 3.2

After tabulating the results into table 1, one thing is evident, the neural network was able to drastically improve the shape ratio. This suggests that the Neural Network found a portfolio with much lower volatility while maintaining a similar level of expected return. Our choice to maximize for the Sharpe ratio inversely reduced the volatility, this property is inherent from the formula in eq 4. We found

Metric	Markowitz Mean-Variance	Neural Network
Optimized Portfolio Weights	AAPL: 0.5813 GOOGL: 0.0 MSFT: 0.4187	AAPL: 0.7262 GOOGL: 0.0 MSFT: 0.2738
Expected Annual Return	34.47%	34.91%
Expected Annual Volatility	29.27%	1.88%
Sharpe Ratio	1.18	17.50

Table 1: Comparison of Portfolio Optimization Methods

this optimization to be more ideal, as the proportionality to the expected return resulted in a stable expected return while reducing the volatility.

$$\text{Sharpe Ratio} = \frac{\text{Expected Return} - \text{Risk-Free Rate}}{\text{Volatility}} \tag{4}$$

In this portfolio optimization, we relied on historical returns data for the selected tickers. However, this approach presents several challenges. One key limitation is the potential lack of generalization to different market conditions, as patterns observed in historical data may not persist in the future. This can lead to overfitting, where the model captures noise rather than meaningful trends. To mitigate this risk, we may employ techniques such as train-test splits and regularization to ensure the model's performance is evaluated on unseen data. Furthermore, survivorship bias or selection bias—can distort model outcomes, emphasizing our need for careful data preprocessing and validation. The performance of these models is always directly related to the quality of data provided and is a common pitfall when creating new models.

Training time reduces the portability of the neural network. Neural networks are black box models, hence the reasoning for the weight allocations may not be entirely known; this is not enticing for risk averse investors. Additionally, the training process is inherently stochastic, retraining the model with the same hyper parameters produces different results. Therefore, for stable and simpler markets, a mean-variance optimization approach may be more suitable for the environment.

3 Question 3

3.1 Part 1.2

For this comparison task we obtained the closing price data for the S&P 500 between 2019 and 2023 to serve as a baseline, calculating the daily returns. We also enforced a 10 day window size as a common constant for testing. Like this we could ensure reliable comparisons to be made between our models and the observed actual volatility.

The value to find is the volatility of the stock at the tenth day, hence our y values consists of the volatility at this day. Therefore, we set our X to be the daily returns leading up to the windowed period. These arrays were proportioned into an 80/20 train/test split for further use in the neural network section.

Using an appropriate package we hence fit a GARCH model to the returns and stored the results to be graphed again the baseline as shown in figure 2. We opted to graph these separately over the true returns as we believe it increases the legibility of our results.

3.2 Part 1.3

We calculated the mean square error of the results by making use of the predicted and observed volatility. This came up to a very small mean squared error of *0.000006*, this indicates that the squared differences between our model's prediction and the actual observed values are very small. This is quite observable in the respective graph.

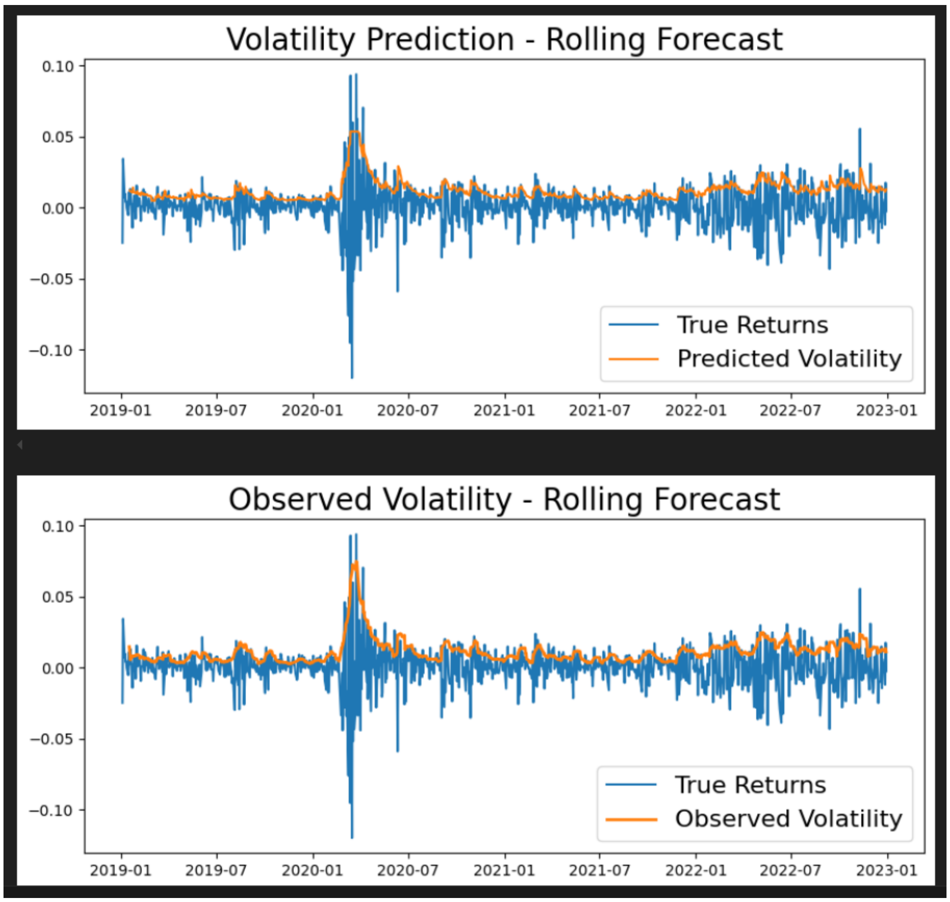

Figure 2: GARCH predicted volatility vs observed baseline shown over the true returns

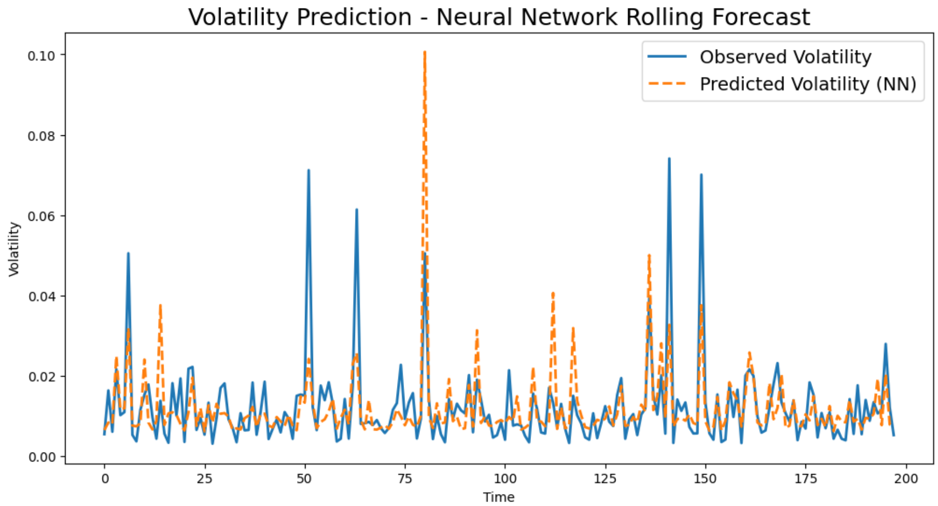

Figure 3: Neural network performance on test set

3.3 Part 2.1

For this task we carried over our data preparation from the previous part to ensure variable continuity, making use of an 80/20 training/test set.

3.4 Part 2.2

The model was defined as suggested with a two hidden layer setup. The network structure can have different configurations and obtain similar results, this is true for all layers more or less with explicit exception to the output layer. As our target, volatility, is a single continues value with need to use a single neuron layer with linear activation. The output layer should always be reflective of the target value.

We opted for $ReLU$ activation for all other layers, its simplicity and efficiency has made it especially popular for shallow neural networks such as this one. It is able to introduce non-linearity without compromising on performance due to multiple complex calculations such as in *tanh*. $ReLU$ activation can result in $ReLU$ neuron death, where neurons become inactive if they consistently receive negative inputs. However, for lightweight models like this, it is less of a concern. We could have opted to include some level of neuron dropout, where some neurons are reset negating this possibility, but given it being such a non-issue we opted to ignore it.

Apart from this the two hidden layer setup was done in a dense configuration, meaning each neuron takes all the previous neurons as input, with a 32 neuron count for both. It is preferred in a dense configuration that the neuron counts match. We made use of the adam optimizer, its ability to adjust its learning rate for each parameter basen on gradients made it ideal to handle volatile patterns, such as in time series forecasting tasks. As our performance metric is the mean squared error we made this our loss function.

We validated the ability for the model to predict new data by making use of a test set. We calculated the mean squared error on this test set to be *0.000091* and graphed the results into graph 3. We were quite satisfied with this result and proceeded to apply this on the whole timeline to compare with the GARCH model.

We graphed the volatility predictions on the same graph as the previous section in figure 4. The result of this graph is quite similar to that observed by the GARCH and baseline.

3.5 Part 3

Despite obtaining similar results, a limitation in the GARCH model was observed when extending the period. As shown in figure 5, the GARCH(1,1) model appears to be overestimating volatility after the 2020 COVID spike, causing extreme spiking. This is apparently a common issue when there is a

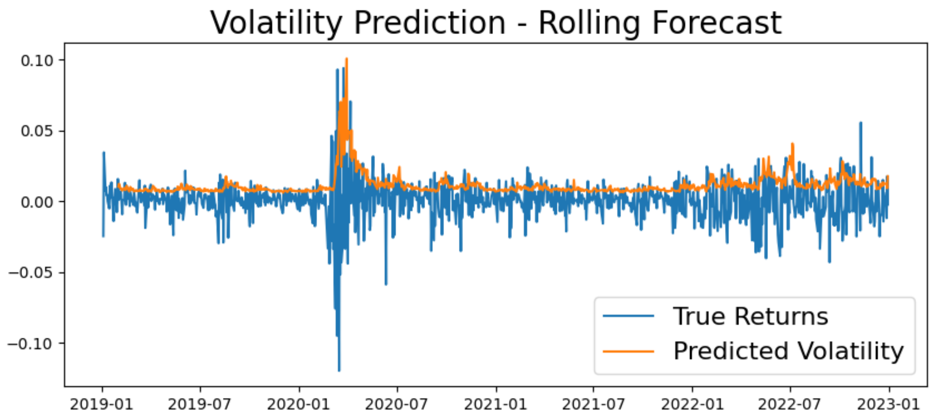

Figure 4: Neural network performance on returns

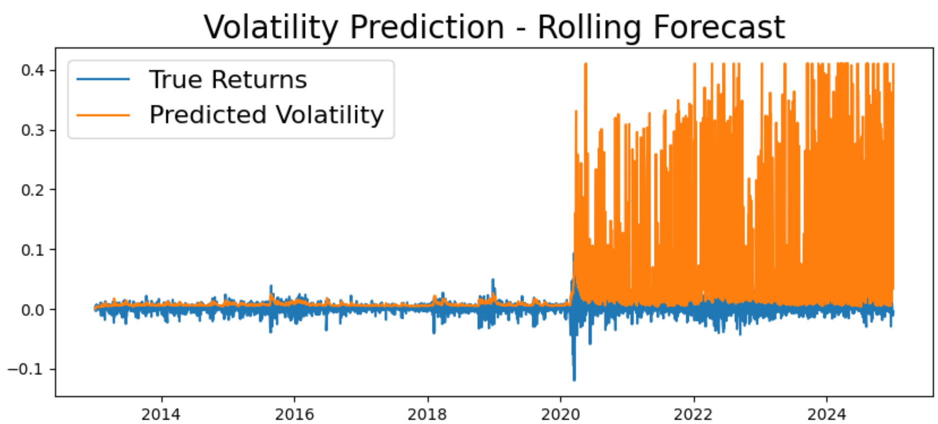

Figure 5: Garch performance when extending the period, clipped past 0.4 to graph

structural break or persistent high volatility in the data. We were abler to resolve this issue by using a more stable GARCH variant such as GARCH (3,0), naturally we omitted its inclusion as is not in the scope for this question. Significant changes in the data may also affect neural networks if not trained for appropriately, perhaps not to the error of the GARCH model but similar wrong predictions may be observed.

Similarly we could have performed similar hyper parameter optimization for both, to find the ideal neuron and layer count. The activation function and GARCH variants like (3,0) and EGARCH, which models volatility in logarithmic space reducing extreme jumps, could have similarly been introduced.

4 Question 4

4.1 Part 1.1

We obtained the required end of month stock data for the past 60 months leading to 31/12/2018. Frustratingly, setting a month interval in the yfinance python package results in closing prices for the first day of the month. After getting the daily closing prices we filtered out the ones at the end of the month and localized it to remove the hour data to match the data provided by the *famaFrench3Factor* package.

4.2 Part 1.2

No significant issues were encountered during the calculations of this process. We combined the price data with that of the FAMA French to obtain a new pandas dataset.

4.3 Part 1.3

Metric	**CAPM**	**FF3**	**FF5**
Alpha	N/A	0.0103	0.0103
Alpha p-value	N/A	0.2057	0.2057
Beta (Market)	0.0123	1.6532	1.6532
Beta (SMB)	N/A	-0.5466	-0.7994
Beta (HML)	N/A	-1.2961	-0.6068
Beta (RMW)	N/A	N/A	-0.7298
Beta (CMA)	N/A	N/A	-1.3922

Table 2: Comparison of CAPM, Fama-French 3-Factor, and Fama-French 5-Factor Models

The results of the CAPM, Fama-French 3-Factor, and Fama-French 5-Factor models were tabulated into table 2, we used these results to reveal distinct insights into Amazon's risk exposures. The CAPM, with a market beta of *1.6048*, indicates substantial sensitivity to market movements, suggesting that Amazon's returns tend to move more than proportionally with the overall market. The model predicts an expected return of *0.0147*, derived from the market premium and the risk-free rate. However, CAPM's simplicity, relying solely on market risk, overlooks other firm-specific characteristics that may influence returns, limiting the model's explanatory power for a company like Amazon, which operates in a dynamic and growth-oriented industry.

Hence to counteract these discrepancies we MADE use of Fama-French models models, by incorporating these dimensions, capture the firm's complex risk profile more effectively. Our implementation of Fama-French3 resulted in a market beta of *1.6532*, reaffirming Amazon's strong market sensitivity. The negative SMB coefficient of *-0.5466* suggests that Amazon behaves similarly to a large-cap stock, while the HML coefficient of *-1.2961* indicates a pronounced growth orientation, this could be indicative with the company's aggressive growth strategy.

Building off this, the Fama-French 5 model extends off Fama-French 3 by introducing two additional factors, profitability (RMW) and investment (CMA) factors respectively. The negative RMW value of *-0.7298* implies lower profitability relative to firms with robust positive earnings, while the negative CMA of *-1.3922* reflects Amazon's tendency to reinvest aggressively rather than maintain conservative investment practices. These additional factors help explain return variations that the 3-Factor model overlooks.

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Declaration

Plagiarism is defined as "the unacknowledged use, as one's own, of work of another person, whether or not such work has been published, and as may be further elaborated in Faculty or University guidelines" (University Assessment Regulations, 2009, Regulation 39 (b)(i), University of Malta).

I / We*, the undersigned, declare that the [assignment / Assigned Practical Task report / Final Year Project report] submitted is my / our* work, except where acknowledged and referenced.

I / We* understand that the penalties for committing a breach of the regulations include loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected, and will be given zero marks.

* Delete as appropriate.

(N. B. If the assignment is meant to be submitted anonymously, please sign this form and submit it to the Departmental Officer separately from the assignment).

Matteo Sammut

Student Name — Signature

Student Name — Signature

Student Name — Signature

Student Name — Signature

ARI5122

Course Code

Assignment 2

Title of work submitted

27/01/2025

Date