Research Plan on Algorithmic Trading Strategy

--- Abhinandan Das

**AIM**

Given a 2-dimensional historical time series for market data that stretches back 10+ years, we would like to develop a system that in real time determines a relevant segment of size n from t\_now (meaning the segment is from (t\_now - n) to t\_now), does a pattern match within the database, and gives back a confidence level for the pattern match, along with the most likely projected future move, and the best parameters (e.g. length of time, max-loss, max-profit) to capture this.

**DATA**

10 years market data showing time stamp, price and volume traded. I have created a synthetic data to give an overview of the entire project. The data looks like this: -

A screenshot of a data table

Description automatically generated

As you can see, here we have data from 2014 to 2024 along with their price and volume traded.

Since this data doesn’t provide that much information to perform deeper analysis so I have decided to do some feature engineering and have some more meaningful and impactful data columns.

**Feature Engineering**

20-day Moving Average (MA\_20)*:* This is the simple average of the closing prices over the last 20 days. It smooths out price data to identify the direction of the trend. A rising MA suggests an uptrend, while a declining MA indicates a downtrend.

Relative Strength Index (RSI): RSI is a momentum oscillator that measures the speed and change of price movements. It ranges from 0 to 100 and is typically used to identify overbought or oversold conditions.

MACD (Moving Average Convergence Divergence): MACD is a trend-following momentum indicator that shows the relationship between two moving averages of a security’s price. It consists of a MACD line (the difference between a short-term and a long-term EMA) and a signal line (an EMA of the MACD line). The difference between these two lines (MACD Signal Line Difference) is used to generate buy or sell signals.

Bollinger Bands: Bollinger Bands are a volatility indicator that consists of a middle band (a 20-day simple moving average) and two outer bands (standard deviations above and below the middle band). The upper band is called the Bollinger High, and the lower band is the Bollinger Low. They expand and contract based on market volatility, and price movement near the bands can suggest overbought or oversold conditions.

50-day Simple Moving Average (SMA\_50): This is the simple average of the closing prices over the last 50 days. It is commonly used to identify longer-term trends in the market.

20-day Exponential Moving Average (EMA\_20): The EMA is a type of moving average that gives more weight to recent prices, making it more responsive to new information. The 20-day EMA is commonly used to identify short-term trends.

On-Balance Volume (OBV): OBV is a momentum indicator that uses volume flow to predict changes in stock price. It adds volume on up days and subtracts volume on down days. An increasing OBV suggests that volume is supporting the price rise, while a decreasing OBV indicates that volume is leading the price down.

This is how the data looks like after adding all these features: -

A table with numbers and letters

Description automatically generated

As we can see null values in some of the rows because of the calculations involved, so those are treated next(dropped).

Now, let us look at the market data with these features to understand the visible patterns.

A screenshot of a graph

Description automatically generated

This graph shows how some of the features are closely related. Next, we need to smoothen the data to keep it consistent, so we need to normalize the dataset using minmax scaler. This will map all data within 0 to 1 thus ensuring there are no anomalies or outliers in data and yet retain the actual information from the data. Scaling also makes training the machine learning model less computationally intensive.

Next up is using LSTM model for price prediction. I have used LSTM in this case because it is able to handle long term dependencies and tackle the vanishing gradient problem effectively. GRU is also a good option and a simpler version of LSTM but it is unable to capture complex time series data sometimes, so LSTM is preferred.

**MODEL**

Key insights about the model I have used:-

**LSTM Layers**:

The model includes two LSTM layers. The first LSTM layer has 50 units and is followed by another with 25 units. Both LSTM layers use L2 regularization, which helps in reducing overfitting by penalizing large weights.

**Dropout Layers**:

After each LSTM layer, a Dropout layer with a dropout rate of 0.3 is added to randomly drop 30% of the units during training, further helping to prevent overfitting.

**Output Layer**:

A Dense layer with 1 unit is added at the end to output the final price prediction.

**2. Model Compilation:**

The model is compiled using the adam optimizer and the mean\_squared\_error loss function, which is standard for regression tasks like price prediction.

**3. Training the Model:**

The model is trained on the training data (X\_train\_lstm and y\_train\_lstm) for 10 epochs with a batch size of 32.

Validation is done on the test data (X\_test\_lstm and y\_test\_lstm) to monitor the model's performance on unseen data during training.

**4. Prediction and Smoothing:**

After training, the model predicts prices on the test data (X\_test\_lstm). The predicted prices are then inverse transformed to their original scale using price\_scaler.

A rolling average with a window of 5 is applied to the predicted prices to smooth out fluctuations and make the trend clearer.

**RESULTS**

This is how the final graph looks with predictions.

A graph showing a price

Description automatically generated with medium confidence

The actual prices and the smoothed predicted prices are plotted on the same chart in blue and red respectively.

This plot helps visualize how closely the LSTM model's predictions match the actual prices, with the smoothing providing a clearer trend comparison.

**APPROACH 2**

In this approach I will be using DTW (Dynamic Time Warping) to match segments from current date to historic segments from the data. It is used to measure the similarity between two temporal sequences that may vary in speed. It is particularly useful when comparing time series data that might have temporal distortions, such as different lengths or different speeds of events.

DTW finds alignment between two sequences by "warping" the time axis to minimize the distance between the sequences. DTW is chosen for pattern recognition because of its flexibility in handling time series data that may vary in speed or amplitude. Unlike simple distance metrics like Euclidean distance, DTW can align sequences that are out of phase, making it ideal for financial time series where patterns may not always align perfectly in time. Constraining the DTW window helps in reducing computation time and avoiding unrealistic pattern matches. Here I have used a rolling window of 10 to consider matches within that specific window of time. This means that each point in the current segment can only be compared with points in the historical segment that are within 10 indices of it.

I was able to get the best match using this approach and the output looked like this: -

Best Match Found at Index: 2604

Minimum DTW Distance: 10.396472446450531

Once the match is found, next up was calculating how accurate the match is so for that we had to determine the confidence interval.

Confidence Interval

I have used a random forest regressor model for this purpose and my ouput or the mean squared error came like Model Mean Squared Error: 0.00017049093260849288

This is a very good score which signifies almost minimal error.

Next up was training the LSTM model as usual and then predict the future price movement. Finally, I developed a backtesting strategy to optimize the algorithm to aid in decision making whether to buy or sell based on market scenario to avoid loss. I used parameters like max-profit, max-loss to do so.

**Steps:**

**Initialization**:

position: Tracks whether a position is held or not. Initially set to 0, meaning no position.

entry\_price: Stores the price at which a position was entered.

pnl: Keeps track of the cumulative profit or loss (PnL) from the trades.

**Loop through Predictions**:

The loop starts from the second prediction (i=1) and iterates through the predicted\_prices.

**Trading Logic**:

**No Position (position == 0)**:

* + - If the model predicts that the price will go up (predicted\_prices[i] > prices[i]), the strategy enters a buy position.
    - position is set to 1, indicating that a position is now held.
    - entry\_price is recorded as the current actual price.

**Holding a Position (position == 1)**:

* + - The strategy checks two conditions:
      * If the actual price has reached or exceeded the entry\_price plus a max\_profit, it takes the profit and closes the position.
      * If the actual price has dropped to or below the entry\_price minus a max\_loss, it cuts the losses and closes the position.
    - When a position is closed, the profit or loss from that trade (prices[i] - entry\_price) is added to the pnl.
    - position is reset to 0, indicating that no position is held after closing the trade.

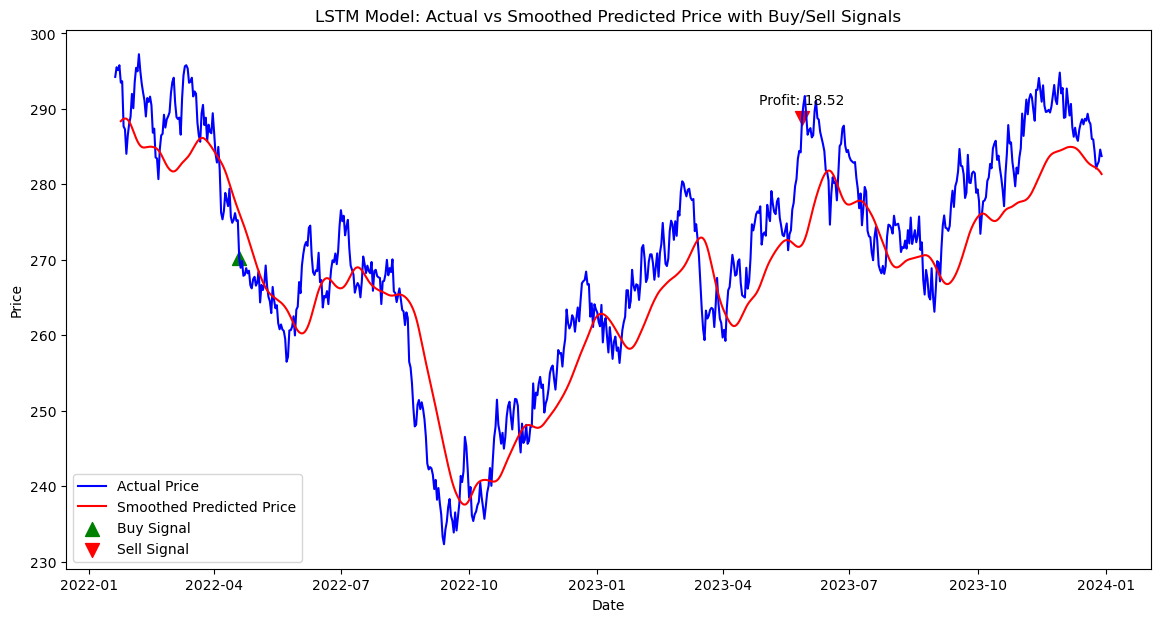
**Return Profit/Loss**:

After looping through all predictions, the function returns the cumulative profit or loss (pnl) from the strategy.

Profit/Loss from Strategy: 32.643955014325854

**Risk Minimization**

Here I have introduced the buy and sell signals based on the RSI indicator implying whether price will go up or down and based on that it will suggest when to buy and when to sell to earn profit.



The profit/loss resulting from each buy/sell pair is annotated next to the corresponding sell signal. In the above figure, there is an annotation indicating a profit of 18.52 units. This means that if I had bought at the buy signal and sold at the sell signal, I would have made a profit of 18.52 units.

ALTERNATIVE APPROACH

Spectogram analysis could have been a good option too for pattern matching but it is mostly used for highly complex data and helps in feature extraction rather than prediction. What it does is it captures the frequency data of a signal over time to understand complex patterns. However, it is computationally intensive so it is not used when dealing with simple patterns.

To perform spectrogram analysis we also need to apply fourier transformation which converts time series signal data to frequencies domain data. The result of the Fourier transform gives complex numbers representing the amplitude and phase of each frequency component. The magnitude of these complex numbers corresponds to the energy of each frequency.

In this approach again we need to add certain indicators like RSI, lower band, upper band to capture more information from the data.

A table with numbers and a number on it

Description automatically generated

The objective of this analysis is to predict future price movements based on historical data. This is achieved using a technique called Dynamic Time Warping (DTW), which is a powerful method for measuring similarity between two sequences that may vary in speed or timing. The analysis also includes spectrogram analysis and Fast Fourier Transform (FFT) to provide additional insights into the frequency components of the time series data.

**2.1 Pattern Matching with Dynamic Time Warping**

The core of the prediction model is based on matching the most recent price sequence (current\_sequence) with historical patterns in the data (historical\_data). The DTW algorithm is used to find the historical segment that most closely resembles the current sequence, even if there are slight variations in timing.

**Key Steps:**

* **Flattening the Sequences:** Both the current sequence and historical data are flattened into 1-dimensional arrays to ensure compatibility with the DTW algorithm.
* **Pattern Matching:** The DTW distance is computed between the current sequence and each possible subsequence of the same length in the historical data. The subsequence with the smallest DTW distance is identified as the best match.
* **Prediction:** The future price movement is predicted based on the segment of historical data that immediately follows the best-matching sequence. Here, I won’t be using any ML model for predictions, instead I will predict the future depending on current mean vs matched segment mean.

To visualize the frequency content of the current sequence, a spectrogram is computed. The spectrogram represents how the frequency of the signal varies over time. In this context:

* **Spectrogram Computation:** The data is split into segments (nperseg), with some overlap (noverlap), and the power spectral density is computed for each segment.
* **Visualization:** The spectrogram is visualized as a heatmap, where the x-axis represents time, the y-axis represents frequency, and the color intensity represents the power of each frequency component.

**2.3 Wavelength Patterns with Fast Fourier Transform (FFT)**

The FFT is used to analyze the frequency domain characteristics of both the current sequence and the predicted future sequence. This helps in understanding the underlying frequency patterns and how they may evolve.

**Key Steps:**

* **FFT Computation:** The FFT of the current sequence and the predicted future sequence are computed, transforming the time-domain data into the frequency domain.
* **Comparison:** The FFT results of the current sequence and the projected future sequence are plotted together, allowing for a visual comparison of their frequency components.

**2.4 Prediction and Decision Making**

Finally, a simple heuristic is used to predict whether the projected future movement indicates a profit or loss:

* **Profit/Loss Indicator:** The mean of the projected future sequence is compared to the mean of the current sequence. If the mean of the projected future sequence is higher, it suggests an upward trend, indicating a potential profit. Otherwise, a potential loss is indicated.

**3.1 Sequence Matching**

The analysis was applied to a sequence of prices over the last 30 days (current\_sequence). The model searched through historical data (historical\_data) to find the most similar sequence and then predicted the next 30 days based on the best match.

**Results**

A diagram of different colored squares

Description automatically generated

A graph with blue and orange lines

Description automatically generated

A graph of blue and orange lines

Description automatically generated

In the above figure, we can see that the model was able to detect the future price movement quite accurately. Along with that I have also displayed profit/loss information for risk management.

Projected Future Price Movement:

[6.03493096e+01 3.57814084e+01 6.47817446e+01 1.22920678e+01

8.88659080e+01 5.03083951e+01 4.49349742e+01 5.85864788e+01

6.24783863e+01 7.17758062e+00 6.82617221e+01 2.41931680e+01

7.13952633e+01 8.22534794e+01 8.03958508e+01 5.52500967e+01

5.20169892e+01 1.42875961e+01 7.75346150e+01 2.71409380e+01

4.96695423e+01 2.84274093e+01 1.33828363e+01 6.29557697e+01

5.43320348e+00 7.48645234e+01 3.17586795e+01 1.34693004e-02

5.11129139e+01 4.68519085e+00]

The projected movement indicates a potential loss.