COMP6212: Computational Finance

Finance Coursework

Charles Powell (29970245) Benjamin Sanati (30880505) Oran Bramble (30744539)

Introduction

This report presents the implementation methodology and findings of investigations into three computational finance concepts; time series analysis, algorithmic trading and portfolio optimisation. For each part, the implementation and testing approaches are described and justified, and the results analysed. Note that part four is not included in this report, as per the information received from the module team for groups of three.

1 Time Series Analysis

1.1 Implementing ARIMA(p, d, q)

To compute the ARIMA model of a time series, the `compute_arima()` method is defined, which takes in the `time_series` to be modelled, lists of weights for `p` and `q`, the value of `d`, and the number of `steps_ahead` to forecast.

The implementation details of the `compute_arima` method are based on those provided in the time series lectures. First, the method applies the required number of rounds of differencing to the to the time series based on the parameter `d`, and stores a record of each round of differencing to ensure the predictions can be properly un-differenced at the end of the process. To avoid bias, the first entry in each difference list is assigned a value of `np.nan`, rather than 0. The method then iterates through the `time_series`, and at each time step will make a prediction of the difference by computing the summation of the auto-regressive (AR) and moving average (MA) components based on the parameters `p` and `q`. The error between the predicted and true difference is then recorded to be used by future MA components, and the true prediction is determined and stored by un-differencing the value. After iterating through the time series and the required number of steps ahead, the list of predictions for the time series is returned.

When considering the start of the time series, no predictions are made until there are enough previous time steps in the difference list to contribute to the entirety of the AR component. Furthermore, when not enough predictions have been made to contribute to part of the MA component, an error of 0 is assumed for this component.

When considering predictions made for time steps greater than one ahead of the time series, the method feeds its previous predictions back into the system as the true values, and uses the current mean error in predictions for all future MA components.

1.2 Testing ARIMA$(2, 1, 1)$

When used on the provided time series with values `p=[0.2, 0.4]`, `d=1`, `q=[0.5]` and `steps_ahead=2`, the model produced the following results. These parameter values are based off of the last three digits of the student ID '29970245'. Note that the first three predicted values are `np.nan`, as the model is not able to make predictions for these points due to a lack of prior points in the time series.

$$Predictions = [nan, nan, nan, 10.4, 22.1, 30.05, 22.75, 24.15]$$

$$\hat{y}_7 = 22.75$$

$$\hat{y}_8 = 24.15$$

1.3 Testing ARIMA(p, d, q) on Stock Data

1.3.1 Fitting The Model

To fit an ARIMA model to a time series, the `train_arima()` method is defined, which takes in the `training_time_series` the model should be fit to, the range of ARIMA hyperparameters to explore as lists `p_range`, `d_range` and `q_range`, a `learning_rate` for the training process and the number of `epochs` the training should run for.

The `train_arima` method works by first defining a search space of all possible ARIMA models based on the provided ranges of values for `p`, `d` and `q`. Then, for each configuration, random weights are initialised and gradient descent is used to find the optimal values. This approach uses the `compute_arima` method defined earlier to model the time series, and uses the mean absolute percentage error (MAPE) of these predictions as the loss function. For simplicity, finite differences are used to approximate the derivatives of the weights with respect to the loss function, as opposed to an exact solution.

For each configuration and weight assignment, the performance of the model is tested, and if its MAPE is better than the current best, it is appended to a history of the loss, and its configuration is recorded as the new best. After performing gradient descent across all possible configurations the optimal values and weights for p d and q are returned, along with the history of MAPE during the training process.

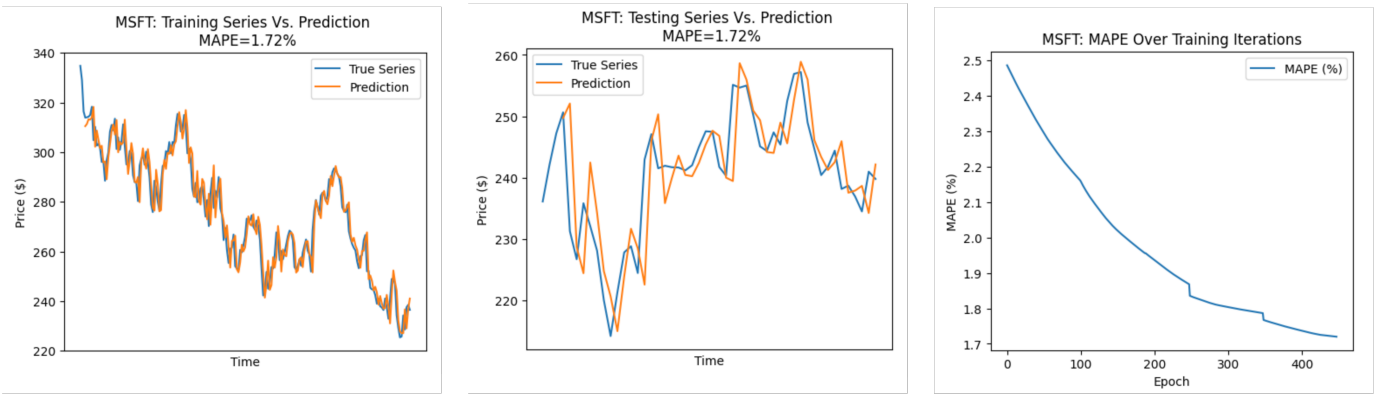

Fig. 1: Fitting an ARIMA model to MSFT

1.3.2 Results

To test the approach defined above, price data was downloaded for a number of prominent stocks including Microsoft (MSFT), Apple (APPL) and Tesco (TSCO) over the year 2022 and divided into a training and testing series according to an 80:20 split. In each case, the training series, along with `[1,2,3]` as the range of values for `p`, `d` and `q` were then passed to the `train_arima()` method to fit an ARIMA model to the data. For each configuration, training was run for 100 epochs at a learning rate of 0.01. Finally, the returned model was used on the testing series and its performance analysed according to the MAPE.

In all cases, the method was able to fit a model to the training data with a high-level of accuracy, and achieve similar performance on the testing series, with the MAPE typically ranging from 1.6-2.0%. Fig. 1 showcases example plots for the ARIMA model fitting to the training and testing data in the case of MSFT, as well as the loss curve during the training. Note that the loss curve is not smooth due to the performance of the 'jumping' as the values of `p`, `d` and `q` change.

2 Algorithmic Trading

2.1 Selecting Suitable Stocks

To determine the most suitable stocks for pairs trading, pricing information was downloaded for all FTSE100 stocks from 01/01/2018 to 01/01/2023 via Yahoo Finance [1]. In order for the stocks to be compared accurately, this list was then filtered to contain only those with pricing information across the entire time period. This approach was taken for simplicity, and in favour of more complex methods such as using filler values or including stocks which only have a small number of missing data points. With this refined list of stocks, the most correlated pairs of stocks were selected based on Pearson's Correlation Co-efficient. For each of these pairs, their relative price change, ratio in price, and spread in price were considered to determine the most suitable choice for pairs trading.

Based on this process, Tesco (TSCO) and Pershing Square Holdings Ltd (PSH) were selected for the following reasons:

- **High Correlation**: The stocks are highly positively correlated meaning that they both show similar patterns of price increase. This correlation makes it possible to profit off of a baseline long position in both stocks when no pairs trading criteria are met.
- **Divergence Followed By Convergence**: While both stocks show a general pattern of price increase, their relative price movements follow a pattern of repeated divergence and convergence. This pattern can be capitalised on by taking opposing long and short positions in the stock when they diverge, and exiting this position upon their convergence. This strategy operates under the assumption that due to their high-positive correlation, when the stocks diverge in price, they will eventually re-converge, and offers increased profit compared to the baseline.
- **Augmented Dickey-Fuller Test**: The price ratio for the stocks passes the Augmented Dickey-Fuller Test, making it a suitable tool for pairs trading.

Fig. 2 illustrates this analysis and justifies the selection of TSCO and PSH by showcasing the correlation matrix for the refined list of stocks, the relative price change between TSCO and PSH, and the z-score for the ratio in price between the two.

2.2 Trading Strategies

Following the selection of TSCO and PSH for pairs trading, two trading strategies were then implemented. Both of these strategies operate on a principle of mean reversion based on the price ratio of the stocks, but make use of different statistical properties to define their entry and exit criteria.

The general approach for both strategies is to hold long positions in both stocks as a baseline, but enter opposing long and short positions in the stocks during times when their price ratio deviates from its mean statistically. Holding long positions in both stocks as a baseline allows for the strategies to profit in the cases where both stocks are increasing in price, while holding opposing long and short positions ensures low-risk profit in the case of price divergence. In all cases,

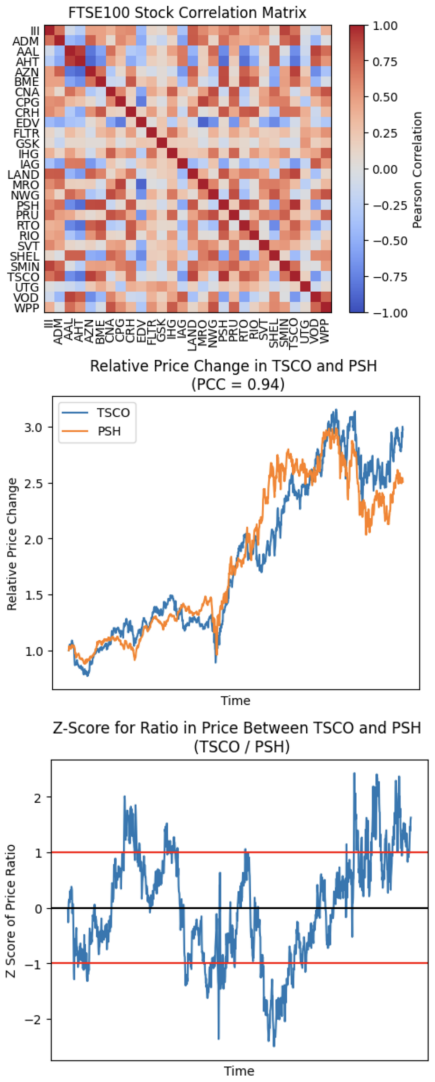

Fig. 2: Pairs Trading Stock Selection

whenever a trade is made, half of the available of money is invested into each position.

The first strategy monitors the z-score of the stocks price ratio directly. The strategy will maintain a long position in both stocks until this z-score falls outside of some threshold distance, at which point it enters opposing long and short positions on the stocks. When the z-score falls back within the acceptable range, the strategy reverts back to holding long positions in both stocks.

The second strategy monitors the short- and long-term simple moving averages (SMA) of the z-score of the price ratio. The strategy will maintain a long position in both stocks until the short-term SMA differences the long-term SMA by some threshold amount, at which point it enters opposing long and short positions on the stocks. When the short term SMA falls back within the acceptable range, the strategy reverts to holding long positions in both stocks.

2.3 Results

To test the efficacy of the defined strategies, back-testing was performed on the TSCO-PSH stock pair from the time period of 01/01/2018 to 01/01/2023, and comparisons were made to a baseline strategy of simply holding long positions in both stocks. For each strategy, the first half of the time period was assumed as prior knowledge, and the remaining dates were iterated through, with the strategy adding the pricing information to its knowledge, and making trades based on its criteria. Throughout this back-testing, the trades and profit made by each strategy were recorded.

Table 1 provides a comparison of key statistics between the baseline approach and each of the trading strategies, while Fig. 3 shows a comparison of the profit made by the baseline and each of the strategies throughout the duration. Both strategies achieved very similar performance and both outperformed the baseline approach. The strategies differed, however, in that strategy one made large amounts of profit off of a small number of trades, where as strategy two made small amounts of profit off of a small number of trades.

Baseline	
Stock 1 Baseline Profit	68.189%
Stock 2 Baseline Profit	43.867%
Average Baseline Profit	56.028%
Strategy 1	
Total Profit	73.213%
Total number of trades	29
Mean length of trade	9.828 Days
Total time spent in trade	285 Days (45.3%)
Mean trade profit	3.314%
Strategy 2	
Total Profit	69.550%
Total number of trades	249
Mean Length of trade	1.522 Days
Total time spent in trade	379 Days (60.254%)
Mean Trade profit	0.259%

Table 1. TSCO-PSH Pairs Trading Statistics

Fig. 3: TSCO-PSH Pairs Trading Results

3 Portfolio Optimisation

3.1 Selecting Suitable Stocks

For the portfolio optimisation task, 5 stocks (**AutoTrader, Experian, Rightmove, Rolls Royce and Shell**) were selected at random from the FTSE 100 index. The data, over a 1-year period, for these stocks was acquired from [1], stripped and converted into a set of daily sampled returns for each stock. Subsequently, each stock's data was split into a training and test set, with 70% of the data being used for training and 30% being used for testing.

The expected returns for these stocks were then calculated by summing each stock's returns over the training set and then dividing by the length of this stock's set of returns. The results are shown in table 3.

Stock	Expected Return	Standard Deviation/Risk
AutoTrader	-0.07%	2.0579
Experian	0.03%	1.7542
Rightmove	-0.07%	2.2174
Rolls Royce	0.13%	2.8016
Shell	0.08%	1.9300

Table 2. Expected Returns and Standard Deviation/Risk of the Selected Stocks

Following this, the covariance matrix was also derived from the training set and is shown in figure 4.

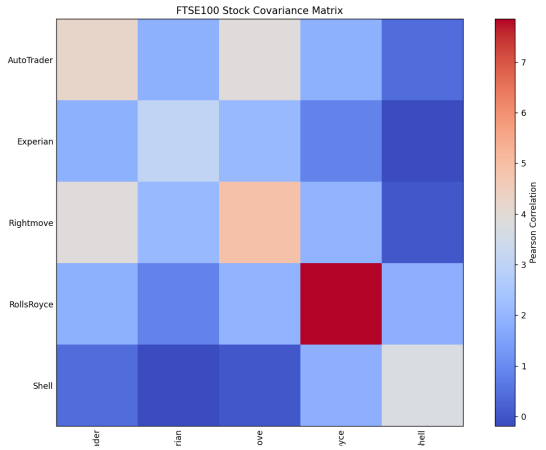

Fig. 4: A plot of the stock covariance matrix.

Although Rolls Royce had the highest expected return it also had the highest variance ($\backsim$7.85), meaning it carries a much higher risk than others such as Shell, which also still had a positive return of 0.08% but a much smaller variance of only $\backsim$3.72. Additionally, AutoTrader and RightMove show a high co-variance, indicating that the prices of the two stocks follow similar patterns and thus have similar risks and returns.

3.2 Optimisation Strategy

After calculating the expected returns for each stock and the covariance matrix of the portfolio, we proceeded to determine the set of efficient portfolios. An efficient portfolio is one with the highest expected return for a set level of risk, where the risk measure used is the standard deviation.

A portfolio can be described using a weight vector $\mathbf{w}$, with each element i corresponding to the proportion of the equity of the portfolio placed into stock i.

$$\mathbf{w} = \begin{pmatrix} \text{AutoTrader} \\ \text{Experian} \\ \text{RightMove} \\ \text{Rolls Royce} \\ \text{Shell} \end{pmatrix}$$

To calculate the set of possible portfolios, a grid search is performed over all defined combinations of the weight vector (using a step size of 0.02). Figure 5 shows a risk vs return plot for the training set, with each point representing a portfolio.

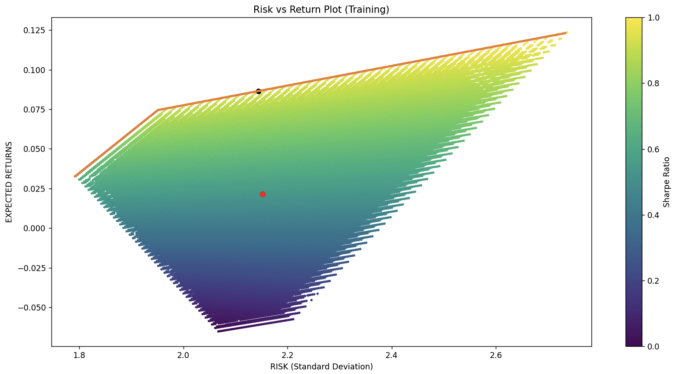

Fig. 5: The optimal and baseline portfolios

Additionally, figure 5 depicts the efficient frontier in red. The efficient frontier represents each portfolio that provides the highest expected return for each level of risk.

Using this plot, any portfolio along the efficient frontier could have been selected as an efficient portfolio. The optimal portfolio in the training set was defined as the portfolio that returned the largest ratio of expected return to risk and corresponds to the following weight vector.

$$\mathbf{w}^* = \begin{pmatrix} \text{AutoTrader} \\ \text{Experian} \\ \text{RightMove} \\ \text{Rolls Royce} \\ \text{Shell} \end{pmatrix} = \begin{pmatrix} 0.02 \\ 0.02 \\ 0.02 \\ 0.24 \\ 0.7 \end{pmatrix}$$

To compare the optimal portfolio to a baseline, a $1/n$ portfolio is defined. This corresponds to an even split of equity across all stocks and therefore has the corresponding weight vector.

$$\mathbf{w}^* = \begin{pmatrix} \text{AutoTrader} \\ \text{Experian} \\ \text{RightMove} \\ \text{Rolls Royce} \\ \text{Shell} \end{pmatrix} = \begin{pmatrix} 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \end{pmatrix}$$

The 2 points in figure 5 represent the two portfolios, with the red dot being the $1/n$ portfolio and the black dot being

the efficient portfolio. It is clear that the optimal portfolio outperforms the $1/n$ portfolio and lies on the efficient frontier.

3.3 Results

Following the selection of this portfolio, it was then compared to the $1/n$ portfolio on the test set.

To compare the portfolios, the expected returns across the test set for each stock were calculated and then weighted according to the weight vectors, and a plot of returns against time was created. These results are presented below.

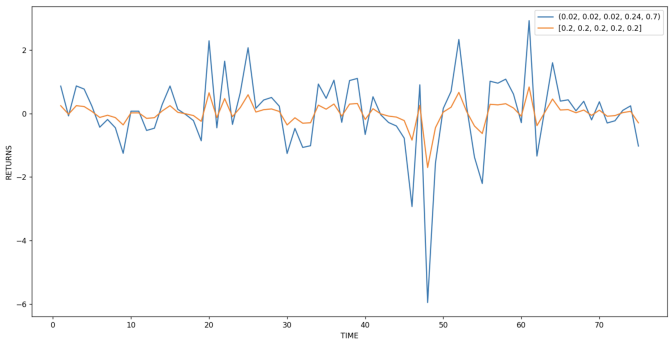

Fig. 6: Optimal Vs. $1/n$ portfolio returns

Portfolio	Weights	Returns	SD
Optimal	[0.02, 0.02, 0.02, 0.24, 0.7]	0.1873832716204515	2.14
$1/n$	[0.2, 0.2, 0.2, 0.2, 0.2]	0.1891476749419203	2.15

Table 3. Expected Returns of the Selected Stocks

Overall, the efficient portfolio was found to outperform the $1/n$ portfolio on the training set, however, overfits the training set. This is evidenced by returns on the test set being greatly diminished when compared to the test set and due to the sporadic nature of the portfolio on the test set. This promotes the use of a more advanced portfolio optimization scheme in future work.

References

[1] Yahoo, "FTSE 100 Components," Yahoo Finance. [Online]. Available: https://uk.finance.yahoo.com/quote//%5EFTSE/components?p=%5EFTSE