Project Report

QF2103

Research on Trading Strategies using Python

Group 6 Team Members:

Lai Hui Qi (A0236474Y)

Lee Hui Yu, Laura (A0240633N)

Megan Ng Kai Lin (A0238870W)

Venus Ong Shi Xuan (A0238128A)

Introduction to Report and Assignment

This report aims to provide a comprehensive analysis on the research on 2 different trading strategies, using Python as a computational tool, and explores the application of machine learning techniques to develop and enhance trading strategies. Specifically, this research focuses on two key strategies: **local extrema** identification and **mean reversion** analysis, both using **logistic regression** as the machine learning model.

The research aims to contribute to the field of quantitative finance by providing practical insights and methodologies for building effective trading strategies using machine learning techniques.

The stock data used can be obtained from the first five stocks in **tr_eikon_eod_data.csv**, named **AAPL.O**, **MSFT.O**, **INTC.O**, **AMZN.O**, and **GS.N**. The completed codes for this research can be found in **Trading_Strategies_Analysis_Codes.ipynb**, along with code-specific instructions and observations. This report is a detailed analysis on the application of the two strategies with the machine learning model.

Team roles and responsibilities

Student Number	**Member**	**Roles and Responsibilities**
A0236474Y	Lai Hui Qi	Documentation Strategy 2 : Mean Reversion with Logistic Regression - Overview of Strategy - Logic behind Strategy - Optimization and Impacts - Improvements Codes 1. Strategy 2: Mean Reversion with Logistic Regression - Basic and Enhanced Strategy Classes - MeanReversion Class - EnhancedMeanReversion Class - Includes : Feature engineering, Strategy logic, ML pipeline, Strategy evaluation tools - Parameter optimization function using grid search approach 2. Functions and plots for comparisons between strategies - Profit, accuracy and precision extraction and comparison for strategy 1, strategy 2 and both 3. In code / notebook documentation for above strategy, classes, and functions - Docstrings, comments and observations
A0240633N	Lee Hui Yu, Laura	Documentation Strategy 1 : Local Extrema with Logistic Regression - Overview of Strategy - Performance of Strategy

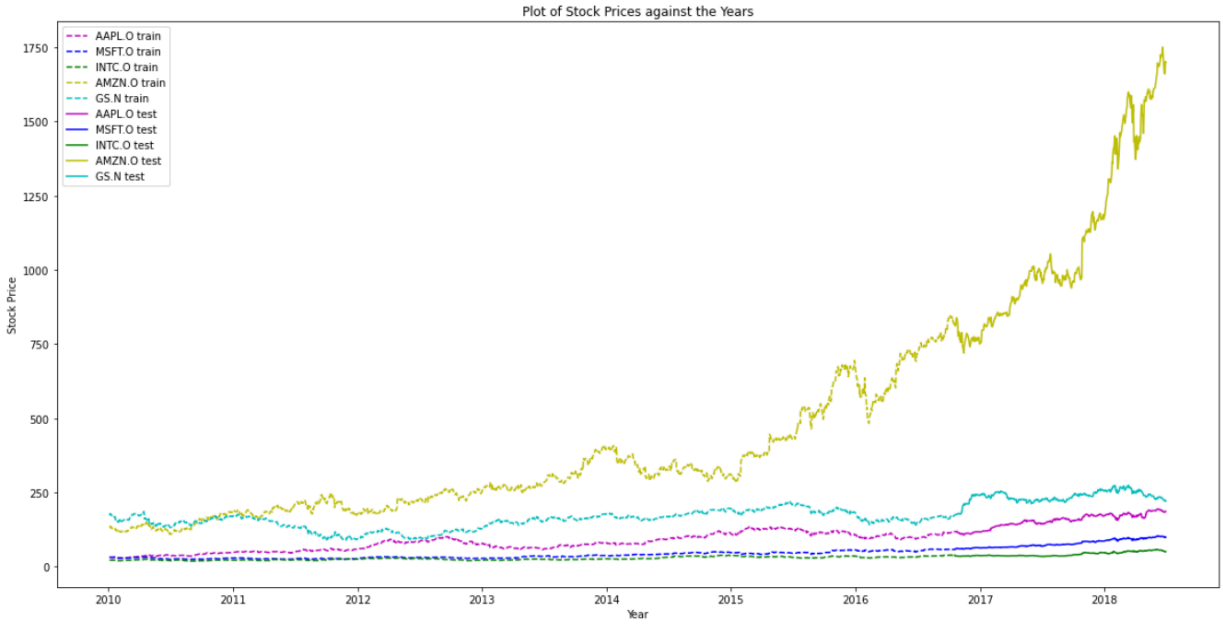
		- Challenges and Risks of Strategy - Improvements <u>Codes</u> 1. Strategy 1: Local Extrema with Logistic Regression - Basic - Model fitting - Model selection - Model evaluation, visualisation and performance analysis - Enhanced - RSI - Selective prediction of 50% - Enhanced model evaluation - Enhanced model training (finding the best stock to train on)
A0238870W	Megan Ng Kai Lin	<u>Documentation</u> Strategy 2 : Mean Reversion with Logistic Regression - Overview of Strategy - Feature Engineering - Performance of Strategy - Challenges and Risks of Strategy - Comparison of both strategies <u>Codes</u> 1. Strategy 2: Mean Reversion with Logistic Regression - Data processing - Feature Engineering
A0238128A	Venus Ong Shi Xuan	<u>Documentation</u> Strategy 1 : Local Extrema with Logistic Regression - Logic of Strategy - Feature Engineering - Optimization <u>Codes</u> 1. Strategy : Local Extrema with Logistic Regression - Basic - Preprocessing and labelling of data - Feature selection (Gradient difference) - Strategy evaluation - Simulate buy and sells to find profit - Derive other trading strategy metrics - Visualisation of graph with only relevant buy-sell pairs - Enhanced - Feature selection (MACD difference) - Relabelling of data to include new features - Evaluation metrics to find enhanced profit and other metrics - Visualisation and plotting performance of strategy

Strategy 1: Local Extrema with Logistic Regression Model

a. Overview of Strategy

Local min-max strategy, otherwise known as the local extrema trading strategy is a method used by traders to identify potential entry and exit points based on local highs (peaks) and lows (troughs) in price movements. This strategy aims to capitalise on short-term price fluctuations within a larger trend.

Justification



Based on the graph, here are a few reasons why a local extrema strategy would be suitable for these stocks. The stock prices show volatility with discernible peaks and troughs, suggesting that the prices do not move in one direction indefinitely. This characteristic is important for the local extrema strategy, as it relies on capturing profits from trend reversals after identifying local maxima or minima. The graph also indicates that the stocks have historical patterns of rising to a peak and then retracting, or falling to a trough and then recovering. These patterns could potentially be exploited using a local extrema strategy by entering and exiting positions at these points. There are clear moments where the upward momentum slows down and results in a local maximum, followed by a period of consolidation or retracement. A local extrema strategy could be used to sell before the retracement and buy back when the price stabilizes or reaches a local minimum.

b. Logic Behind Strategy

This methodology is geared towards exploiting short-term price oscillations within the broader context of a prevailing trend.

Preprocessing

```
def generateLabel(data):
    gradient = np.gradient(data)
    label = np.zeros_like(gradient)

    for i in range(1, len(gradient)):
        if gradient[i] == 0:
            if gradient[(i - 1)] > 0:
                label[i] = -1
            else:
                label[i] = 1
        else:
            if gradient[i] > 0 and gradient[i - 1] < 0:
                label[i] = 1
            elif gradient[i] < 0 and gradient[i - 1] > 0:
                label[i] = -1
            else:
                label[i] = 0

    return pd.Series(label)
```

In implementing this strategy, we labelled data points according to certain criteria:

1. **'Buy' / 'Long' (1)** will be signalled on the next day when it is a local minimum
    a. A local minimum is either seen from current day gradient being zero, and the gradient of the previous day is a negative
    b. Or when the gradient is positive but the previous day gradient is negative

2. **'Sell' / 'Short' (-1)** will be signalled on the next day when it is a local maximum
    a. A local maximum is either seen from current day gradient being zero, and the gradient of the previous day is a positive
    b. Or when the gradient is negative but the previous day gradient is positive

3. **Otherwise, 'Hold' / 'Keep' (0)** will be signalled, no trading on stock is recommended on the next day.

Feature Selection: Our strategy involves the selection of feature derived from the price data, namely the difference between the gradient of the current day and the gradient of the previous day. This feature serves as input to our machine learning model, aiding in the prediction and decision-making process.

Model Fitting and Evaluation

1. Sequentially splitting data into 80% training and 20% testing.
2. Train a Logistic Regression model using the training data.
3. Make predictions on the test data.
4. Generate a classification report to evaluate the model's performance.
5. Visualise the performance of the model using a confusion matrix.

Simulating buying and selling calls

```
def simulate(call, data, capital = 10000):
    stocks = 0
    profit = capital

    for i in range(len(call)):
        if call[i] == 1 and capital > data[i + 1]:
            stocks += profit // data[i + 1]
            profit = profit % data[i + 1]
        elif call[i] == -1 and stocks > 0:
            new_profit = data[i + 1] * stocks
            profit += new_profit
            stocks = 0

    return ((profit + (stocks * data.iloc[-1]))/ capital) - 1
```

The stock trading process begins with the predicted (suggested) stock position, using a default base principal of $10,000 (the specific value is less significant as performance is evaluated in percentages). 'Buy All' and 'Sell All' strategies are employed during the stock trading period.

'Buy all':
When a 'buy' signal is generated and the available capital exceeds the stock price, the strategy is to purchase as many shares as possible with remaining capital after each purchase. This continues until the remaining capital is insufficient to buy additional units.

'Sell all':
When a 'short' or 'sell' signal is generated and there are shares in hand, the strategy is to sell all of the shares held.

```
def evaluate_metrics(call, data, dataname, capital = 10000, risk_free_rate = 0,):
    shares = 0
    df = pd.DataFrame()

    base_wealth = capital + shares * data[0]

    df['current'] = 0

    for i in range(len(data)):
        curr_price = data[i]

        if i < len(call):
            # 'Buy all' : Buy as many shares as possible
            if call[i - 1] == 1 and capital >= curr_price:
                share = capital // curr_price
                shares += share
                capital -= share * curr_price

            # 'Sell all' : Sell all shares held
            elif call[i - 1] == -1 and shares > 0:
                capital += shares * curr_price
                shares = 0
```

The current wealth is calculated as:
Current wealth = Current capital + Current number of shares * Current stock price

```
# Calculate current wealth
df = pd.concat((df, pd.DataFrame({'current': (shares * curr_price) + capital}, index = [0])), ignore_index=True)
```

Gradient of current day stock - gradient of previous day stock

```
def getFeatures(data):
    gradient = np.gradient(data)

    return pd.Series(gradient) - pd.Series(gradient).shift(1)
```

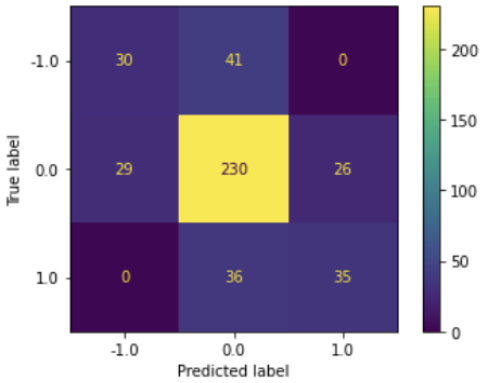
→ gradient represents the rate of change of the stock price to identify local extrema. Defining Trading Signals: Once local highs and lows are identified, traders establish trading signals based on the relationship between these extrema and the current price. A buy signal is generated when the price breaks above a local high, indicating potential upward momentum. Traders may interpret this as a signal to enter a long (buy) position, anticipating further price increases. Conversely, a sell signal may be generated when the price breaks below a local low, suggesting potential downward momentum. Traders may interpret this as a signal to enter a short (sell) position, anticipating further price declines.

d. Performance

Classification Report

	precision	recall	f1-score	support
-1.0	0.51	0.42	0.46	71
0.0	0.75	0.81	0.78	285
1.0	0.57	0.49	0.53	71
accuracy			0.69	427
macro avg	0.61	0.57	0.59	427
weighted avg	0.68	0.69	0.68	427

Confusion Matrix



Trading Strategy Metrics

To assess the performance of our trading strategy, we employ the following metrics:

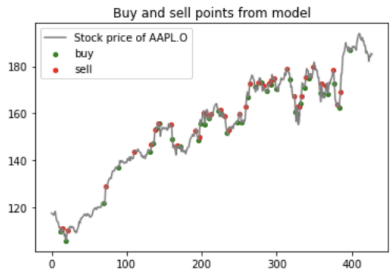
- Base Profit
  - The total profit generated from the strategy based on the initial capital
- Annualised return
  - The average rate of return per year, calculated as the geometric mean of the cumulative returns. It allows traders to compare the performance of different assets using the same trading strategy.
- Annualised volatility
  - A measure of the fluctuation in the price of an asset over the number of trading days in a year. A higher annualised volatility indicates greater price variability and potential risk. Traders may use this metric to compare the risk levels of different investments or to set risk tolerance levels for their portfolio. It is calculated by taking the standard deviation of the daily returns and multiplying by the square root of the number of trading days in a year (usually 252).
- Sharpe ratio
  - A measure of risk-adjusted return, calculated as the ratio of the excess return to the volatility of the strategy's return. It helps traders assess whether the additional return they received from the trade is worth the additional risk taken on for holding a riskier asset. A

higher Sharpe ratio suggests better risk-adjusted performance, while a lower Sharpe ratio may indicate that the investment is not compensating for the level of risk involved.

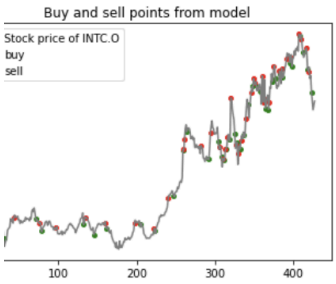
- Maximum drawdown
    - A measure used to assess the risk of an investment. It represents the largest peak to trough decline in the value of an asset during a specified period, giving traders an insight to the potential downside of a trade. A larger maximum drawdown indicates greater volatility and risk, as the trade experienced a significant decline from its peak value.
- Calmar ratio
    - Computed as a ratio of the annualised return of the trading strategy and its maximum drawdown. A higher ratio indicates better risk-adjusted returns, meaning that the trading strategy has generated more returns relative to its maximum drawdown.

Basic Model

	Base Profit	Annualized Return	Annualized Volatility	Sharpe Ratio	Max Drawdown	Calmar Ratio
AAPL.O	1.804559	0.835253	0.14154	5.901196	-0.06088	13.719754

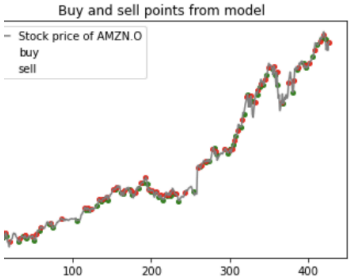


ase Profit	Annualized Return	Annualized Volatility	Sharpe Ratio	Max Drawdown	Calmar Ratio
2.298004	1.01901	0.178808	5.698909	-0.067433	15.11154



	Base Profit	Annualized Return	Annualized Volatility	Sharpe Ratio	Max Drawdown	Calmar Ratio
MSFT.O	2.523972	1.099348	0.13988	7.859202	-0.023345	47.09203

Base Profit	Annualized Return	Annualized Volatility	Sharpe Ratio	Max Drawdown	Calmar Ratio
4.822722	1.821596	0.181761	10.021932	-0.077695	23.445381



	Base Profit	Annualized Return	Annualized Volatility	Sharpe Ratio	Max Drawdown	Calmar Ratio
GS.N	2.536753	1.103828	0.14233	7.755404	-0.037387	29.524367

Average Base Profit: 2.797
Average Annualized Return: 1.194

We achieved satisfactory returns and profits on our base local extrema trading strategy model as evidenced by the high returns and base profit. Of the 5 stocks, AMZN.O generated the best results, with a base profit os 4.8227 and annualized return of 1.822. Conversely, AAPL.O returned the smallest profits, a considerable 1.805 and an annualized return of 0.8352. This could be attributed to the volatility of the stocks, as the AAPL.O stocks have an annualized volatility of 0.14154, which is low compared to AMZN.O. The model's buy and sell points suggest fewer significant price swings to capitalize on, which may result in

lower profits. In contrast, the AMZN.O chart shows a stronger upward trend with more pronounced fluctuations, which are opportunities for a trading model to generate profits. The annualized volatility is 0.181761, slightly higher than AAPL.O, which may indicate more price movement to exploit. In addition, AAPL.O's Sharpe ratio of 5.901196 indicates a decent risk-adjusted return, but the Calmar ratio of 13.719754, while high, suggests the risk-adjusted returns over the maximum drawdown period were not as strong as AMZN.O. In contrast, the Sharpe ratio of AMZN.O is extremely high at 10.021932, indicating that the excess return per unit of risk is very favorable. The Calmar ratio is also outstanding at 23.445381, suggesting a very strong performance relative to the maximum drawdown.

Therefore, from the trends observed in the stock charts, we concluded that our machine learning model used may be better suited to stocks like AMZN.O, which show more pronounced trends and volatility. Since the model is based on local extrema trading strategy, stocks with clear local minima and maxima for the model to trade on offer more opportunities for the model to make profitable trades

.

e. Challenges and Risks of Strategy

The local extrema trading strategy tends to work best in trending markets where prices consistently move in one direction, allowing traders to capitalize on the continuation of trends from one extra point to another.

Lagging Indicators

One of the biggest problems with the local extrema trading strategy is that of lagging indicators, meaning they confirm a trend after it has already begun. As a result, traders may miss out on early stages of a trend or experience delayed signals, reducing the effectiveness of the strategy, particularly in fast-moving markets.

Whipsaws

Additionally, whipsaws can present significant risks in the context of the local extrema trading strategy. Whipsaws refer to the situation in financial markets where the price of an asset moves sharply in one direction and then rapidly reverses couse. When a whipsaw occurs, traders who enter or exit positions based on initial price movements may find themselves on the wrong side of the trade as the price suddenly changes direction, otherwise known as premature entry or exit into trading positions in the market. It can also increase trading costs for traders who engage in high trading frequency in response to short-term price movements. Since whipsaws are difficult to predict and drastic in nature, they often catch traders off guard and can incur significant losses.

False Signals

Lastly, another significant challenge faced by the local extrema strategy is false signal. Not every local high or low indicates a significant trend reversal or continuation. Price movements can often be noisy, leading traders to make incorrect decisions based on false signals. For example, inflection points can also generate false signals, especially in volatile or choppy markets. A temporary change in price direction at an inflection point may not always signify a sustainable trend reversal. This is further exacerbated by local extrema being a lagging indicator which only confirms changes in trend directions after it has begun. Traders may interpret these false signals as genuine opportunities, resulting in losses if the price quickly reverts to its previous direction.

Like all trading strategies, these risks can be mitigated with the implementation of appropriate and effective risk management measures. One prime example of which is setting appropriate stop-loss levels. Defining a predetermined price level at which to exit the trade can prevent traders from making significant losses if the market moves in an unexpected direction in false signals and whipsaws. In the case of delayed signals, it also helps to protect against signals which fail to materialize during the confirmation period. Another mitigation tactic is position sizing where the amount of capital exposed to trade is a predefined percentage of the trader's account balance. By allocating a smaller portion of capital to each trade, traders can mitigate the impact of the risks of local extrema on their overall portfolios.

f. Optimisation and Impact

In order to enhance our model, we did 5 changes to our basic model, namely:

1. Added new features
   - Added MACD
   - Added RSI
2. Trained the model based on the best training data out of the 5
3. If the prediction of the model is lower than a selected probability, do not act on it

New Feature

1. **MACD - SIGNAL:** Moving Average Convergence Divergence, is a popular technical analysis indicator used to identify trends and potential buy or sell signals

```
def getMACD(data, short_window=12, long_window=26):
    short_ema = data.ewm(span=short_window, min_periods=1, adjust=False).mean()
    long_ema = data.ewm(span=long_window, min_periods=1, adjust=False).mean()
    macd_line = short_ema - long_ema
    signal_line = macd_line.ewm(span=9, min_periods=1, adjust=False).mean()

    return macd_line - signal_line
```

2. **RSI :** Relative Strength Index is popular technical analysis indicator used to assess the strength of price movements and identify potential overbought or oversold conditions in financial markets

```
def getRSI(data, window=14):
    delta = data.diff()
    gain = (delta.where(delta > 0, 0)).rolling(window=window).mean()
    loss = (-delta.where(delta < 0, 0)).rolling(window=window).mean()
    rs = gain / loss
    return 100 - (100 / (1 + rs))
```

Enhanced Labelling

MACD - Signal: Check whether the MACD just crosses the signal line

1. Buy/ Long position (1)
   a. When MACD-Signal = 0 and the previous difference is < 0
   b. When MACD-Signal > 0 but the previous difference is < 0
2. Sell/ Short position (-1)
   a. When MACD-Signal = 0 and the previous difference is > 0
   b. When MACD-Signal < 0 but the previous difference is > 0

```
for i in range(1, len(macd)):
    if macd[i] == 0:
        if macd[(i - 1)] > 0:
            macd_call[i] = -1
        else:
            macd_call[i] = 1
    else:
        if macd[i] > 0 and macd[i - 1] < 0:
            macd_call[i] = 1
        elif macd[i] < 0 and macd[i - 1] > 0:
            macd_call[i] = -1
        else:
            macd_call[i] = 0
```

RSI

1. Buy/ Long position (1)
   a. 50 < RSI < 70
2. Sell/ Short position (-1)
   a. 30 < RSI < 50

```
label = []
for i in range(len(rsi_call)):
    call = rsi_call[i] + gradient_call[i] + macd_call[i]
    if call > 0:
        label.append(1)
    elif call < 0:
        label.append(-1)
    else:
        label.append(0)

return pd.Series(label)
```

Labelling: take the sum of gradient, MACD and RSI labelling
If it is positive: Buy (overall 1), if negative: sell (overall -1)
Else hold position (overall 0)

Testing the model on enhanced features

```
e_data = aapl

featureWithLabel = pd.concat((getEnhancedFeatures(e_data), generateEnhancedLabel(e_data)), axis = 1).dropna().re
featureWithLabel.columns = ["grad_diff", "macd_diff", "rsi", "label"]
e_x_train, e_x_test, e_y_train, e_y_test = sequential_split(featureWithLabel.iloc[:, :-1], featureWithLabel.iloc

enhancedModel = sklearn.linear_model.LogisticRegression(max_iter = 5000)

enhancedModel.fit(e_x_train, e_y_train)
```

Using Gradient difference, MACD difference and RSI to train the ML model using logistic regression

Change: Training the Data using the best training model

```
best = []

for i in range(len(stock_list)):
    perf = []
    for j in range(len(stock_list)):
        perf.append(findEnhancedProfit(stock_list[j], stock_list[i], stock_name_list[i])[0])

    best.append(np.mean(perf))

print("{} is the best training data for the five stocks.".format(stock_name_list[best.index(max(best))]))
```

To find the best training model, we found the best training data which gives the average best profits across the 5 stocks. Using this, we found that amazon is the best training model to be used to train all 5 of stocks

Change: If the prediction of the model is lower than a selected probability, do not act on it

```
def selective_pred(enhancedModel, x_test, y_test):
    prob_pred = enhancedModel.predict_proba(x_test)
    y_pred = []

    for prob in prob_pred:
        if max(prob) >= 0.5:
            y_pred.append(enhancedModel.classes_[list(prob).index(max(prob))])
        else:
            y_pred.append(0)

    mets = sklearn.metrics.classification_report(y_test, y_pred, output_dict=True, zero_division=1)
    accuracy = mets['accuracy']
    precision = {key: value['precision'] for key, value in dict(list(mets.items())[:3]).items() if 'precision'

    return y_pred, accuracy, precision

e_y_pred = selective_pred(enhancedModel, e_x_test, e_y_test)[0]
```

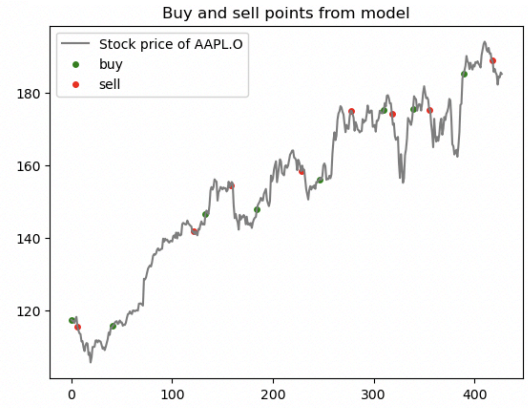
Justification:

Acting on predictions with low probabilities can introduce unnecessary risks. When the model's confidence in a prediction is low (below 50%), there's a higher likelihood of incorrect predictions. By refraining from acting on such predictions, we mitigate the risk of making poor investment decisions that could lead to financial losses.

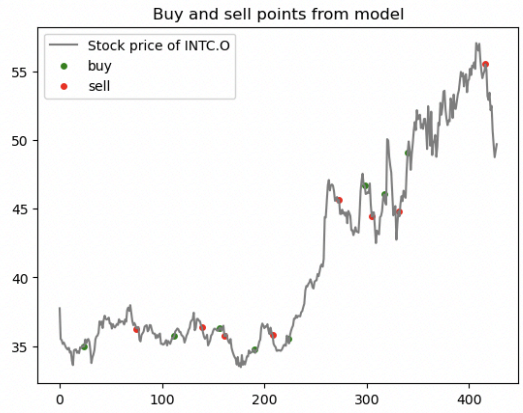
By focusing our actions on predictions with probabilities above 50%, we prioritize those instances where the model exhibits higher confidence. This strategy allows us to allocate resources more efficiently and concentrate on opportunities with greater profit potential.

Evaluation metrics for enhanced model

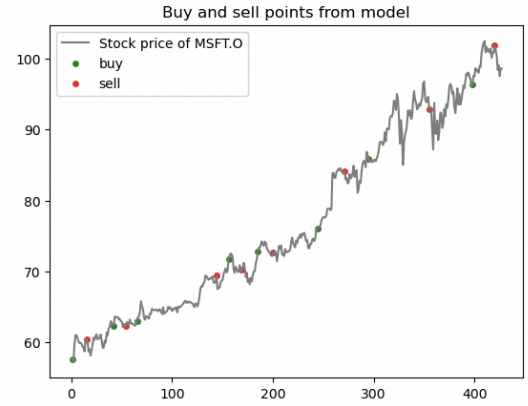
	Base Profit	Annualized Return	Annualized Volatility	Sharpe Ratio	Max Drawdown	Calmar Ratio
AAPL.O	0.409985	0.224208	0.126068	1.778472	-0.067235	3.334714



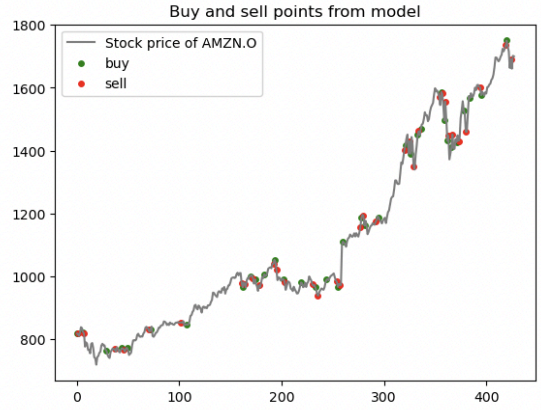
	Base Profit	Annualized Return	Annualized Volatility	Sharpe Ratio	Max Drawdown	Calmar Ratio
INTC.O	0.36137	0.199177	0.20432	0.974828	-0.148929	1.337396



	Base Profit	Annualized Return	Annualized Volatility	Sharpe Ratio	Max Drawdown	Calmar Ratio
MSFT.O	0.29752	0.165735	0.134686	1.230534	-0.104901	1.579925



	Base Profit	Annualized Return	Annualized Volatility	Sharpe Ratio	Max Drawdown	Calmar Ratio
AMZN.O	0.076438	0.044323	0.161982	0.273626	-0.232884	0.190321



	Base Profit	Annualized Return	Annualized Volatility	Sharpe Ratio	Max Drawdown	Calmar Ratio
GS.N	0.266596	0.149296	0.140021	1.066238	-0.121111	1.232719

g. Improvements

Risk Management

Firstly, we could implement advanced Stop-Loss Mechanisms such as trailing stops or time-based stops to mitigate the risks associated with the local extrema strategy.

- Trailing Stops: Automatically adjust the stop-loss level as the price moves in the desired direction. Trailing stops lock in profits while allowing for potential further upside.
- Time-Based Stops: Set predefined time-based stop-loss levels to exit trades if they haven't reached their target within a specified time frame. This helps prevent prolonged exposure to unfavorable market conditions.

Another risk management strategies to help mitigate potential losses is dynamic position sizing. By dynamically adjusting the position size for each trade based on factors such as volatility and market conditions. By optimizing position sizes in response to market dynamics, traders can better manage risk and improve the risk-adjusted returns of their trading strategies.

Hyperparameter Tuning and Cross Validation

We could perform cross-validation and hyperparameter tuning to customise our model in order to better fit our stock data. Cross-validation techniques like k-fold cross-validation to assess model performance more reliably and mitigate the risk of overfitting. On the other hand, hyperparameter tuning can help to optimize model hyperparameters systematically using techniques like grid search, random search, or bayesian optimization to find the best combination of parameters for improved performance of our logistic regression model in our local extrema strategy. By systematically tuning hyperparameters and evaluating model performance through cross-validation, the model becomes better at generalizing to unseen data. This reduces overfitting and ensures that the model captures underlying patterns in the data rather than noise.

Market Regime Detection Techniques

Market regimes refer to different states or conditions of the market, such as trending (bullish or bearish) or range-bound (sideways). Each regime exhibits distinct characteristics in terms of price behavior, volatility, and momentum. Regime detection algorithms analyze historical price data and other relevant indicators to classify the current market regime. These algorithms can range from simple rule-based approaches to more complex machine learning techniques. For example, moving average crossovers, volatility-based indicators, or machine learning algorithms like clustering or Hidden Markov Models (HMM) can be employed to identify market regimes. Once the current market regime is identified, the local extrema strategy can be adjusted accordingly to optimize performance under that specific regime. For example, In a trending market, where price tends to move in one direction with momentum, the strategy may focus on capturing trend-following opportunities by placing more emphasis on breakouts from local extrema points. Conversely, in a range-bound market, where price fluctuates within a defined range, the strategy may prioritize mean-reversion trades, buying at local lows and selling at local highs. Market regime detection can also inform dynamic risk management decisions. During periods of high volatility or uncertainty, risk parameters such as stop-loss levels and position sizes may be adjusted to account for the elevated risk environment. Therefore, by incorporating market regime detection

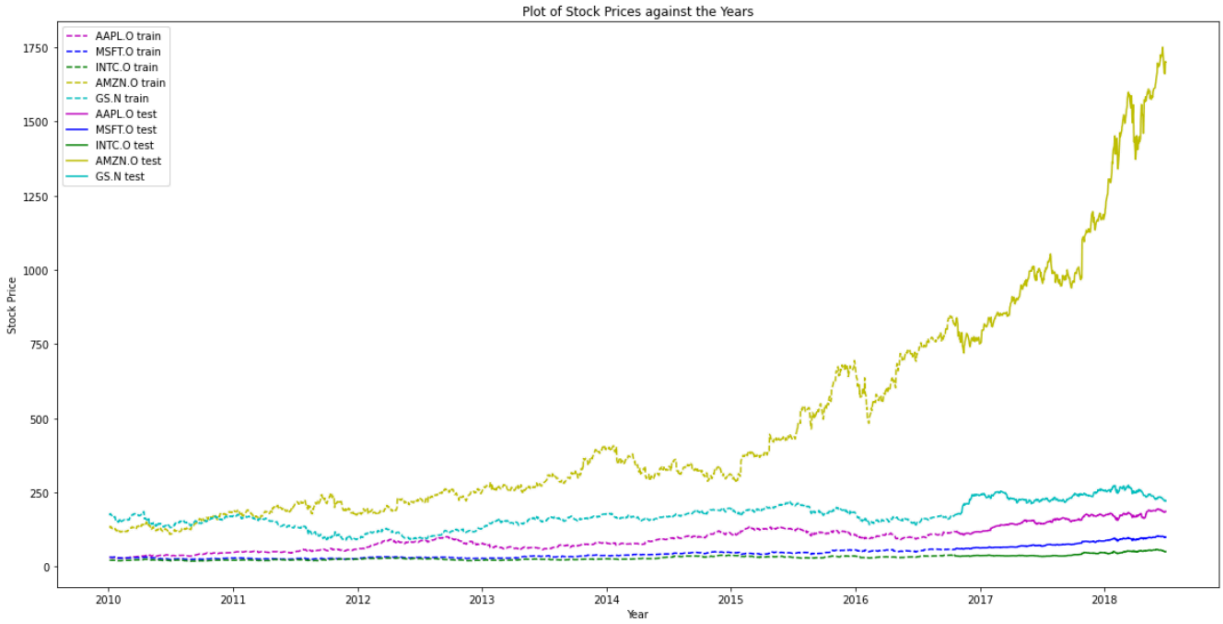
techniques into your local extrema trading model, you can improve its adaptability and performance across various market environments. This enhancement enables the strategy to dynamically adjust to prevailing market conditions, enhancing its effectiveness and risk management capabilities.

Strategy 2: Mean Reversion with Logistic Regression Model

a. Overview of Strategy

Mean reversion is a financial theory suggesting that asset prices and returns tend to move back towards their long-term mean over time. The mean reversion trading strategy is based on the idea that when the price of an asset deviates significantly from its historical average, it is likely to revert back to the mean. It aims to identify assets that are significantly overvalued or undervalued and take positions based on the expectations that it will revert back to the mean.

Justification



The stock price trends are predominantly flat, characterised by stable increases, which aligns with the concept of mean reversion. Most stock prices exhibit increasing trends, except for GS.N, which shows a relatively larger portion of fluctuating and decreasing trends. This observation can provide a focused direction for enhancing the basic model.

b. Logic Behind Strategy

In the raw mean reversion component of the basic model, the next-day positions of stocks are determined using RSI and Z-score as evaluation features. Z-score boundaries are calculated as the difference between the simple moving average (SMA) and the simple moving standard deviation.

```python
def run_mean_reversion(self, stock):
    """
    Run the mean reversion strategy on the stock data.

    Parameters:
        stock (DataFrame): DataFrame of stock prices and features.

    Returns:
        (df, df['position']) (pd.DataFrame, Series): DataFrame with added position column and Series of positions.
    """
    df = stock.copy(deep=True)

    # Run mean reversion strategy
    long_signal = (df['price'] < df['roll_mean'] - df['roll_std']) & (df['rsi'] < 30)
    short_signal = (df['price'] > df['roll_mean'] + df['roll_std']) & (df['rsi'] > 70)

    # 1 : long ; -1 : short ; 0 : otherwise
    df['position'] = long_signal.astype(int) - short_signal.astype(int)
    # End-of-day data is used, position is predicted for the next day, prevent lookahead bias
    df['position'] = df['position'].shift(1)
    df['position'] = df['position'].fillna(0)

    return df, df['position']
```

'Buy' / 'Long' (1) will be signalled on the next day when there is an oversold condition:

i. Current stock price < SMA_20 - Rolling_Std_20 (when Z-score < -1)
ii. RSI < 30

'Sell' / 'Short' (-1) will be signalled on the next day when there is an overbought condition:

i. Current stock price > SMA_20 + Rolling_Std_20 (when Z-score > 1)
ii. RSI > 70

Otherwise, **'Hold' / 'Keep' (0)** will be signalled, no trading on stock is recommended on the next day.

Justification

Z-score :

The Z-Score, a statistical measure, aids traders in quantifying oversold/overbought conditions by assessing how far an asset's price has deviated from its historical mean. These boundaries indicate the intervals during which stock prices are expected to increase or decrease until the next reversion. They assist in identifying optimal trading times to increase the likelihood of higher profits.

RSI :

Complements the Z-score by imposing stricter trade constraints. It is particularly valuable and applicable when stocks experience significant growth or decline within a given interval.

The **logistic regression model** is chosen as the machine learning model for this strategy. The planned input is the derived features while the output should be the predicted positions of the stock, which are classified as

-1 : sell / short
0 : hold / keep
1 : buy / long.

Firstly, features are added to both the train and test datasets in feature engineering. The processed train data with features :

'price_lag1', 'price_lag2', 'price_lag3', 'price_lag4', 'price_lag5', 'roll_mean', 'roll_std', 'rsi'

is fitted into the LogisticRegression model as train input, while **'position'** of train data is fitted as the train output. Then, the processed test data with the same input features will be the test input while the output from the model is the predicted position for the test input.

```
self.features = ['price_lag1', 'price_lag2', 'price_lag3', 'price_lag4', 'price_lag5', 'roll_mean', 'roll_std', 'rsi']
self.model = LogisticRegression(max_iter=self.max_iter)

self.model.fit(self.train_rev[self.features], self.y_train_rev)
```

The stock trading process begins with the predicted (suggested) stock position, using a default base principal of $10,000 (the specific value is less significant as performance is evaluated in percentages). 'Buy All' and 'Sell All' strategies are employed during the stock trading period.

```
for i in range(len(df)):
    curr_price = df['price'].iloc[i]

    if df['position'].iloc[i] == 1 and capital >= curr_price:
        share = capital // curr_price
        shares += share
        capital -= share * curr_price

    elif df['position'].iloc[i] == -1 and shares > 0:
        capital += shares * curr_price
        shares = 0
```

'Buy all':
When a 'long' or 'buy' signal is generated and the available capital exceeds the stock price, the strategy is to purchase as many shares as possible with the remaining capital after each purchase. This continues until the remaining capital is insufficient to buy additional units.

'Sell all':
When a 'short' or 'sell' signal is generated and there are shares in hand, the strategy is to sell all of the shares held.

The current wealth is calculated as:

Current wealth = Current capital + Current number of shares * Current stock price

```
df['current'].iloc[i] = shares * curr_price + capital
```

c. <u>Parameters - Feature Engineering</u>

For the implementation of our strategy, we added the following features to be trained by the model:

- Price lag
- Rolling mean
- Rolling standard deviation
- RSI

```python
# Calculate RSI
df['delta'] = df['price'].diff()
df['gain'] = df['delta'].where(df['delta'] > 0, 0)
df['loss'] = -df['delta'].where(df['delta'] < 0, 0)
df['ema_gain'] = df['gain'].ewm(alpha=0.8, min_periods=self.window).mean()
df['ema_loss'] = df['loss'].ewm(alpha=0.8, min_periods=self.window).mean()
df['rs'] = df['ema_gain'] / df['ema_loss']
df['rsi'] = 100 - 100 / (1 + df['rs'])
```

- Position indicator

```python
long_signal = (df['price'] < df['roll_mean'] - df['roll_std']) & (df['rsi'] < 30)
short_signal = (df['price'] > df['roll_mean'] + df['roll_std']) & (df['rsi'] > 70)

# 1 : long ; -1 : short ; 0 : otherwise
df['position'] = long_signal.astype(int) - short_signal.astype(int)
# End-of-day data is used, position is predicted for the next day, prevent lookahead bias
df['position'] = df['position'].shift(1)
df['position'] = df['position'].fillna(0)
```

d. Performance Analysis

To assess the performance of our trading strategy, we employ the following metrics:

- Base Profit
  - The total profit generated from the strategy based on the initial capital
- Annualised return
  - The average rate of return per year, calculated as the geometric mean of the cumulative returns. It allows traders to compare the performance of different assets using the same trading strategy.
- Annualised volatility
  - A measure of the fluctuation in the price of an asset over the number of trading days in a year. A higher annualised volatility indicates greater price variability and potential risk. Traders may use this metric to compare the risk levels of different investments or to set risk tolerance levels for their portfolio. It is calculated by taking the standard deviation of the daily returns and multiplying by the square root of the number of trading days in a year (usually 252).
- Sharpe ratio
  - A measure of risk-adjusted return, calculated as the ratio of the excess return to the volatility of the strategy's return. It helps traders assess whether the additional return they received from the trade is worth the additional risk taken on for holding a riskier asset. A higher Sharpe ratio suggests better risk-adjusted performance, while a lower Sharpe ratio may indicate that the investment is not compensating for the level of risk involved.
- Maximum drawdown
  - A measure used to assess the risk of an investment. It represents the largest peak to trough decline in the value of an asset during a specified period, giving traders an insight to the potential downside of a trade. A larger maximum drawdown indicates greater volatility and risk, as the trade experienced a significant decline from its peak value.
- Calmar ratio
  - Computed as a ratio of the annualised return of the trading strategy and its maximum drawdown. A higher ratio indicates better risk-adjusted returns, meaning that the trading strategy has generated more returns relative to its maximum drawdown.

Model	Stock	Base Profit	Annualized Return	Annualized Volatility	Sharpe Ratio	Max Drawdown	Calmar Ratio	Profit diff in ML (vs MR)
test_ml	MSFT.O	0.344384	0.200019	0.117124	1.707754	-0.069064	2.896139	0.014486
	AAPL.O	0.297928	0.174297	0.127560	1.366391	-0.098841	1.763413	0.006457
	INTC.O	0.283836	0.166425	0.151717	1.096939	-0.132948	1.251802	-0.082704
	AMZN.O	0.237265	0.140170	0.109824	1.276313	-0.073775	1.899966	0.090749
	GS.N	0.056031	0.034161	0.151562	0.225391	-0.151136	0.226026	0.031010
test_rev	INTC.O	0.366540	0.212166	0.158813	1.335947	-0.119914	1.769314	NaN
	MSFT.O	0.329898	0.192035	0.121079	1.586031	-0.069145	2.777293	NaN
	AAPL.O	0.291471	0.170694	0.125432	1.360843	-0.097922	1.743154	NaN
	AMZN.O	0.146516	0.087893	0.116055	0.757342	-0.112236	0.783111	NaN
	GS.N	0.025021	0.015343	0.154227	0.099485	-0.150506	0.101944	NaN

Table of Evaluation Metrics on Test Data for Basic Model

With reference to the table above, the mean reversion machine learning model performs better than the raw mean reversion on the MSFT.O, AAPL.O, AMZN.O and GS.N test data. The outlier being the INTC.O test data where the raw mean reversion performed better than the mean reversion machine learning model. This could be attributed to the fact that the INTC.O train data has relatively sharp downwards price trends.

Model	Base Profit	Annualized Return	Annualized Volatility	Sharpe Ratio	Max Drawdown	Calmar Ratio	Profit diff in ML (vs MR)
train	0.770534	0.068482	0.190471	0.345463	-0.293122	0.286227	NaN
test_ml	0.243889	0.143014	0.131558	1.134558	-0.105153	1.607469	0.012
test_rev	0.231889	0.135626	0.135121	1.027930	-0.109945	1.434963	NaN

Table of Summary of Average Evaluation Metrics on Test Data for Basic Model

As seen from the table above, the average base profit from the mean reversion machine learning model is +24.39%, 1.2% better than the standard mean reversion model. Overall, the mean reversion machine learning model outperformed the raw mean reversion on the test data, with a higher annualised return at +14.30%, lower annualised volatility and max drawdown at 13.16% and - 10.52% respectively indicating that the model has lower volatility and risk, higher Sharpe ratio at 1.13 and higher Calmar ratio at 1.61 thus suggesting that it generates higher risk-adjusted returns.

Measure	precision	recall	f1-score	support
accuracy	NaN	NaN	0.888998	409.0
keep	0.896209	0.929422	0.912223	255.2
long	0.870319	0.721167	0.784991	42.6
macro avg	0.882115	0.837066	0.855332	409.0
short	0.879816	0.860609	0.868782	111.2
weighted avg	0.889908	0.888998	0.887569	409.0

From the table above, the average accuracy of the model, which is defined to be the proportion of correct prediction (for both true positives and true negatives) among the total number of predictions is 88.90%

e. <u>Challenges and Risks of Strategy</u>

The mean reversion strategy works best in ranging or sideways markets where the price of an asset fluctuates within a range over time. Traders can take advantage of these price movements buying when prices are low and selling when prices are high. In trending markets, prices can continue moving in a single direction, leading to losses for mean reversion traders who are banking on the reversion.

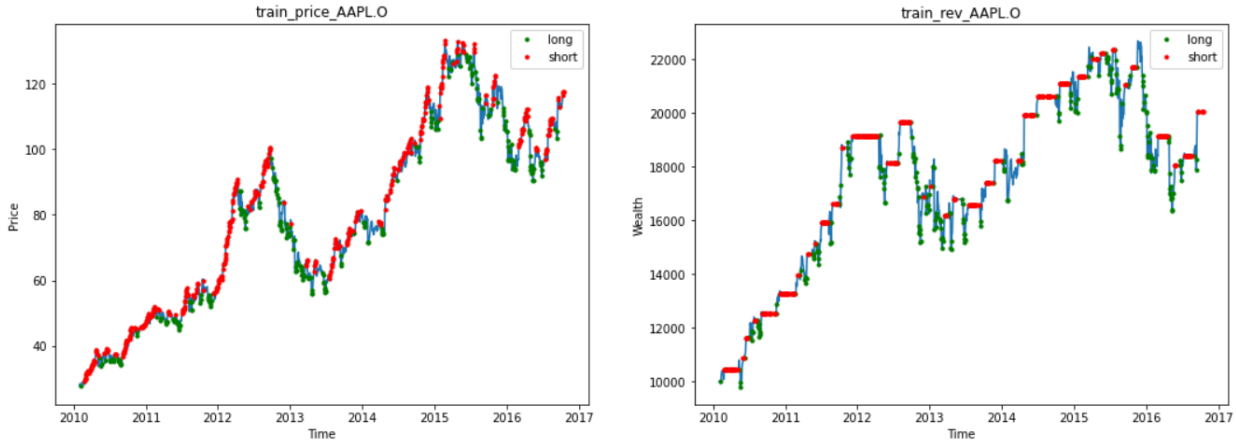
The strategy also faces the risk of prolonged deviation of asset price from mean. In such scenarios, the price of an asset stays significantly above or below its mean for an extended period of time. Traders who employ the mean reversion strategy would face missed potential profits by waiting for a reversion that might not materialise.

Another risk would be its inability to navigate through black swan events. Black swan events are characterised by their extreme and unforeseen nature, often leading to widespread market turmoil and sharp price fluctuations. The sudden shifts in market sentiment, liquidity constraints and increased volatility may lead to heightened levels of investor anxiety and uncertainty, potentially amplifying market downturns. Examples of such events include the 2008 Financial Crisis, which had profound global economic implications, and Black Monday which was a severe and unprecedented stock market crash in 1987. These events typically deviate significantly from historical norms and can disrupt conventional market expectations and risk models. As a result, the mean reversion strategy which relies heavily on historical data will struggle to effectively navigate or mitigate the consequences of black swan events.

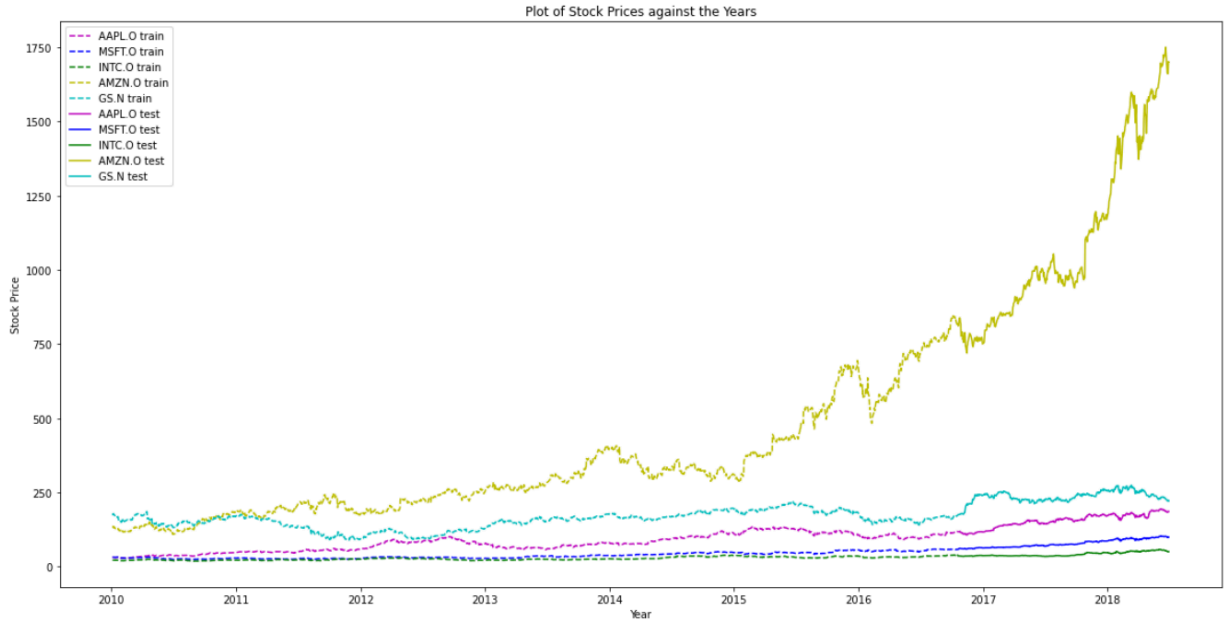
To mitigate these risks and challenges, traders should diversify their portfolio when adopting the mean reversion strategy. Diversifying across different assets can help mitigate the impact of prolonged deviation in any single asset. Furthermore, when adopting the mean reversion strategy, traders can employ risk management techniques such as setting stop-loss orders to help limit potential losses in the event of a downward trend in the stock's price.

f. Optimizations and Impacts

Discovering spaces for enhancement:



The stock positions are visualised by plotting them on the price trends graph and the current wealth graph, using AAPL.O as an example. Early executions of 'short' and 'long' signals resulted in non-optimal returns, particularly for the short signals, which created long idle intervals on the current wealth graph. Therefore, the model could be enhanced from the aspect of Z-score boundaries, which makes use of the Bollinger Bands concept to optimise the moving standard deviation multipliers.



To investigate further, considering the plot of stock price trends, it appears that the available stocks are mostly in increasing trends. Therefore, the model can be more constrained when signalling 'short' or 'buy', as the increasing boundary may be more easily hit in a rising stock price trend. Including the Moving Average Convergence Divergence (MACD) as an additional feature could help overcome this issue.

Furthermore, to create a more optimal model, tests and evaluations on various combinations of window and factor sizes can be conducted.

Include MACD as new features:
MACD utilises short and long SMAs to measure trend momentum, offering a broader view of price trends and helping to prevent premature short positions. It's important to note that the MACD signal is only incorporated into the short signal decision, given that most stocks exhibit increasing trends and to optimise for the specific stocks being analysed.

The constraint
MACD signal (Exponential moving average EMA on signal window) >= MACD (Difference in EMA_short and EMA_long)
is considered in addition to the constraints in the basic model.

```
def add_macd_feature(self, stock):
    df = stock.copy(deep=True)
    short_ema = df['price'].ewm(span=self.macd_short_window, adjust=False).mean()
    long_ema = df['price'].ewm(span=self.macd_long_window, adjust=False).mean()
    df['macd'] = short_ema - long_ema
    df['macd_signal'] = df['macd'].ewm(span=self.macd_signal_window, adjust=False).mean()
    return df
```

```
long_signal = (df['price'] < df['roll_mean'] - self.long_f * df['roll_std']) & (df['rsi'] < 30)
short_signal = (df['price'] > df['roll_mean'] + self.short_f * df['roll_std']) & (df['rsi'] > 70) & \
               (df['macd_signal'] >= df['macd'])
df['position'] = long_signal.astype(int) - short_signal.astype(int)
```

Optimization test using grid search:
Different test ranges on attributes

```
sma_window = np.arange(10, 30, 10)
short_window = np.arange(10, 20, 5)
long_window = np.arange(20, 40, 10)
signal_window = np.arange(3, 9, 2)
long_f = np.arange(1.6, 1.8, 0.05)
short_f = np.arange(1.2, 1.4, 0.05)
```

i. sma_window (window for SMA and SM-std in Z-score)
ii. short_window, long_window, signal_window (windows for MACD)
iii. long_f, short_f (multipliers to create asymmetric Bollinger Bands in Z-score)

are initialised and assigned into combinations. The average test profits from the 5 stocks using each combination are recorded, and the combination that provides an average optimal yield is selected. Additionally, correlations between features and decisions are measured, and most features included in the machine learning model have a correlation greater than 0.
sma_window=10, short_window=10, long_window=20, signal_window=5, long_f=1.7, short_f=1.2

Logistic Regression model features update:
Both MACD and MACD signals are included as features in the machine learning model.
Current features:

```
self.features = ['price_lag1', 'price_lag2', 'price_lag3', 'price_lag4', 'price_lag5', 'roll_mean', 'roll_std', 'rsi', \
                 'macd', 'macd_signal']
```

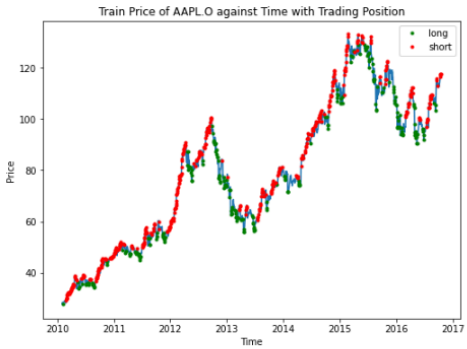
'price_lag1', 'price_lag2', 'price_lag3', 'price_lag4', 'price_lag5', 'roll_mean', 'roll_std', 'rsi', 'macd', 'macd_signal'

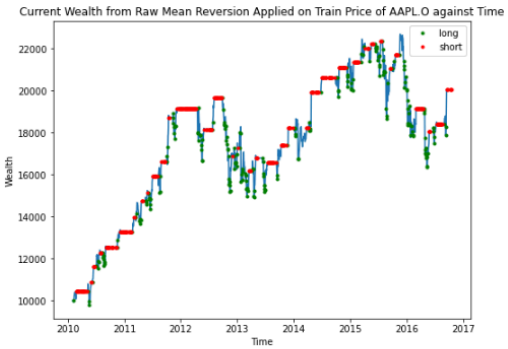
Optimization and Impacts

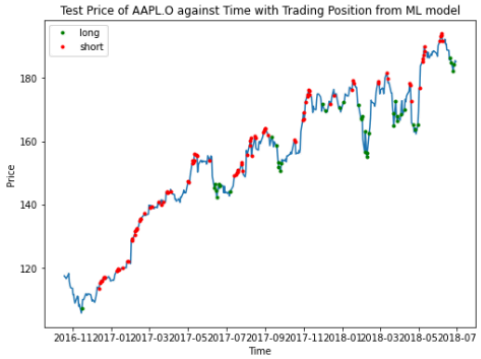
Example:

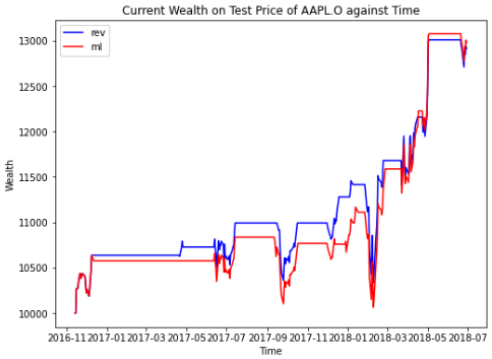
AAPL.O

Basic model

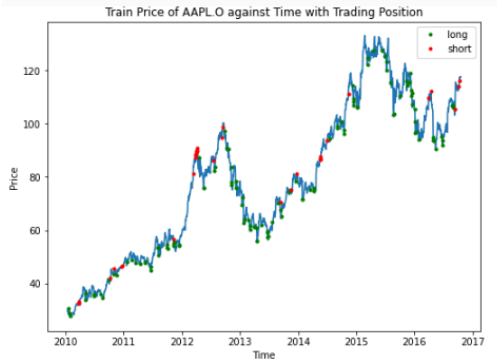


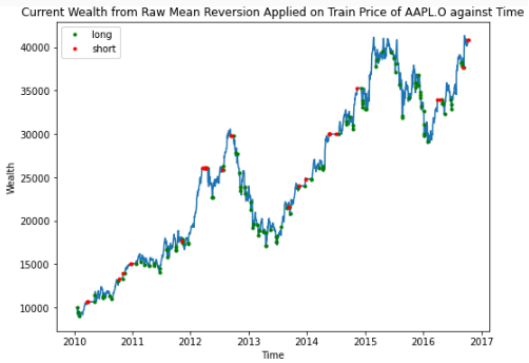


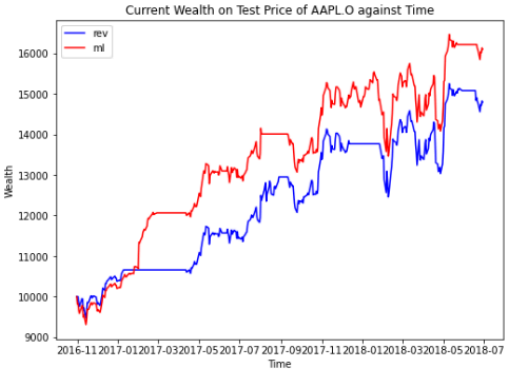




Enhanced model





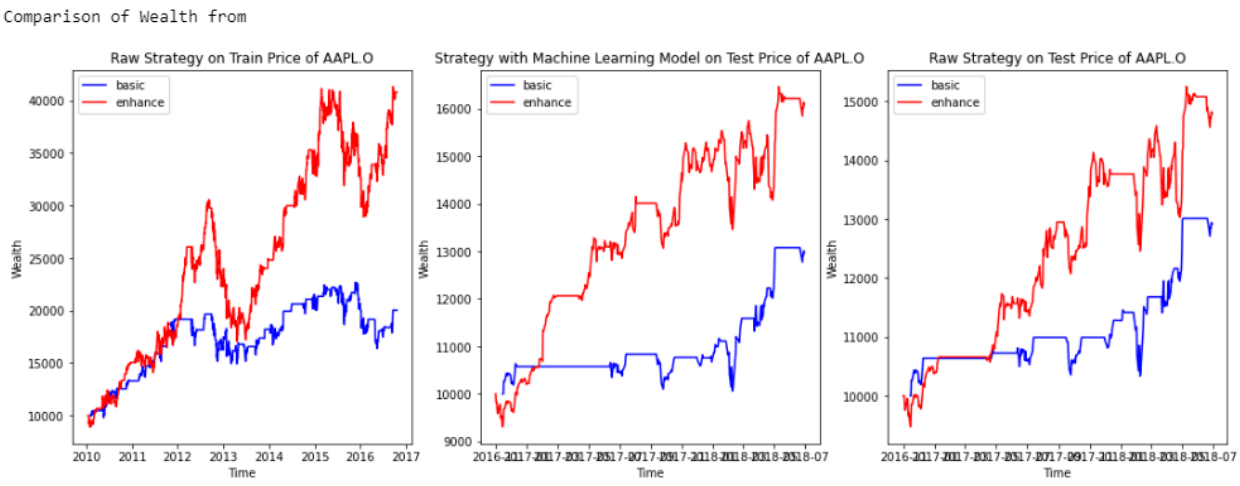


By applying the raw enhanced mean reversion strategy to the train data, the lengths of short intervals decreased. Most of the short options were considered only when they were near some local peaks. This resulted from the tighter constraints applied in the short signal decision boundary. Additionally, by tuning the standard deviation factor in the long signal decision, the number of long signals was also reduced. Long signals were generated when they were located further away from the local peaks, thereby constraining early signals.

Moving to the test data, there was a significant decrease in the number of signals, but it generated a higher amount of profit (higher wealth in a shorter period with the same base capital=10,000 and shares=0). More suitable positions of shorts and longs were observed in the enhanced strategy with the machine learning model on the test data. Short positions were mostly predicted to be around the local peaks, while long positions were predicted around the local valleys. This supported the model to be closer to the concept of 'Buy Low, Sell High', which is a general approach to generate higher profit.

Furthermore, by plotting the current wealth curves of raw enhanced mean reversion and enhanced mean reversion with logistic regression, both on test data, together, the machine learning model performs much better than the raw strategy. This is because the test data has a similar growing trend as the train data, which provides a better decision parameter (the logistic regression model with selected features) that aids the strategy in making predictions with the help of historical trends to yield a higher profit.

AAPL.O - Effectiveness of Optimization

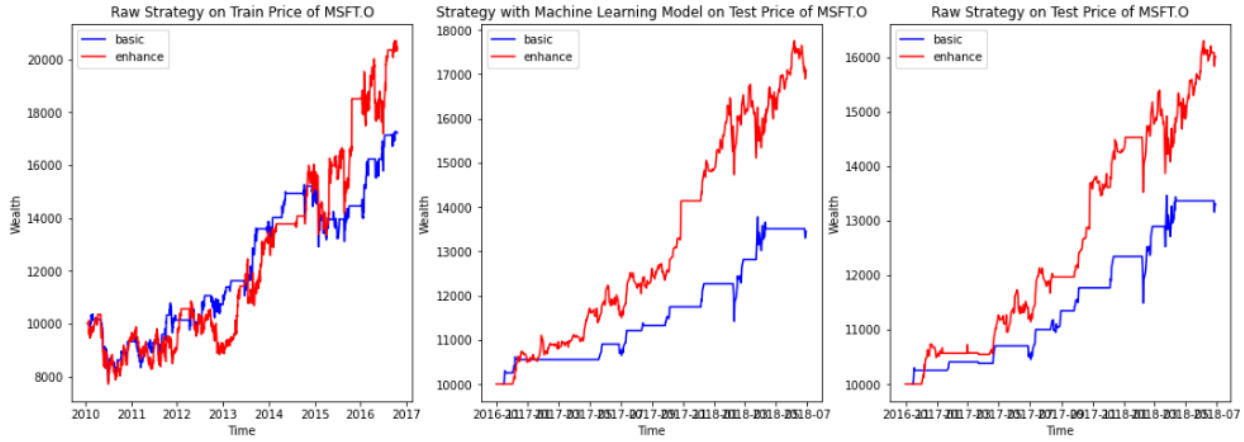


Based on the comparison graph between the basic models and the enhanced models on different datasets (train data, test data with machine learning and test data with raw strategies), the enhanced models showed significant increases in wealth (profits / returns). These analysis results indicate that the optimizations made are effective for the AAPL.O stock.

This situation can be applied to MSFT.O and AMZN.O as well.

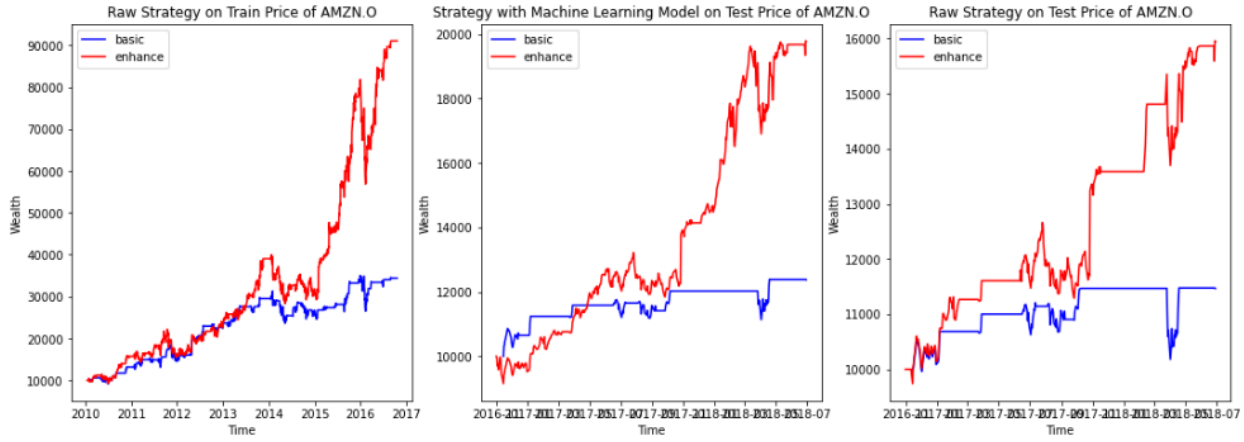
MSFT.O - Effectiveness of Optimization

Comparison of Wealth from

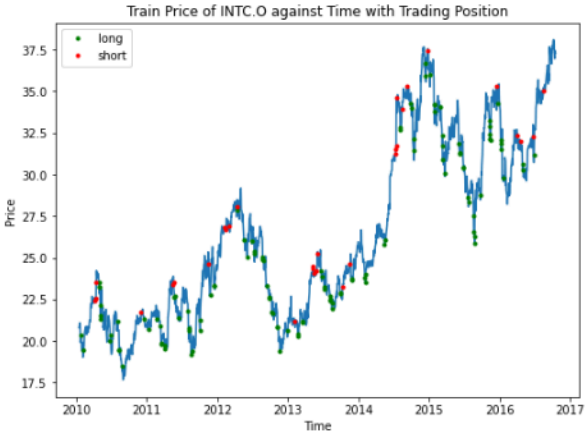


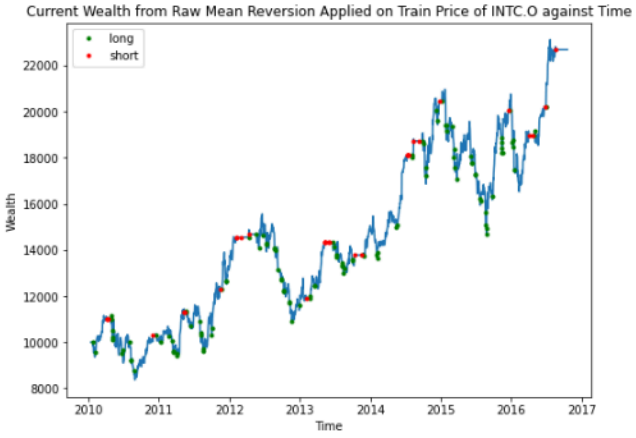
AMZN.O - Effectiveness of Optimization

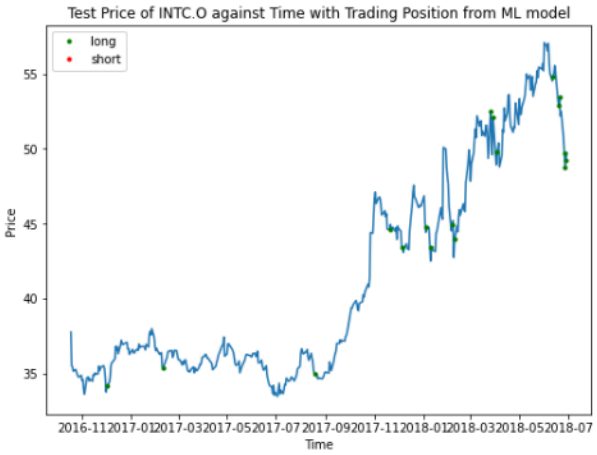
Comparison of Wealth from

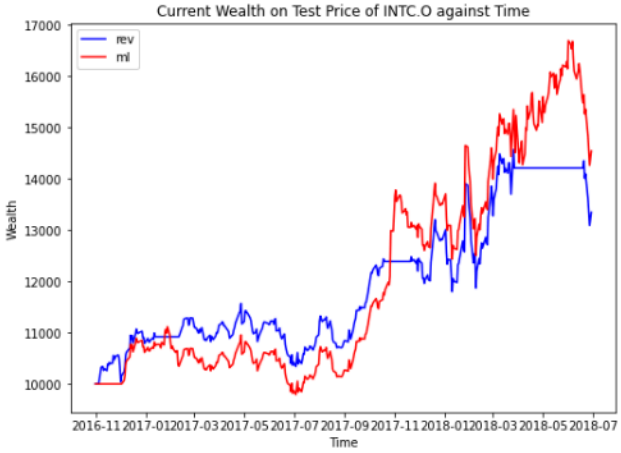


Example: INTC.O
Enhanced Model



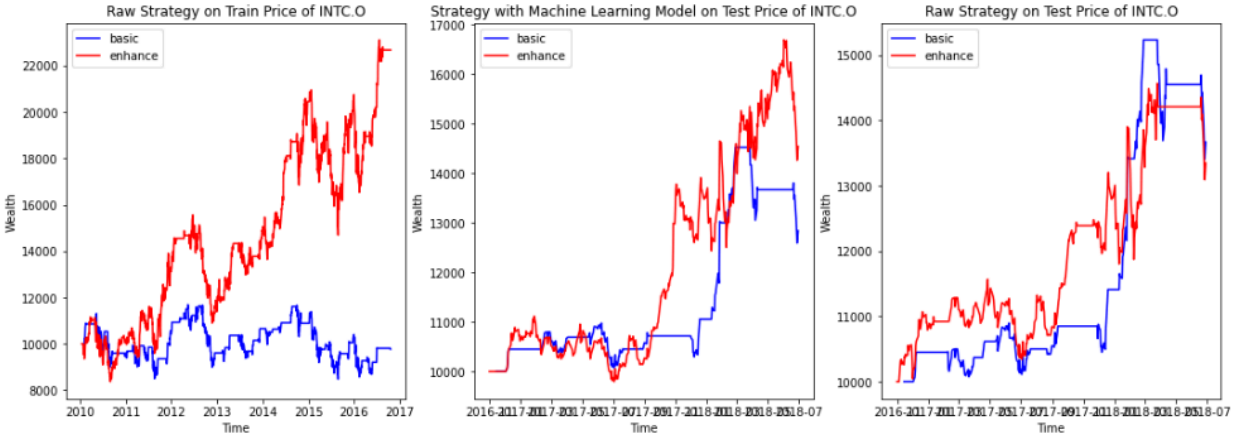






INTC.O - Effectiveness of Optimization

```
Comparison of Wealth from
```

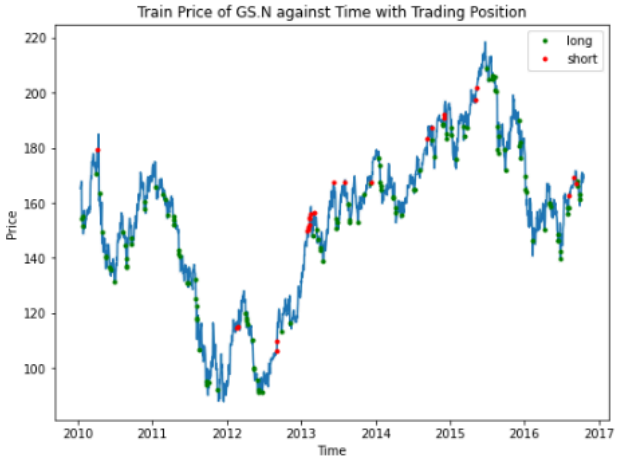


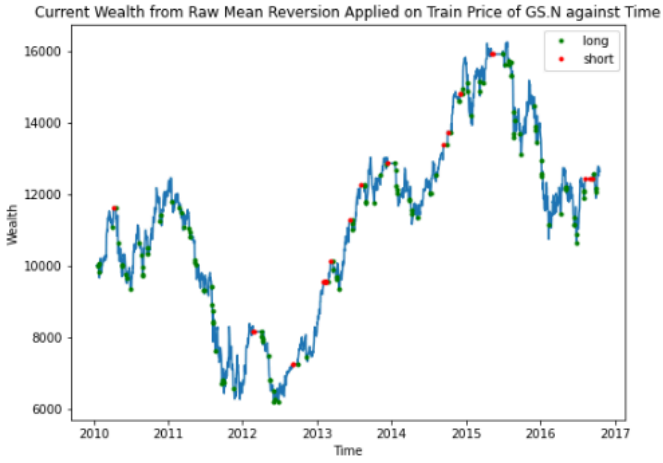
For INTC.O, the basic strategy shows better performance than the enhanced strategy (raw strategies without machine learning modelling). This outcome is because the enhancement is based on the direction of increasing price trends, which may be more optimal for stocks with increasing price trends within the strategy windows and overall. Based on the test data price graphs, INTC.O tends to have smoother growths (not large enough to signal a 'short') and relatively sharp decreases, which slightly deviates from the signal evaluation windows (easily balanced out by historical data and generates relatively normal moving data). Hence, opportunities to generate more wealth are lost due to the universal constraints on the model.

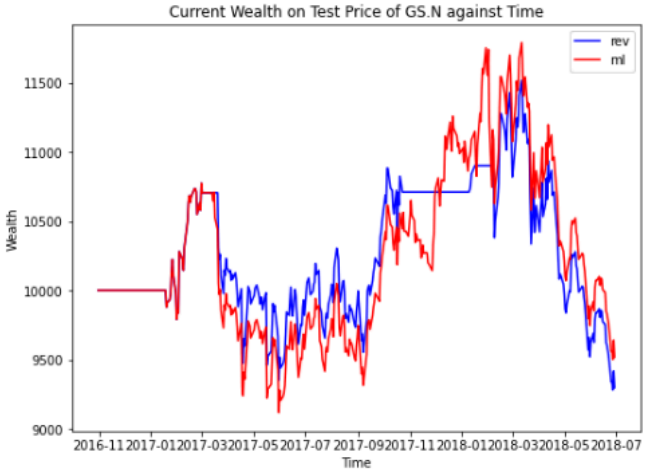
However, when the test data was inputted into the enhanced model with logistic regression, the enhanced model outperformed the basic model. This outcome is because the test data exhibits some similar trends to the train data, including relatively sharp decreases and long intervals of smooth growths (similar level peaks in short windows). The model is trained using these features (price lags), which take the specific characteristics into account and provide better prediction results.

Example: GS.N

Enhanced Model

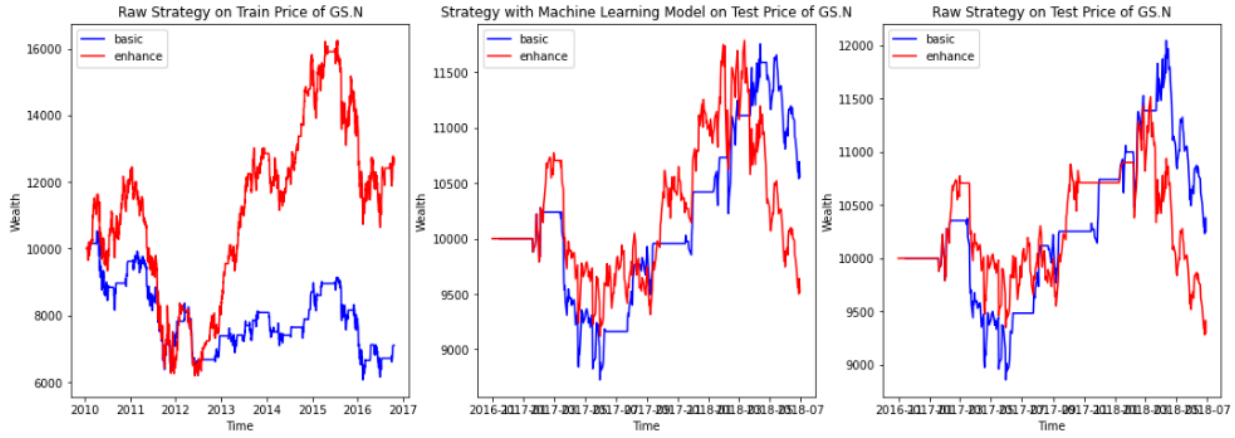






GS.N - Effectiveness of Optimization

Comparison of Wealth from



For GS.N, the enhanced model underperformed the basic model, yielding negative returns for the test data. There are several reasons which could be attributed to this. Firstly, GS.N exhibits a similar trend characteristic to INTC.O, with small window fluctuations. Additionally, GS.N shows an overall decreasing trend near the end periods, which contradicts the enhancement direction. The optimal parameters are selected based on grid search, and the 5 stocks show an average increasing trend, causing the enhanced strategy to make worse position decisions for GS.N.

Moreover, the test data has a very different trend in the first half periods compared to the train data. For example, the stock price increases significantly at the beginning of the test data but experiences an overall decrease at the beginning of the train data. This difference caused the logistic regression modelling to fail to capture the price trend characteristics, affecting the classification results. Therefore, the enhanced model is less suitable for predicting optimal positions in GS.N.

Performance Evaluation Metrics

Trading Strategies

Model	Stock	Base Profit	Annualized Return	Annualized Volatility	Sharpe Ratio	Max Drawdown	Calmar Ratio	Profit diff in ML (vs MR)
test_ml	AMZN.O	0.977381	0.506878	0.222354	2.279603	-0.138458	3.660883	0.383107
	MSFT.O	0.708398	0.380024	0.182047	2.087503	-0.105096	3.615975	0.107612
	AAPL.O	0.609537	0.331425	0.182484	1.816189	-0.133466	2.483216	0.130488
	INTC.O	0.454060	0.252513	0.236002	1.069959	-0.146113	1.728200	0.119522
	GS.N	-0.048203	-0.029276	0.197911	-0.147924	-0.194049	-0.150868	0.021997

Table of Evaluation Metrics on Test Data

Model	Base Profit	Annualized Return	Annualized Volatility	Sharpe Ratio	Max Drawdown	Calmar Ratio	Profit diff in ML (vs MR)
test_ml	0.540235	0.288313	0.204160	1.421066	-0.143436	2.267481	0.152545

Table of Average Evaluation Metrics on Test Data

From the average results, the enhanced model with logistic regression yields positive base profit (+54.02%), annualised return (+28.83%), and profit different from the enhanced model with logistic regression with respect to raw enhanced strategy model (+15.25%). Its Sharpe ratio (1.421) and Calmar ratio (2.267) are also larger than 1. This indicates that the model is effective in producing profit overall.

Model	Stock	Base Profit	Annualized Return	Annualized Volatility	Sharpe Ratio	Max Drawdown	Calmar Ratio	Profit diff in ML (vs MR)
test_ml	AMZN.O	0.740116	0.366708	0.112529	1.003290	-0.064683	1.760917	0.292358
	MSFT.O	0.364014	0.180005	0.064923	0.379749	-0.036032	0.719836	0.093126
	AAPL.O	0.311609	0.157128	0.054924	0.449797	-0.034625	0.719803	0.124031
	INTC.O	0.170224	0.086088	0.084285	-0.026980	-0.013165	0.476398	0.202226
	GS.N	-0.104234	-0.063437	0.046349	-0.373315	-0.042912	-0.376894	-0.009013

Table of Performance of Enhanced Model w.r.t Basic Model on Test Data

Model	Base Profit	Annualized Return	Annualized Volatility	Sharpe Ratio	Max Drawdown	Calmar Ratio	Profit diff in ML (vs MR)
test_ml	0.296346	0.145298	0.072602	0.286508	-0.038283	0.660012	0.140546

Table of Average Performance of Enhanced Model w.r.t Basic Model on Test Data

From the average results, compared to the basic model, the enhanced model yields 29.63% more base profit, a 14.53% higher annualised return, and a 14.05% higher profit difference from the enhanced model with respect to the raw enhanced strategy (the enhanced machine learning model works better than the basic machine learning model). Its Sharpe ratio and Calmar ratio also increase by 0.2865 and 0.66 respectively. In addition, the max drawdown drops by 3.83%.

Logistic Regression Model

Measure	Stock	precision	recall	f1-score	support
accuracy	AMZN.O	NaN	NaN	0.935561	419
	AAPL.O	NaN	NaN	0.940334	419
	INTC.O	NaN	NaN	0.937947	419
	GS.N	NaN	NaN	0.926014	419
	MSFT.O	NaN	NaN	0.928401	419

Table of Accuracy for the Enhanced Model on Test Data

Measure	precision	recall	f1-score	support
accuracy	NaN	NaN	0.933652	419.0

Table of Average Accuracy for the Enhanced Model on Test Data

Relatively high accuracies (both individuals or average) are measured for the enhanced model, all above 90%.

Measure	Stock	precision	recall	f1-score
accuracy	AMZN.O	NaN	NaN	0.057810
	AAPL.O	NaN	NaN	0.047914
	INTC.O	NaN	NaN	0.052862
	GS.N	NaN	NaN	0.021369
	MSFT.O	NaN	NaN	0.043315

Table of Performance Enhanced Model w.r.t Basic Model on Test Data on Accuracy

Measure	precision	recall	f1-score
accuracy	NaN	NaN	0.044654

Table of Performance Enhanced Model w.r.t Basic Model on Test Data on Average Accuracy

The accuracy scores increase for every stock, which indicates a closer relationship between the machine learning model and the enhanced mean reversion algorithm. On average, there is a 4.47% increase in accuracy.

The increased annualised volatility (+7.26%) and negative profit in GS.N suggest that there is still room for improvement in the stability and universality of the model. However, the significant improvements in the evaluation results on trading returns and logistic regression scores encourage the conclusion that, on average, the enhanced model performs better than the basic model for these five stock

g. Improvements

Taking the enhanced model as sample model for improvement, since it has a better performance:

Enhance Flexibility

The model can be improved to be more suitable for different stocks with various trend characteristics. It can be implemented by:

i. **Inputting different windows and factors parameters when fitting the train data into the model.**

```
def fit(self, symbol, train, window=10, capital=10000, shares=0, risk_free_rate=0, \
        macd_short_window=10, macd_long_window=20, macd_signal_window=5, long_f = 1.7, short_f = 1.2):
```

This method is already implemented in the project code, but optimal base parameters are used to maintain a uniform model for comparison. By using different parameters derived from grid search, the model can be optimised to maximise stock returns for specific stocks.

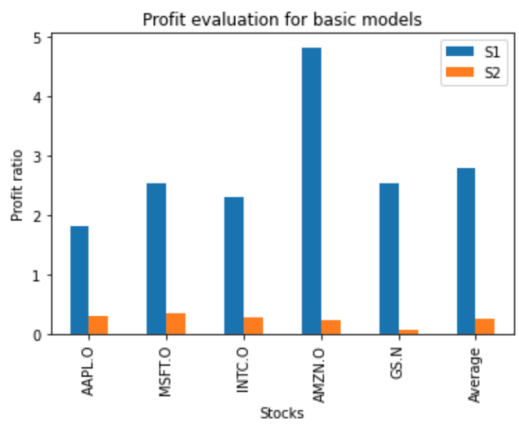
ii. **Integrate parameter optimization approaches into the strategy.**

The parameters are currently evaluated using an external grid search function. Integrating this process into the strategy pipeline would allow for internal execution, tailored to different stocks, without the need to manually find optimal combinations and input parameters. While this provides convenience to users, it introduces computational and time complexity overheads in calculating the optimal parameters as a trade-off.

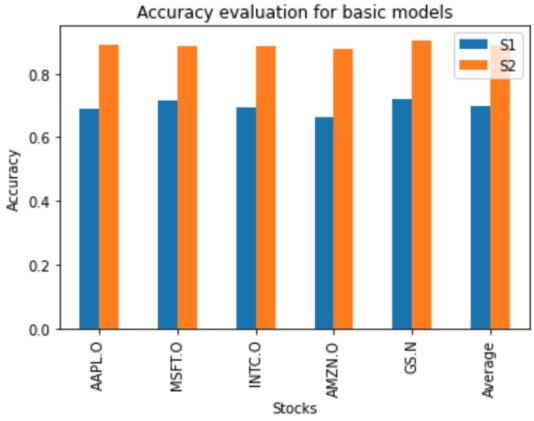
Enhance Stability

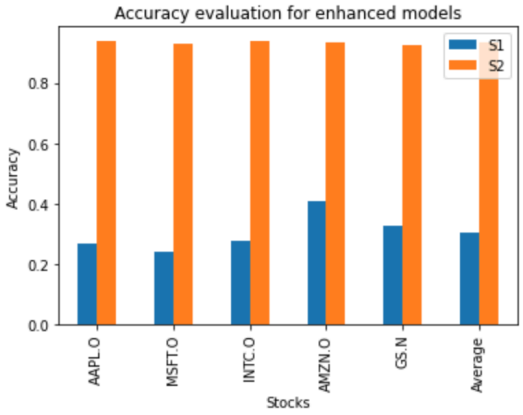
The increase in annualised volatility in the enhanced model may not be negative, as a significant increase in return can also lead to higher volatility, as seen in the wealth comparison graph. However, there are still instances of drastic decrements in wealth contributing to the increased volatility. Therefore, further research on better constraints or additional algorithms for deciding signals could help reduce these large drawdowns.

Comparison of Both Strategies



Comparing between the basic models of local extrema trading strategy and the mean reversion trading strategy we noticed that the local extrema trading strategy was able to generate much greater profits across all 5 stocks , about 3.28 times more than the mean reversion trading strategy as seen by the profit evaluation graph on the left. But after enhancement of the basic models, there was a drastic increase in profits made from the mean reversion strategy, except for GS.N for reasons explained above. Thus, coupled with the decrease in profits from the enhanced model of the local extrema strategy, the enhanced mode for the mean reversion trading strategy outperformed that of the local extreme trading strategy in terms of generating profits.





The accuracy evaluation for the basic models and enhanced models for both strategies were relatively the same. Overall, the mean reversion strategy models had higher accuracy levels than the local extrema strategy models.

Conclusion

In conclusion, from the differing performance on accuracy and profit on both strategies depending on which stock is used to train the model, we learnt that different stocks exhibit unique patterns and behaviours, requiring tailored strategies. The performance variation across different stocks indicates that a one-size-fits-all approach may not be optimal.

References

1. https://medium.com/analytics-vidhya/how-im-using-machine-learning-to-trade-in-the-stock-market-3ba981a2ffc2
2. https://algocraft.xyz/how-to-get-131-return-with-mean-reversion-trading-strategy-from-stock-selection-to-backtesting-c623870adf31
3. https://wire.insiderfinance.io/mean-reversion-strategy-using-python-6dfed9ed988c
4. https://machinelearning-basics.com/mean-reversion-trading-strategy-using-python/
5. https://www.linkedin.com/pulse/algorithmic-trading-mean-reversion-using-python-bryan-chen/
6. https://blankly.finance/list-of-performance-metrics/
7. https://kernc.github.io/backtesting.py/