LUISS Guido Carli
MSc in Data Science and Management - Machine Learning Course
May 20, 2024

Euklid Report

A Systematic AI Trading Challenge

Project Technical Report

Gian Lorenzo Marchioni
ID: 788811, email: gianlorenz.marchioni@studenti.luiss.it

David Paquette
ID: 789331, email: d.paquette@studenti.luiss.it

Elena Tomasella
ID: 781321, email: elena.tomasella@studenti.luiss.it

Collaborative Business Case Proposed by Euklid

1 Introduction

In the rapidly evolving landscape of financial markets, technology, and more specifically, Artificial Intelligence, now plays a crucial role when shaping investment strategies. Through the use of Machine Learning, one of the main branches of AI, investors can now create complex financial models, with even more complex algorithms, that can instantly process enormous amounts of financial data. These models can then automatically make investment decisions, removing the emotional biases that humans often deal with, thus optimizing the investment process, and the investment outcome.

As this technology becomes more and more prominent, the growth of the science-backed financial industry. One of the leading players in this market is Euklid, an innovative hedge-fund who develop algorithms based on *biocomputing, a science linked to maths, physics and biology* [1]. These algorithms are then optimized through the use of various technologies, including swarm intelligence, neural networks, and genetic logic.

This challenge, to design a real systematic trading model based on the prediction of historical time series, was initiated at the request of Euklid. As such, this report will focus on the tasks undertaken in the creation of said model, including the preprocessing of the data, the feature engineering and selection, the model training, the metrics used to assess the model performance, and the model results, as well as the logic behind each of these tasks.

2 Methods

2.1 Data

The datasets used to create the features, train the model, and ultimately, create the final models were provided by Euklid. In total, there were six datasets. Three of these datasets consisted of the weekly historical financial data for tech companies `Apple, Microsoft,` and `IBM`. The other three consisted of the weekly historical financial data of funds following the `S&P 500 index`, the `NASDAQ index`, and the `CAC 40 index`. The variables in the dataset were as follows:

- `Date`: The week during which the financial data was recorded. Ranges from the 6^{th} of March, 1994 until the 3^{rd} of March, 2024. Only the Amazon dataset begins at a later date due to the company Initial Public Offering occurring on May 15^{th}, 1997.
- `Open`: The price at which the stock or index fund began trading at for that week.
- `High`: The price at which the stock or index fund peaked at for that week.
- `Low`: The price at which the stock or index fund was at its lowest for that week.
- `Close`: The price at which the stock or index fund finished trading at for that week.
- `Volume`: The total number of shares for the stock or index fund that were traded that week.

2.2 Libraries

A total of 8 python libraries were used in the creation of the model. The libraries used in the preprocessing and feature engineering are:

- `Pandas`: Library used for data analysis and manipulation.
- `Numpy`: The universal data for working with numbers in Python.
- `Pandas-TA`: Contains over 130 Technical Analysis Indicators.
- `Skicit-Learn`: Library containing efficient tools for machine learning and statistical modeling.
- `Hurst`: Allows for the calculation of the Hurst Exponent.

The models were created with the usage of the following two libraries:

- `Keras`: Library used to implement neural networks. Used for the `ARIMA` and `LSTM` models.
- `PyTorch`: Framework for deeplearning models. Used for the creation of the ANFIS model.

Lastly, the `Matplotlib` library was used for visualizations.

2.3 Preprocessing

2.3.1 Missing Values

When analyzing the datasets, it appeared that certain weeks had missing values, either for the entire week, or only for select financial variables. For instance, all index datasets were missing all values for the weeks of the 14^{th} of May, 2023 and the 21^{st} of May, 2023. As such, these values were manually inputted from with the usage of **YahooFinance**. The `S&P 500` values were taken from the the `^SPX` fund [2], the `NASDAQ` values were taken from the `^IXIC` fund [3], and lastly, the ones for the `CAC 40` were taken from the `^FCHI` fund [4].

2.3.2 Stock Splits

After further investigating the dataset, it became apparent that the stocks had undergone various stock splits, i.e., "when a company increases the number of its shares to boost the stock's liquidity" [5]; consequently, while the volume numbers for the stocks increased due to the number of shares increasing, the price of said stock was divided by the number with which one stock was divided into. The following are the dates which the stock splits occurred per stock dataset, for the range of dates present in said dataset:

- `Amazon`:
  - `2022-06-06`: 20:1 stock split
  - `1999-09-05`: 2:1 stock split
  - `1999-01-05`: 3:1 stock split
  - `1998-06-02`: 2:1 stock split
- `Microsoft`:
  - `2003-02-18`: 2:1 stock split
  - `1999-03-29`: 2:1 stock split
  - `1998-02-23`: 2:1 stock split
  - `1996-12-09`: 2:1 stock split
  - `1994-05-23`: 2:1 stock split
- `IBM`:
  - `2021-11-04`: 1046:1000 stock split
  - `1999-05-27`: 2:1 stock split
  - `1997-05-28`: 2:1 stock split

In order to ensure a natural price progression, to calculate accurate indicators, and due to the fact that an "unadjusted stock chart would contain lots of bearish [signals]"[6], it was deemed necessary to convert the current prices into their adjusted prices. This adjustment was done by dividing all financial variables, i.e., `Open, High, Low, Close`, and `Volume`, by their split ratios for the dates prior to when their respective stock splits occurred. For example, in the `Amazon` Dataset, all variables for dates prior to `2022-06-06` were divided by 20, then followed by the dates prior to `1999-09-05` which were divided by 2, then followed by the dates prior to `1999-01-05` which were divided by 3, and lastly, the dates prior to `1998-06-02` which were divided by 3. This process was done with all datasets, and the calculated values were subsequently compared to the numbers on **TradingView**, the source of the six original datasets. Please refer to the following images in order to best comprehend the effects of the price adjustments.

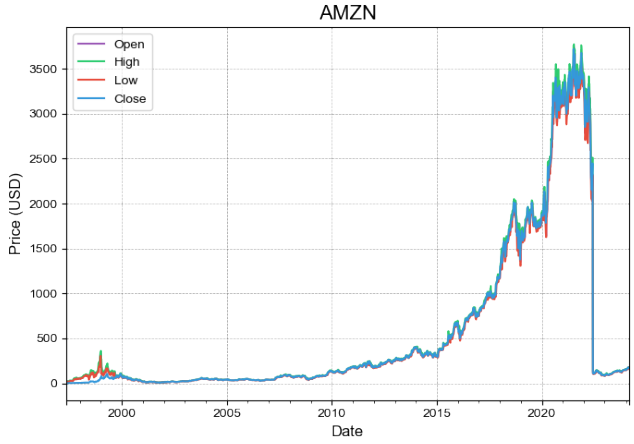

Figure 1: Amazon Stock Prices Before Split Adjustment

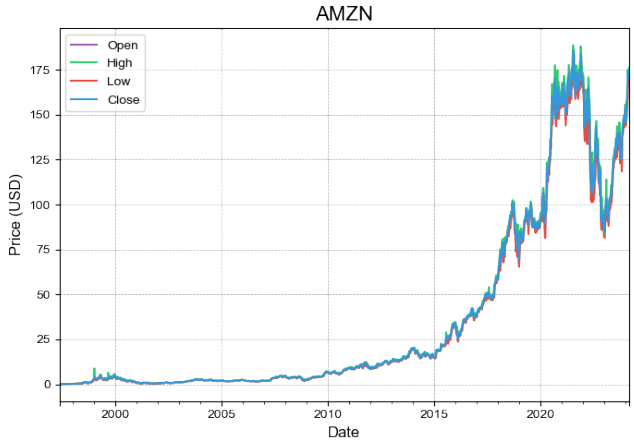

Figure 2: Amazon Stock Prices After Split Adjustment

On the other hand, the index datasets did not require adjustments for splits, as the indexes present are all capitalization-weight indexes, suggesting that they are calculated market capitalization, i.e., "the total dollar market value of a company's outstanding shares of stock" [7], which is unaffected by stock splits.

2.3.3 Incorrect Close Values

While the price adjustment for stock splits finalized the preprocessing for the `Amazon` dataset, the `IBM` and `Microsoft` dataset still had incorrect price values even after the correction for splits. The main issue was that the `Close` price values were lower than the `Low` price values, which was theoretically impossible given that the `Low` variable represented the lowest price point the stock or index had reached that week. In total, the `IBM` dataset had **1370** instances of this occurring between 1994 and 2024, and the `Microsoft` dataset had **1285** instances of this between 1994 and 2023. This posed significant problems for the creation of the model, as the technical indicators which would be used to create

the algorithms utilized the close price.

In the end, in order to resolve this issue, an assumption was made that for the weeks where the `Close` prices were smaller than the `Low` prices, the close price for said week would be equal to next weeks open price. While this assumption is theoretically wrong as stock prices can still move once the exchanges close due to after-hours trading, this assumption greatly simplified the model and ensured that the discrepancy between close and low prices was fixed.

2.4 Feature Engineering & Selection

Once the datasets were cleaned, and the price values had all been corrected, the selected features were prepared. The following are the selected features that require engineering:

- `Relative Strength Index`
- `Exponential Moving Average`
- `Current Price Change`
- `Commodity Channel Index`
- `Exponential Moving Average Normalized Difference`
- `Weekly Return`

2.4.1 Relative Strength Index (RSI)

The **Relative Strength Index (RSI)** is a momentum indicator used in technical analysis which measures the size and speed of recent price changes. It is calculated with the following equation:

$$RSI = 100 - \frac{100}{1 + RS} \tag{1}$$

where RS is calculated as the ratio of the average gain of the up periods to the average loss of the down periods during the specified time frame. Specifically, RS is defined as:

$$RS = \frac{\text{Average Gain of Up Periods}}{\text{Average Loss of Down Periods}} \tag{2}$$

The number of periods used to calculate the average gain or loss can vary; however, J. Welles Wilder, the creator of the indicator, suggested the usage of 14 periods[8].

RSI is an extremely used to identify the general trend of the stock, as well as to measure its oversold and overbought levels. If the calculated RSI between 70 and 100, it indicates that the stock price is potentially at a resistance level as it is overbought, and as such, it is a bearish signal. If an RSI between 0 and 30 is calculated, it indicates that the stock price may be at a support level, suggesting that the stock is oversold, and is therefore, a bullish signal. Lastly, if the RSI is calculated to be between 30 and 70, it indicates that the price point for the stock is at a neutral level, suggesting that the stock is not exuding any buy or sell market conditions.

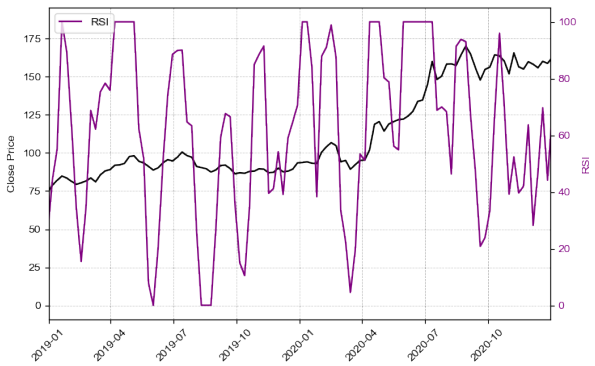

Figure 3: Example of the time series of the RSI indicator compared to the underlying price.

2.4.2 Exponential Moving Average (EMA)

The **Exponential Moving Average (EMA)** is a type of moving average which "places a greater weight and significance on the most recent data points" [9] than a simple moving average, which places an equal weight on data points. It is calculated with the following formula:

$$EMA_t = \left(V_t \times \left(\frac{s}{1+d}\right)\right) + EMA_{t-1} \times \left(1 - \frac{s}{1+d}\right) \tag{3}$$

where V_t is the value of the stock today, s is the smoothing factor (typically set to 2), and d is the number of days for the moving average period.

Experts often use `EMA's` of various periods of time, with the typical being a 50 day EMA, i.e., 10 weeks, as the short-term `EMA`, and a 200-day one, i.e., 40 weeks, as the long-term `EMA`. If the short-term `EMA` were to surpass the long-term one, that would be referred to as a **Golden Cross**, which would be interpreted as a bull signal; conversely, should the long-term `EMA` surpass the short-term one, it would be referred to as a **Death Cross**, a bear signal.

2.4.3 Current Price Change (CPC) & Commodity Channel Index (CCI)

The following two indicators were both taken from an academic paper titled, "Stock Forecasting using Fuzzy Neural Networks, Technical Indicators, and Foreign Exchange Rates" [10], by Seok-Woo Jang of the department of software of Aanyang University in South Korea.

The first of the aforementioned indicators is the **Current Price Change (CPC)**, which is used to measure recent stock price movements by comparing the closing price to moving average of periods. It is computed with the following formula:

$$CPC = \frac{1}{1 + e^{-\left(\frac{C_t - MA_{t-1,t-i}}{MA_{t-1}} \times 100\right)}} \tag{4}$$

where C_t is the closing price at time t, and MA_{t-1} is the moving average for the previous period for a predefined time period of t.

The second is the **Commodity Channel Index (CCI)**, a technical indicator developed by Donald Lambert to help traders "[assess] price trend direction and strength"[11]. It is similar to the RSI, as they are both momentum-oscillators, but the CCI is unbounded, suggesting that it can go higher or lower indefinitely. As such, in order to understand if a stock is overbought or oversold, it is essential to compare CCI levels to historic CCI levels for said stock. It is calculated with the following formula:

$$CCI = \frac{M_t - SM_t}{0.015 \times D_t} \tag{5}$$

where, M_t is the sum of the high, low, and close price at time t divided by 3, SM_t is the simple moving average for period t, and D_t is the mean deviation of M_t from SM_t over n periods.

2.4.4 Exponential Moving Average Normalized Difference

Due to the unbounded nature of the `EMA` indicator, and due to the nature of the `ANFIS` model (more to be discussed about this in the **Experimental Design** section), it was necessary to create a stationary indicator. The solution was the **Exponential Moving Average Normalized Difference (EMADN)**, which is essentially the percentage change between the short and long `EMA`. It is a simplified way of interpreting Golden and Death crosses. It is calculated as following:

$$EMADN = \frac{EMA_{\text{short}} - EMA_{\text{long}}}{EMA_{\text{long}}} \tag{6}$$

2.4.5 Hurst Exponent

The **Hurst Exponent** is a single scalar value used to identify if a "time series is purely random, trending, or rather mean reverting"[12]. While not a traditional financial indicator, the ability to predict the movement of time-series data has extremely interesting financial applications. It is calculated with the following equation:

$$H = \lim_{T \to \infty} \frac{\log(R/S)}{\log(T)}$$

where:

- R is the range of the cumulative deviations from the mean,
- S is the standard deviation of the observations,
- T is the length of the time series.

If the Hurst Exponent is calculated to be greater than 0.5, it indicates that the time-series is trending, suggesting that the current trend will continue. If the exponent is calculated to be less than 0.5, it indicates that the time-series is mean-reverting, suggesting that the current trend will reverse. Lastly, if the exponent is approximately equal to 0.5, it indicates that the time-series' next direction is random.

2.4.6 Weekly Return

This **Weekly Return** indicator measures the percentage change between the closing prices of two consecutive weeks. It is calculated with the following formula:

$$WR = \frac{C(t) - C(t-1)}{C(t-1)} \tag{7}$$

This indicator was created in order to create the final target variable. Should the weekly return be positive, the target variable would be assigned a `1`, creating a sign to buy the stock. If the return was negative, the target variable would be assigned a `-1`, suggesting a sell/short signal. If the return was between $\pm 0.5\%$, the target variable would be assigned a `0`, suggesting to the model that no action should be taken.

2.5 Scaling

Lastly, in order to use the price variables and volumes as features in the final models, said features needed to be scaled. Due to the fact that neither prices nor volumes could be negative, the `MinMaxScaler` from the `Sckicit-Learn` python library was used as it scales the data to a range between 0 and 1.

3 Experimental Design

The trading decision model is based on two approaches:

1. Close price time series prediction.

2. Market behavior prediction.

While the two seem similar, their ML implementation is instead quite different.

Time series prediction requires a model to be fitted or trained on the historical data for each different financial product. This can be done either through autoregressive models like ARIMA, or through a recurrent neural network.

For behavior prediction, a model classifies whether the following week will be bullish, bearish, or flat, based on the learned relationships between features, such as the indicators listed in section 2.4, computed at the current week. This allows for a model to be trained on a single dataset, and then to be tested on every other financial product, provided that the necessary features are calculated and scaled.

3.1 ARIMA

As a baseline, the ARIMA model was chosen. As this model requires stationarity of the inputs, *log* was used both on prices and returns. By starting with a deep study on how to select the proper parameters of the model (please refer to the paper by Robert Nau [13] for all the following notation concerning this model), it became clear that available data wasn't fulfilling the requirements to obtain good results in the end. Indeed, by testing the PACF and ACF, it was revealed that lollipops didn't have the shape one could expect from the theory. In theory, once a dot is inside the confidential interval, it should stand for all the following dots too; however, this wasn't the case for any PACF nor ACF of the datasets. An instance follows in figure 4 below: keep in mind the PACF plots are used to select p parameter, while ACF plots do the same for q.

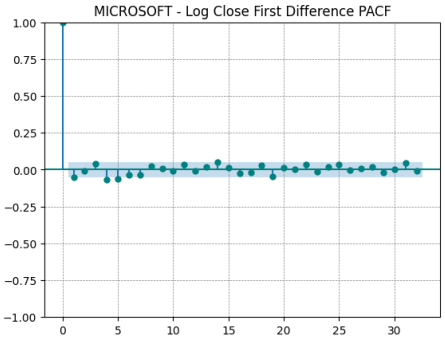

Figure 4: Log Close First Difference PACF for MICROSOFT

Once **ARIMA model** was constructed via setting the parameters p,q and d, the plotted results for the prices' prediction of the validation set were coherent with the overall trend, on the stocks and indexes. Conversely, all datasets in more recent years were characterized by a particularly new range of the prices that at the end resulted in having a prediction as in figure 5, which is not accurate enough for the foundations of an AI trading model.

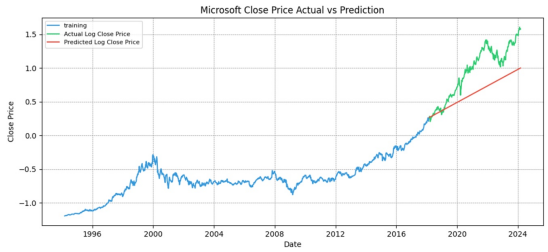

Figure 5: ARIMA prediction for IBM on log returns.

3.2 LSTM

The **Keras** library was used to implement the LSTM models. Many `for` loops were used to iterate on different sets of features to be considered in the window, but also on different amount of epochs. The preparation of the input dataset was a time-consuming task because of issues in the dimensions' request of the library. Different **Sequential** architectures were tried, but for all of them, the LSTM layers were surrounded by DropOut layers. This was thought to reduce overfitting, complementing the reasonable choice of 0.9 as percentage of the training set (a trade-off between smaller test set and MSE reduction driven the decision of this percentage). In particular, the model wasn't performing well on the validation set because it was learning too much from the history of stock and indexes prices. Increasing the number of epochs was attempted through brute force, but the MSE plot revealed that it was not the ideal method to obtain a substantial increase in the quality of the model.

3.3 Decisional Neural Network

The **ARIMA** and **LSTM** models were then combined to construct a first AI trader. The price predictions provided by the two models are weighted in three ways, each with increasing complexity: simple mean, minimization of a cost function, and also with a neural network. For the combined model, the main takeaway concerns the low accuracy in predicting the price, showing a significant difference in the model's performance between the training and test sets. Addressing the Hurst coefficient as a measure for the autocorrelation of the time-series, the exponents for the returns and the log-returns have been calculated to investigate which of the two had higher stationarity. It would be mindful to use the quantity posing the Hurst exponents the furthest from 0.5. As the **Hurst coefficient** goes from 0 to 1, having a value which is not centered means high predictability and consequent better results in the AI-trading strategy. However, after this analysis, the team suggested that the issues with the initial data are blocking any possibility for LSTM and ARIMA to perform well in this task. In the end, with this combined model the AI-trading strategy results in the selection always being flat.

3.4 ANFIS

The Adaptive Network-based Fuzzy Inference System first proposed by Jang [14], is a 5-layer neural network architecture that leverages fuzzy logic.

The implementation suggested for this project was developed by James Power [15], who adapted the basic version of ANFIS as introduced by Jang to the PyTorch framework. Some additions were made to the modules, specifically `anfis.py`, `membership.py` and `experimental.py`, mainly in order to adapt the module into a market behavior classifier. These modifications will be discussed shortly, upon illustrating the steps involved in the ANFIS network.

Before training the model, a thorough feature selection process was conducted. The effectiveness of technical indicators such as RSI, CPC, CCI, and EMAs is heavily influenced by their time window parameters. Since the literature on 'optimal' values for these parameters is limited, a grid search was employed. This search aimed to adjust the parameters to bring the Hurst Exponent of each indicator's time series closer to 1. By doing so, the features would better capture the trend of the underlying price, based on the assumption that a time series with a Hurst Exponent of 1 indicates persistent behavior and is more predictable (see Table 1).

Indicator	Window Parameter	Hurst Exp.
CCI	36	0.90
EMADN	24 / 52	0.92
CPC	39	0.99
RSI	38	0.96

Table 1: Optimized parameters for the indicators for the AMZN dataset. The grid search lead to quite similar parameters for all 6 datasets.

Fuzzy logic is based on relating numerical data, such as the *RSI* value at a certain week, with semantic labels, such as '*oversold*', '*neutral*' or '*overbought*'. This is done by defining a *membership function* (MF) for each label i, which takes the numerical input x and outputs a *membership score*:

$$\mu_i(x) : R \to [0; 1]$$

In the *RSI* example, its value $x(t)$ would be 'embedded' into a vector of three membership values corresponding to:

$$x(t) \to (\mu_{oversold}(x(t)), \mu_{neutral}(x(t)), \mu_{overbought}(x(t)))$$

This step is performed in the first layer of ANFIS, the 'fuzzification' layer. To set it up, appropriate membership functions (MFs) and their parameters must be chosen. Initially, the available MFs included Gaussian, bell, triangular, and trapezoidal shapes. To enhance the model's capability, sigmoid MFs were added to the existing modules. These were specifically chosen for their ability to model the threshold-like signals exhibited by indicators at extreme values. The step-like nature of sigmoid functions is particularly advantageous as it preserves the distinct threshold characteristics of such signals even after model training, which is less reliably achieved with MFs like Gaussian. Referring to the *RSI* example, where 'oversold' and 'overbought' conditions are traditionally marked by threshold values of 30 and 70 respectively, sigmoids are ideal for capturing these extreme points. Given that all features were scaled, a reasonable initial set of MFs is therefore the one depicted in figure 6.

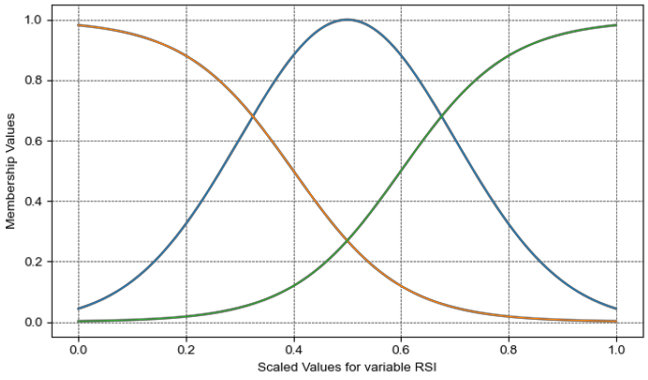

Figure 6: Initial MFs for the *RSI* feature, corresponding to *overbought* (orange), *neutral* (blue), *oversold* (green)

Since all features were scaled to the same range using a MinMax scaler, the same set of MFs as in figure 6 was applied to all of them.

Membership scores for all features, in this classic implementation of ANFIS, are used to compute the 'firing strength' of Takagi-Sugeno rules, in a '*consequent layer*'. Each rule outputs a number which is a linear combination of input features according to a set of parameters, called '*consequent parameters*'.

The last layer performs a sum of each rule's output, weighted by the respective firing strength. In this case the output variables are the three class scores, therefore the consequent layer contains 3 distinct sets of rules, each one contributing to one output variable. This mechanism simulates intuitive decision-making, such as: if RSI is 'Overbought' and the Past Week Return is 'Strongly Positive', then modify the probability that the next week's behavior is 'Bearish'. Model training optimizes the parameters for both the membership functions and the formulae for rules output. Figure 7 displays how the *premise parameters*, dictating the shape of the membership functions, also get optimized during training.

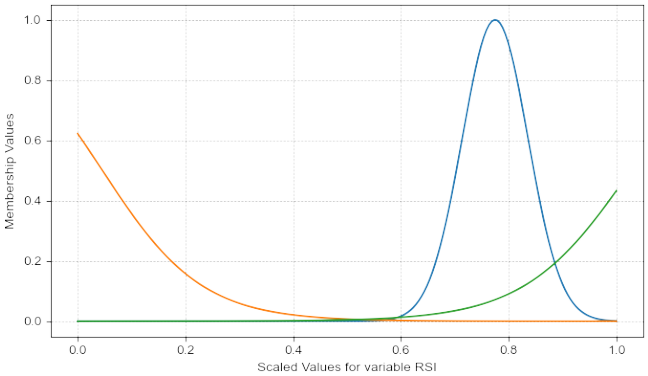

Figure 7: MFs for the *RSI* feature, after training

4 Results

The ANFIS model was evaluated using accuracy as a metric and also assessed by the cumulative return and the drawdown period. The cumulative return can be defined as the compound profits generated by the long-short equity trading strategy, while the drawdown is "a peak-to-trough decline during a specific period" of the cumulative return [16].

Given the poor performance of the price prediction models, the baseline used for comparison of cumulative returns was a simple 'buy and hold' strategy. This corresponds to a classifier that always outputs 'Bullish' market behavior, thereby opening a long position at week 1 of a dataset and cashing out at the end of the time period. The related return, therefore, is equal to the percentage change in price over the time period.

The cross-validation process involved, for each dataset (e.g., AMZN):

- Training the model on the training set of AMZN, which comprises 90% of the time series.
- Using the AMZN test set as validation.
- Scaling the other test sets using the scaler fitted on the original dataset. For example, before making predictions on IBM's test set using the model trained on AMZN, the IBM test set was scaled using the min-max scaler fitted on the AMZN data.
- Testing the model by making predictions using the test sets obtained from other datasets.

The cumulative returns were measured over the entire validation/testing period spanning just over 2 years and 7 months. Yearly returns were also estimated for the 2 complete years of the test sets.

For brevity, only the best results from the cross-validation are included, i.e., the ones using the model trained on the AMZN dataset, in Table 2.

Figures 8 and 9 display the related performance, in terms of return and drawdown, on the validation set and on one of the cross-tests.

5 Conclusion

This group project was lead by a strong commitment from all of the team's members to learn from each other as much as possible. The different backgrounds allowed for the opportunity to share domain knowledge in finance topics, and maths and Physics' methods. As for general understanding the task given with was not easygoing at all with the main take-away entailing the collaborative skills, as well as the ideal way in which a Machine Learning project should be organized when issues from working with real data arise.

Considering the specific financial environment, a technical take-way regards the pros and cons that should be leverage before deciding which Python library to implement. As this project was supposed to deal with time series from the beginning, having a solid understanding of the theory behind it is an advantage.

Trained on: AMZN				
Test Set	Accuracy	Overall Cumulative Return	1^{st} Year Return	2^{nd} Year Return
AMZN	47%	100.9% (6.2%)	20.2% (-14.5%)	32.8% (-2.8%)
MSFT	49.0%	25.6% (57.2%)	14.5% (6.6%)	7.0% (6.1%)
IBM	43.0%	16.2% (45.3%)	4.6% (0.2%)	6.6% (2.6%)
CAC	44.0%	24.2% (27.3%)	-0.3% (5.3%)	10.5% (14.3%)
S&P500	44.0%	21.6% (21.7%)	2.1% (5.1%)	8.1% (-3.1%)
NASDAQ	47.0%	36.7% (14.1%)	12.2% (-4.7%)	10.5% (-5.6%)

Table 2: Results from cross-testing the ANFIS classifier trained on the AMZN dataset. Baseline strategy returns in parentheses.

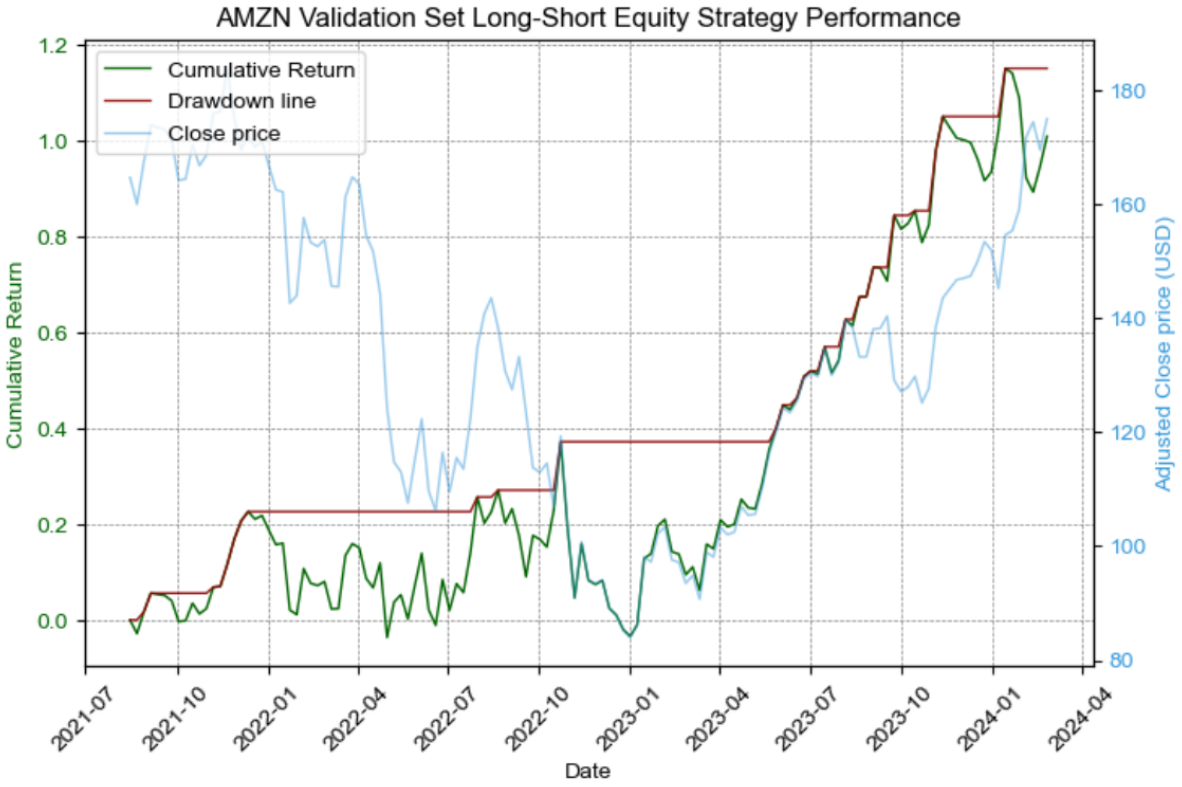

Figure 8: Cumulative return and drawdown over the AMZN validation set. Maximum drawdown: 32 weeks

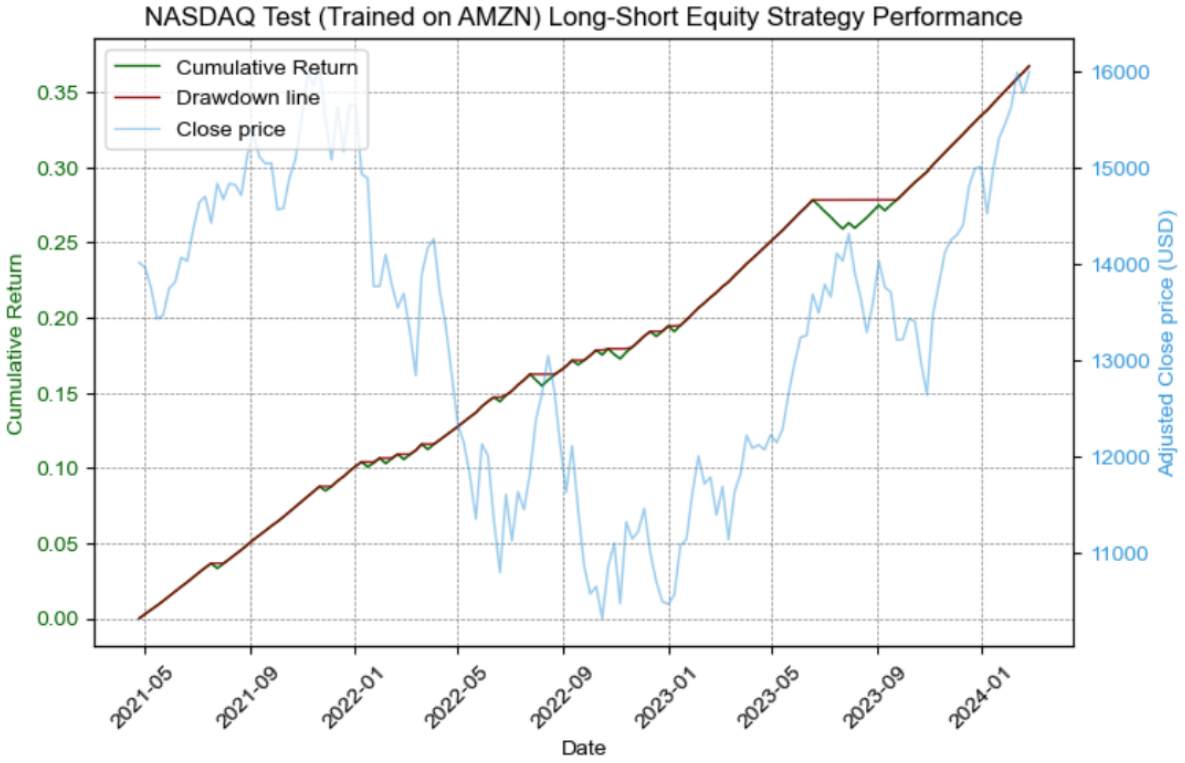

Figure 9: Performance of the model trained on AMZN, using the NASDAQ dataset. Maximum drawdown: 13 weeks

References

[1] Euklid. Our algos. https://euklid.uk.com/home, 2024.

[2] Yahoo Finance. S&P 500 INDEX (ŜPX). https://finance.yahoo.com/quote/%5ESPX/history?frequency=1wk.

[3] Yahoo Finance. NASDAQ Composite (ÎXIC). https://finance.yahoo.com/quote/%5EIXIC/history.

[4] Yahoo Finance. CAC 40 (F̂CHI). https://finance.yahoo.com/quote/%5EFCHI/history.

[5] Adam Hayes. What a stock split is and how it works, with an example. https://www.investopedia.com/terms/s/stocksplit.asp, 2023.

[6] StockCharts. Historical price data is adjusted for splits, dividends and distributions. https://support.stockcharts.com/doku.php?id=policies:adjusted_data.

[7] Jason Fernando. Market capitalization: What it means for investors. https://www.investopedia.com/terms/m/marketcapitalization.asp, 2024.

[8] StockCharts. Relative strength index (rsi). https://school.stockcharts.com/doku.php?id=technical_indicators:relative_strength_index_rsi.

[9] James Chen. What is ema? how to use exponential moving average with formula. https://www.investopedia.com/terms/e/ema.asp, 2024.

[10] Seok-Woo Jang. Stock forecasting using fuzzy neural networks, technical indicators, and foreign exchange rates. *International Journal of Advanced Trends in Computer Science and Engineering*, 9:3345–3349, 06 2020.

[11] Cory Mitchell. What is the commodity channel index (cci)? how to calculate. https://www.investopedia.com/terms/c/commoditychannelindex.asp, 2024.

[12] Detecting trends and mean reversion with the hurst exponent. https://macrosynergy.com/research/detecting-trends-and-mean-reversion-with-the-hurst-exponent, 2023.

[13] Nau Robert. Introduction to arima models. https://people.duke.edu/~rnau/411arim.htm, 2019.

[14] J-SR Jang. Anfis: adaptive-network-based fuzzy inference system. *IEEE transactions on systems, man, and cybernetics*, 23(3):665–685, 1993.

[15] James Power. ANFIS in pyTorch: Implementation of the ANFIS system using pyTorch. https://github.com/jfpower/anfis-pytorch/tree/master, 2023. Accessed: 2024-05-14.

[16] Cory Mitchell. Drawdown: What it is, risks, and examples. https://www.investopedia.com/terms/d/drawdown.asp, 2023.