DEVELOPING MACHINE LEARNING AND DEEP LEARNING STRATEGIES FOR REAL TIME TRADING IN FINANCIAL MARKETS

By Pradeep Kumar Reddy Asi

Student ID: 30118327

Contents

1. ABSTRACT

Financial markets are increasingly characterised by high speed, noise, and non-stationarity, creating significant challenges for forecasting and automated trading. This dissertation conducts a systematic review of machine learning, deep learning, and deep reinforcement learning approaches for real-time and high-frequency trading. The review examines commonly used data modalities such as OHLCV (Open, High, Low, Close, Volume) series, limit order book tensors, technical indicators, and sentiment measures, alongside the feature engineering and label design strategies that underpin predictive modelling. It evaluates a wide spectrum of model families, including tree-based ensembles, sequence models (LSTM, GRU), convolutional and temporal networks, transformer architectures, and DRL agents for execution and market-making tasks. A critical finding is that while advanced DL and hybrid models can capture richer temporal and structural dependencies, simpler ML ensembles often remain competitive due to their robustness, interpretability, and lower latency. DRL shows potential in execution and inventory control, though outcomes remain highly sensitive to simulator fidelity and cost modelling. The review highlights that predictive accuracy alone does not guarantee profitability, and that only evaluations which account for realistic trading frictions such as transactions costs, slippage, and latency can provide meaningful insights into live trading performance.

2. INTRODUCTION

The rapid digitisation of global financial markets has transformed trading into a highly competitive, data-intensive environment. Modern markets operate at sub-second speeds, where price dynamics are shaped by complex microstructural interactions, abrupt regime shifts, and heavy-tailed returns. In such settings, developing models that can forecast short-term price movements and support profitable trading decisions requires approaches that go beyond traditional statistical techniques. Machine Learning (ML) and Deep Learning (DL) provide this capability by capturing non-linear relationships in heterogeneous data sources such as price histories, limit order book states, order flow, and sentiment indicators.

The focus of this dissertation is to investigate the potential of ML and DL models to improve real-time price prediction and trading performance. In particular, the study

builds on technical indicators such as Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), Bollinger Bands, Average True Range (ATR), and Average Directional Index (ADX) to enrich raw market data with momentum, volatility, and trend-based features. A wide range of ML models like Random Forests, Support Vector Machines, Logistic Regression, Gradient Boosting, XGBoost, and LightGBM are examined for their ability to classify market direction and identify profitable signals. In parallel, DL models such as Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Convolutional Neural Networks (CNN) are employed to capture temporal dependencies and structural patterns within financial time series.

A key challenge addressed in this work is the translation of predictive accuracy into sustainable trading profitability. While certain models may achieve strong forecasting performance on historical data, this does not necessarily imply robustness in live markets. Transaction costs, slippage, latency, and order queue dynamics frequently erode apparent gains, underscoring the importance of execution-aware design and realistic back testing. To this end, the project evaluates models using both predictive metrics (accuracy, precision, recall, RMSE) and financial measures such as return on investment, Sharpe ratio, Sortino ratio, and maximum drawdown. Walk-forward validation and cost-sensitive back testing are used to replicate real trading conditions and mitigate the risks of overfitting and look-ahead bias.

The review and experimentation presented here aim to provide a comprehensive perspective on the strengths and limitations of current ML and DL approaches in algorithmic trading. By critically analysing model architectures, input features, and evaluation practices.

3. AIM

The aim of this research is to design, implement, and evaluate machine learning and deep learning models for real-time trading in financial markets, with a focus on building predictive algorithms that can capture patterns in historical market data and generate reliable forecasts for future price movements.

4. OBJECTIVE

4.1 Data Preparation and Feature Engineering

Collect and preprocess historical financial datasets using publicly available sources. Derive technical indicators such as SMA, EMA, RSI, MACD, ATR, and Bollinger Bands to enhance predictive signal strength.

4.2 Model Development

Implement a range of machine learning models (e.g., Ridge, Random Forest, XGBoost, LightGBM) and deep learning models (e.g., LSTM, GRU, CNN).

Explore the impact of model complexity and regularisation techniques in addressing challenges such as multicollinearity, noise, and non-stationarity in financial data.

4.3 Model Training and Optimisation

Apply time-aware train–test splits and cross-validation to avoid data leakage.

Perform hyperparameter tuning using grid search and advanced validation methods to enhance predictive accuracy and generalisation.

4.4 Evaluation and Backtesting

- Assess models using statistical error metrics (MSE, MAE, R^2) and financial performance metrics (Sharpe ratio, Sortino ratio, maximum drawdown, cumulative return).
- Back test strategies on unseen market data to simulate real trading conditions and evaluate risk-adjusted returns.

4.5 Comparative Analysis

Compare machine learning and deep learning approaches to determine their relative strengths and limitations in financial forecasting.

Identify the best-performing model in terms of accuracy, stability, and practical applicability for high-frequency trading.

4.6 Practical Contribution

Provide insights into how advanced AI-driven models can support algorithmic trading strategies.

Highlight the trade-offs between interpretability, computational cost, and predictive power in real-world financial applications.

5. METHODOLOGY

5.1 Overview

This chapter presents the end-to-end pipeline for building, validating, and assessing real-time trading models. The methodology comprises data ingestion and preprocessing, feature engineering (technical and optional sentiment/event signals), label construction, model training across ML/DL/DRL families, walk-forward evaluation with strict temporal causality, and execution-aware back testing with transaction costs and slippage. Figure 3.1 summarises the system architecture.

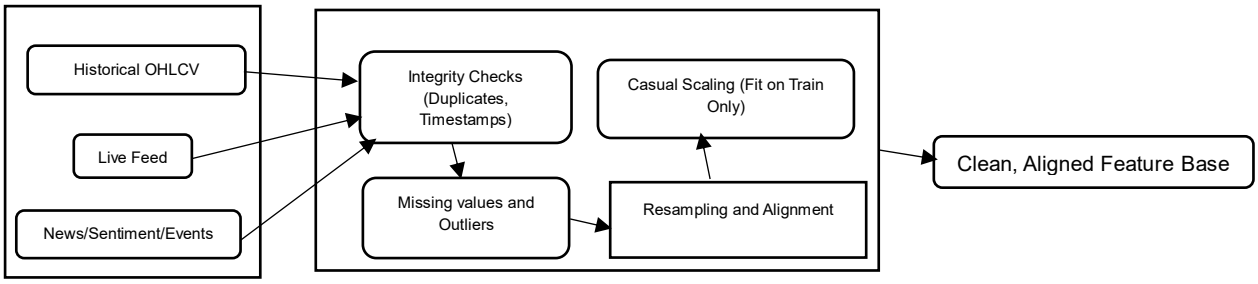

Figure 1: Data Ingestion and Preprocessing Flow

5.2 System Architecture

The pipeline is organised into six stages:

I. **Data Ingestion:** Historical market bars (OHLCV) and alternative data (e.g., news/sentiment) are imported from approved sources.

II. **Preprocessing & Feature Engineering:** Cleaning (missing/outliers), causal scaling, and computation of indicators (RSI, MACD, ATR, ADX, Bollinger

Bands, etc.). Optionally, lagged sentiment/event flags are aligned to market time.

III. **Model Layer:** Multiple model families are trained under identical splits and inputs:
   - **Machine Learning Models:** Logistic Regression, SVM, Random Forest, XGBoost/LightGBM/CatBoost.
   - **Deep Learning Models:** LSTM/GRU (sequence models), CNN/TCN (local patterns), Transformer (long-range context).
   - **Deep Reinforcement Learning Models:** DQN/PPO/A3C for execution/market-making policies (where environment fidelity permits).

IV. **Signal & Risk Controls:** Models output directional probabilities/returns; risk modules apply position sizing, stops, and exposure limits.

V. **Execution & Back testing:** Order generation (market/limit), cost & slippage modelling, and a simulator for walk-forward periods.

VI. **Evaluation & Monitoring:** Predictive metrics (R^2/MSE/MAE or Acc/F1) and risk-adjusted metrics (Sharpe, Sortino, Max Drawdown, Cumulative Return). Drift monitoring triggers periodic retraining.

5.3 Data Sources and Preprocessing

5.3.1 Data Sources

Market bars (OHLCV): Time-barred OHLCV (Open, High, Low, Close, Volume) is the primary substrate for feature engineering and label construction due to its reliability, availability across venues, and compatibility with standard technical indicators (RSI, MACD, ATR, ADX, Bollinger, CCI). OHLCV can also contextualize volatility regimes when combined with microstructure or alternative data.

Alternative data: Global news and sentiment feeds provide orthogonal signals which must be latency-aware: every record is timestamped to the provider/exchange clock, aligned to the relevant bar close (or decision time), and filtered to market hours to prevent staleness. In high-frequency settings, misalignment of alternative streams can introduce label leakage or spurious correlations; therefore, features are constructed from lagged or rolling transforms (e.g., sentiment_lag_k, rolling sentiment volatility) with explicit causal offsets.

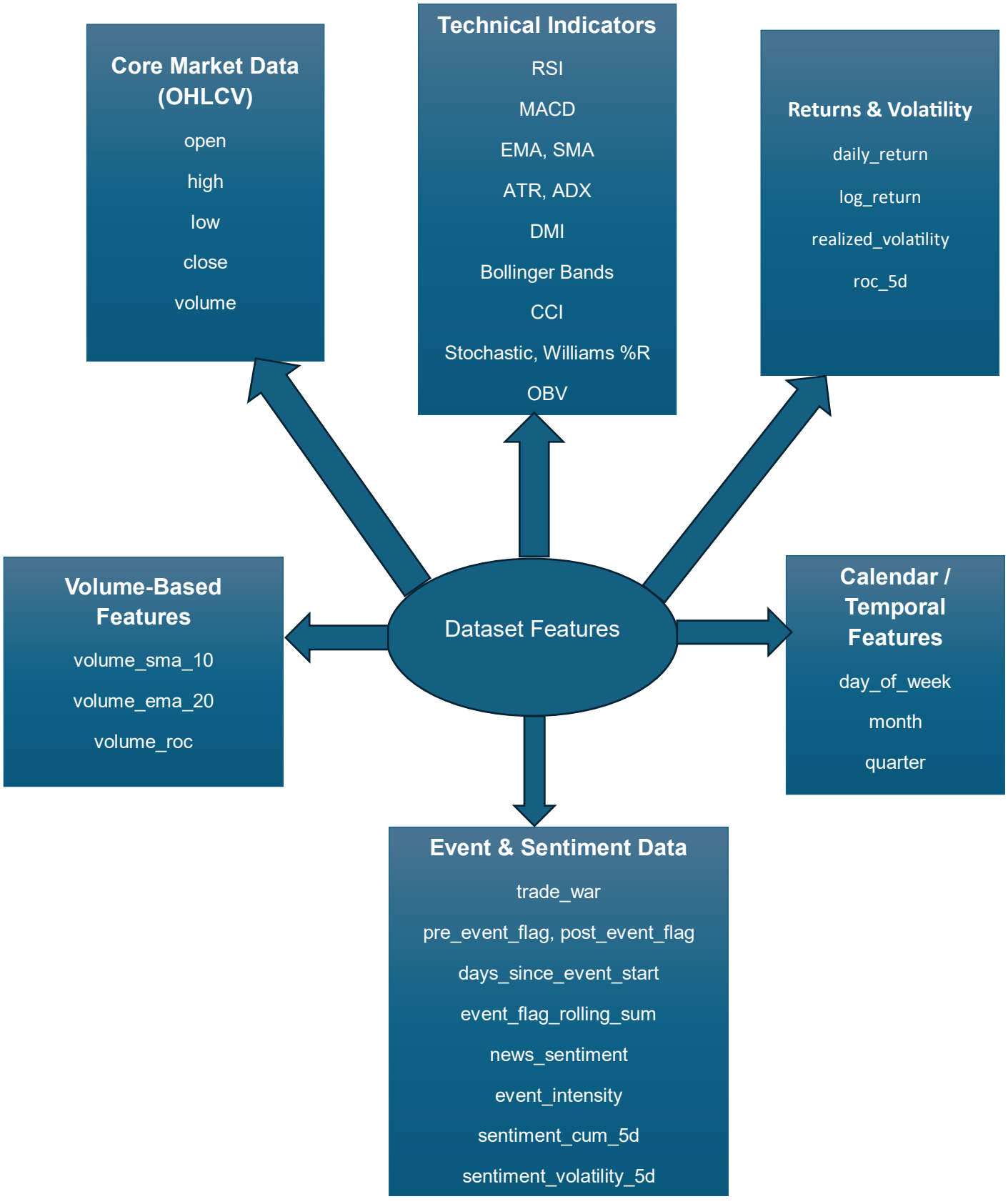

Figure 2: Feature Taxonomy of the trading Dataset

5.3.2 Data Cleaning and Integrity Checks

Ensuring the integrity of high-frequency financial datasets is a prerequisite for building reliable machine learning and deep learning models. Market microstructure data are prone to irregularities such as missing ticks, duplicated records, misaligned timestamps, and spurious price spikes. If left untreated, these artefacts can propagate into derived indicators and distort back test outcomes, leading to misleading estimates of profitability and risk. The cleaning pipeline implemented in this project therefore incorporates systematic procedures for missing value treatment, outlier handling, duplication resolution, and validation checks, with the overarching goal of preserving causality, reproducibility, and robustness.

(a) Missing Values

High-frequency feeds often contain short interruptions due to packet loss or exchange-level dissemination delays. To address this:

- Short gaps (e.g., one or two missing bars) are handled using a forward-fill strategy constrained by a conservative lookback horizon. This ensures continuity of rolling-window indicators such as moving averages and volatility estimates, without spanning across potential regime changes or trading halts.
- Extended gaps (e.g., market halts or vendor outages) are excluded entirely from the dataset. Rolling statistics are reinitialised after the gap to avoid embedding unintended look-ahead bias. No imputation method that uses future observations is employed, since this would compromise the temporal causality required for fair evaluation.

(b) Outlier Detection and Treatment

Financial time series exhibit heavy-tailed return distributions and volatility clustering. Hence, Gaussian-based thresholds for outliers are inappropriate. Instead:

- Outliers are detected using robust statistical measures such as the interquartile range (IQR) and the median absolute deviation (MAD), both of which are less sensitive to fat-tailed behaviour.

- Data points identified as artefactual such as zero or negative prices, or price jumps arising from exchange reporting glitches are winsorised to the nearest valid boundary.
- Genuine extreme observations (e.g., sudden volatility spikes during news events) are retained, as they represent valuable signals of market stress. By doing so, the models are exposed to realistic tail risks rather than smoothed approximations, which is crucial for accurate risk-adjusted performance evaluation (e.g., Sharpe and Sortino ratios).

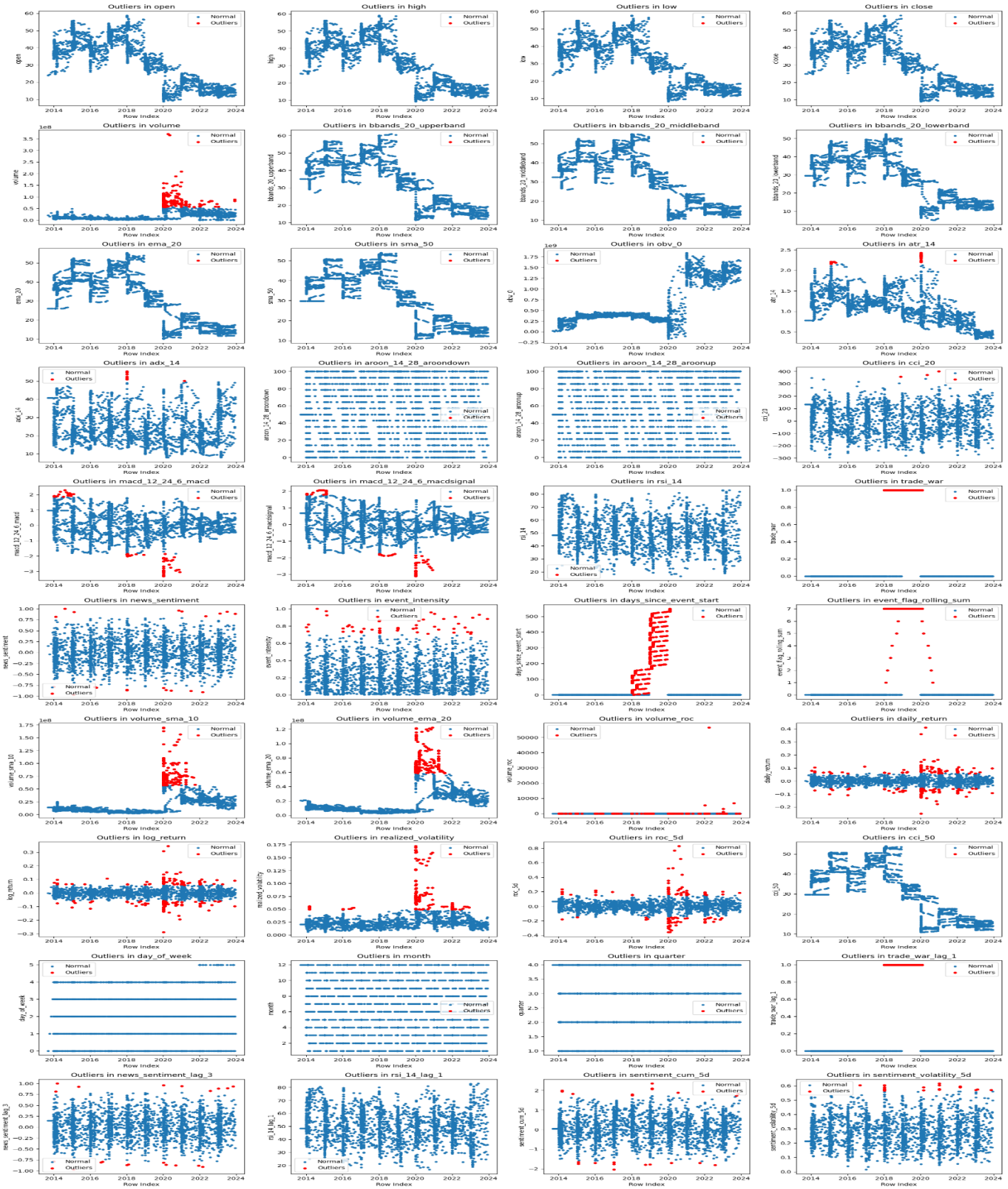

Figure 3: Outlier in each column

Explanation of Outlier Handling in My Dataset

Outlier treatment was designed to ensure that the data remains realistic, avoids artificial distortion of trading signals, and still reflects the true behaviour of financial markets. The key steps are:

I. Removal of impossible values

- Any records with negative or zero prices (open, close) or zero trading volume were eliminated. Such values are not feasible in real markets and most likely indicate data errors or feed glitches.
- This step ensures that the dataset does not contain structurally invalid information that could confuse the models.

II. Winsorisation of fat-tailed features

- Instead of deleting rows with extreme values, I applied *winsorisation*, which caps the values at statistically reasonable thresholds.
- I calculated the interquartile range (IQR) for each feature prone to extreme variation (e.g., volume, daily_return, log_return, realized_volatility). Any observation beyond Q1 – k×IQR or Q3 + k×IQR was capped at the boundary rather than dropped.
- This prevents extreme spikes from dominating model training while keeping the event itself in the time series.

III. Preservation of genuine market shocks

- Financial markets naturally exhibit heavy tails: large jumps often occur around major news, policy changes, or crises. Such movements are not errors but carry important predictive information.
- For this reason, I avoided deleting entire rows when an outlier appeared. Instead, I adjusted only the extreme feature values so that the continuity of the time series is preserved.

IV. Outcome and reproducibility

- The method produces a cleaned dataset that keeps all trading days intact while reducing the influence of extreme, non-representative values.
- The function also records how many values were capped in each feature, which allows full transparency and reproducibility in the data preparation process.

(c) De-duplication and Temporal Alignment

Vendor data feeds occasionally produce duplicated or overlapping bars, typically due to retries in real-time ingestion systems. To mitigate this:

- Each time series is enforced to maintain strictly increasing timestamps with at most one record per bar. Any duplicates are removed and overlapping records are resolved by retaining the first consistent tick.
- All timestamps are normalised to a single reference time zone and aligned with the official exchange trading calendar, accounting for market holidays and daylight-saving adjustments.
- When merging multiple data streams (e.g., OHLCV data with sentiment or event-based features), explicit causal lags are applied to ensure that only information observable up to the decision time is included in feature engineering. This step directly reduces look-ahead bias and supports causal modelling.

(d) Sanity and Integrity checks

Finally, a suite of validation checks ensures logical and financial consistency:

- Cumulative volumes are verified to be non-decreasing within a trading session.
- Bid-ask spreads are enforced to be non-negative, ruling out invalid quoting states.
- Price movements are constrained within feasible tick-size multiples to filter erroneous jumps.
- Abnormally high ratios of quote messages to trade prints are flagged for manual review, as these may indicate vendor artefacts or exchange-level anomalies.

All data processing steps, such as removing records, filling missing values, adjusting extreme values, and aligning timestamps, are carefully recorded. This makes it possible to reproduce the dataset exactly in future experiments, including walk-forward testing. By following these practices, the cleaned dataset gives a more accurate view of real market behaviour and prevents test results from being overly optimistic.

5.4 Feature Engineering

The feature layer is designed to maximise predictive signal per unit latency while preserving strict temporal causality. Indicators are computed on OHLCV with *no* look-ahead; all transforms at time tt rely only on information available by the decision cut-off for bar t. To reduce turnover and cost sensitivity, indicators are combined with robust scaling (fit on train only), winsorisation of extreme artefacts, and neutrality bands that suppress trades on weak signals. This approach reflects best practice reported across ML/DL studies in trading, where feature quality and leakage control are decisive for out-of-sample reliability.

5.4.1 Technical Indicators

A) Moving Averages

Implemented a series of moving averages (MAs) on the closing price of the stock to study the short- and medium-term dynamics of price movement. Moving averages are one of the most widely used tools in financial research because they smooth out short-lived noise in market data and make longer-term trends easier to observe.

Figure 4: Moving Averages of Stock Prices

Four different simple moving averages (SMA) were calculated using rolling windows of 3, 7, 15, and 30 periods.

SMA3 captures very short-term momentum, reacting quickly to local price changes.

- SMA7 represents slightly slower movement, balancing responsiveness with reduced noise.
- SMA15 reflects a medium-term trend, filtering out much of the daily volatility.
- SMA30 serves as a long-horizon baseline, allowing comparison between short-term signals and a broader trend.

These choices create a layered view where the shorter SMAs respond quickly to new information, while the longer ones reveal underlying directional persistence. Such contrasts are often used in trading and forecasting to detect crossovers or divergence between short- and long-term market behaviour.

Main Price Plot

The closing price was plotted in red against these SMAs in different colours. By overlaying all of them on the same axis, we can visually examine how short-term movements deviate from longer-term averages. For example, if SMA3 cuts above SMA30, it suggests short-term bullish momentum, whereas the opposite may indicate weakening prices.

Zoomed Inset

A zoomed inset was added for the period January 2020 to June 2020. This allows the analysis to focus on a six-month sub-interval where the interplay between the close price and its moving averages can be seen in more detail.

- The inset shows the same variables as the main plot but at a finer resolution.
- The area highlighted in the main chart clearly points out where the zoom is applied, ensuring interpretability.

This approach mirrors professional research practice, where global views of the data are complemented with detailed inspections of windows (for example, during market stress events).

The combined chart provides a multi-scale representation of trend behaviour:

- Short-term SMAs (3 and 7) reveal immediate reactions to shocks.
- Medium and long-term SMAs (15 and 30) serve as stabilisers, capturing persistent directions.
- The comparison of short versus long averages is useful in detecting shifts, reversals, or sustained trends, which are crucial signals in high-frequency and algorithmic trading models.
- The inset analysis ensures that periods of abnormal volatility are studied in detail without losing the overall market context.

B) Exponential Moving Average

The exponential moving average is a technical indicator used in trading practices that shows how the price of an asset changes over a certain period. The EMA is different from a simple moving average in that it places more weight on recent data points.

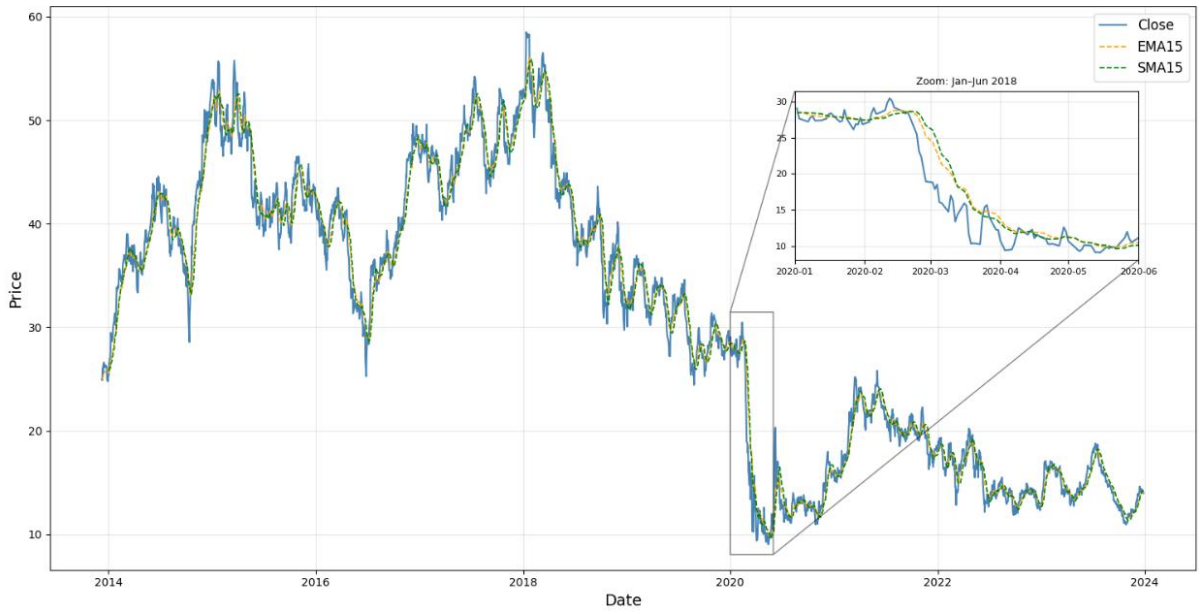

Figure 5: Comparison of EMA15 and SMA15 with Close Prices

In this analysis, the closing price of the stock was compared against two different moving averages: the Exponential Moving Average (EMA15) and the Simple Moving Average (SMA15). Both indicators are widely used to smooth financial time series, but they differ in how they assign weight to past observations.

- The SMA15 takes the average of the last 15 closing prices, treating each day equally. This makes it stable but slower to react to sudden market changes.
- The EMA15 also considers the last 15 days but gives more weight to recent prices. As a result, EMA15 responds faster to new information compared to SMA15, which makes it useful in volatile or rapidly changing market conditions.

By plotting both against the actual closing price, we can observe how quickly each moving average adjusts when price shocks occur.

- Closing price (steel blue line) as the ground truth.
- EMA15 (orange dashed line) and SMA15 (green dashed line) to show trend smoothing.

This comparison highlights whether the exponential weighting provides an advantage in terms of capturing turning points earlier than the simple moving average. A zoomed inset was created for the six-month period **January to June 2020**. This

allows a closer inspection of how EMA15 and SMA15 behave during a specific time window.

- The zoom shows how EMA15 tends to stay closer to the closing price during periods of fast market movement.
- The SMA15, in contrast, lags behind, offering a smoother but slower-moving signal.

The inset window is highlighted on the main chart using connection lines, making it clear which portion of the data is magnified.

It demonstrates the trade-off between responsiveness (EMA15) and stability (SMA15). Traders and machine learning models often face this choice when designing features. For instance, a model that relies on EMA based features may capture short-term dynamics better but might also overreact to noise. SMA-based features, on the other hand, are more conservative but risk missing sudden turning points.

By including both, the feature space becomes richer and more capable of representing different market behaviours. This duality is particularly valuable in high-frequency and algorithmic trading, where regime changes can occur abruptly, and both responsiveness and stability are needed.

C) MACD (Moving Average Convergence Divergence)

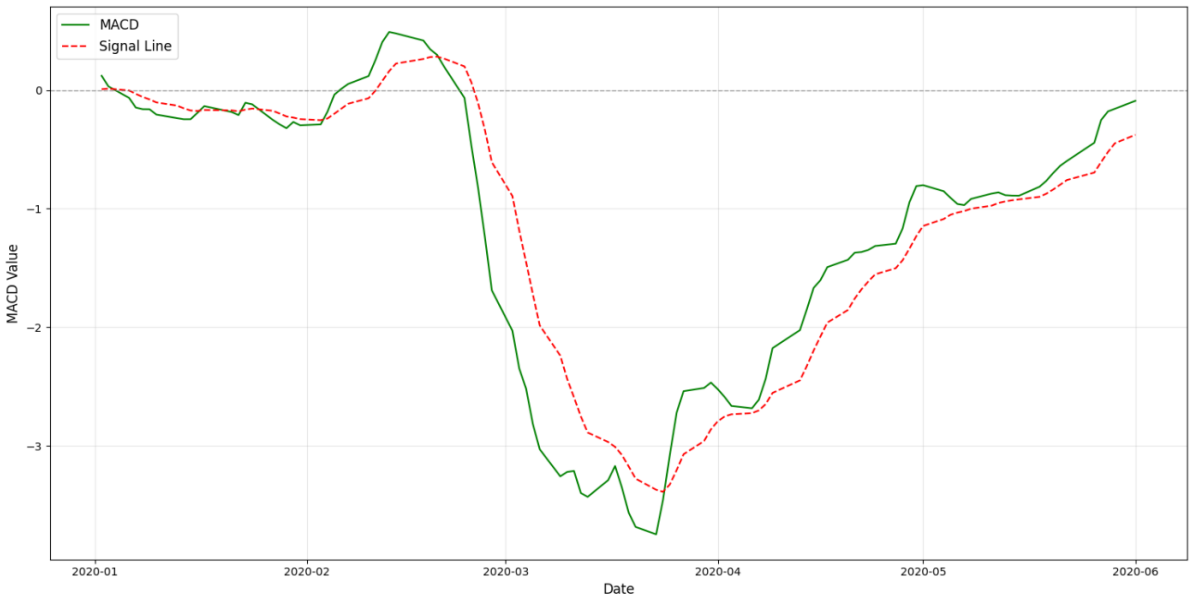

Figure 6: MACD and Signal Line

The Moving Average Convergence Divergence (MACD) was implemented as a technical indicator to capture both momentum and trend shifts in stock prices. The MACD is calculated as the difference between a short-term exponential moving average (EMA12) and a long-term exponential moving average (EMA 26) of the closing price. This difference provides insight into how rapidly prices are changing relative to recent history.

To complement the MACD line, a signal line was computed using a 9-period EMA of the MACD itself. The signal line acts as a smoother reference that traders often use to confirm the strength and direction of the MACD movements.

- When the MACD line is above the signal line, it generally suggests upward momentum.
- When it is below the signal line, it indicates downward momentum.
- The horizontal zero line is important because crossovers above or below zero often mark shifts in the underlying trend.

This dual structure makes the MACD especially useful in financial modelling: it not only measures the distance between short and long-term trends but also highlights turning points.

The chart focuses on the period January to June 2020, allowing detailed inspection of MACD and signal dynamics during that timeframe.

- The green line represents the MACD values.
- The red dashed line represents the signal line.
- A horizontal zero line was added to indicate neutral momentum, where neither bullish nor bearish pressure dominates.

This visualisation makes it easier to detect key crossovers and divergences between MACD and price behaviour during the chosen period.

The MACD provides a more sensitive momentum feature than simple moving averages. It responds faster to shifts in market direction while preserving trend information across different horizons (short-term EMA vs long-term EMA).

Including the MACD and its signal line in the feature set enriches the dataset by providing a continuous, real-valued momentum measure rather than binary trading

rules (e.g., “buy” or “sell” signals). This is especially beneficial for machine learning and deep learning models, which can exploit the size of the divergence between the MACD and signal line rather than just the crossover events.

The use of a zoomed six-month window reflects good practice in local regime analysis: by restricting the view to a turbulent period, the behaviour of the MACD can be studied in contexts where market direction changed rapidly. This helps demonstrate the indicator’s role in distinguishing sustained momentum from short-lived fluctuations.

D) RSI (Relative Strength Index)

The relative strength index is used to show the current and historical strength or weakness of a stock or market based on the closing prices of a recent trading period. The RSI is a momentum oscillator, measuring the velocity and magnitude of price movements.

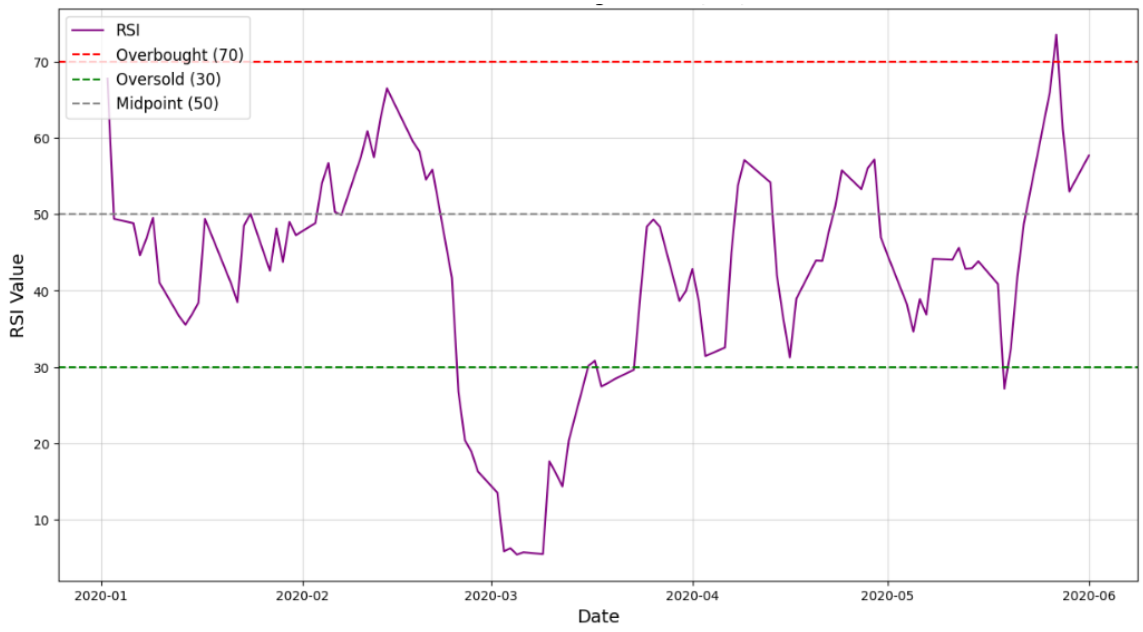

Figure 7: RSI (Relative Strength Index)

This section implements the Relative Strength Index to measure short horizon momentum on the closing price series. The plot shows the computation of one step price change as the difference of the close. Positive changes are kept as gains and negative changes are converted into losses by taking the absolute value, while all other values are set to zero. A fourteen-period rolling mean of gains and a fourteen-period rolling mean of losses are then formed. The strength ratio is the mean gain divided by the mean loss. The RSI is computed as one hundred minus one hundred

divided by one plus the ratio. Missing values at the start of the window are set to fifty so the series begins from a neutral level rather than an arbitrary value. This creates a bounded oscillator between zero and one hundred that is stable to scale changes in the underlying price.

The figure plots the RSI between January twenty and June twenty. The series is shown in purple, together with three reference levels. Seventy marks an overbought zone where recent gains have dominated losses. Thirty marks an oversold zone where losses have dominated gains. Fifty is the midpoint and represents a neutral balance between gains and losses. A grid and labelled axes are added for readability.

From an analytical perspective the RSI condenses recent up moves and down moves into a single dimensionless signal. Readings rising above fifty indicate that average gains exceed average losses and therefore momentum is positive. Readings falling below fifty indicate the opposite. Excursions toward seventy or thirty reveal stronger imbalance and often coincide with short run exhaustion after a strong move. In research use the raw RSI level is more informative than a simple binary flag because it preserves the magnitude of the imbalance. This allows machine learning models to learn how the strength of momentum relates to subsequent returns rather than reacting only to threshold crossings.

Causality is preserved throughout. Every quantity used to compute the RSI at time t uses only prices up to and including time t, and the series used for prediction of t plus one is lagged by one bar in the modelling pipeline. The warmup period from the rolling window is left as missing before the neutral fill at fifty, and no forward filling is applied. This avoids introducing information from the future and supports fair evaluation in the walk forward experiments.

Finally, the RSI is used alongside other indicators such as moving averages and MACD. Together these variables capture complementary aspects of the price process. The RSI focuses on the balance of gains and losses, moving averages describe the underlying trend level and slope, and MACD characterises the separation between fast and slow trends. The joint use of these features creates a richer and more robust representation for forecasting tasks.

E) Bollinger Bands Construction and Interpretation

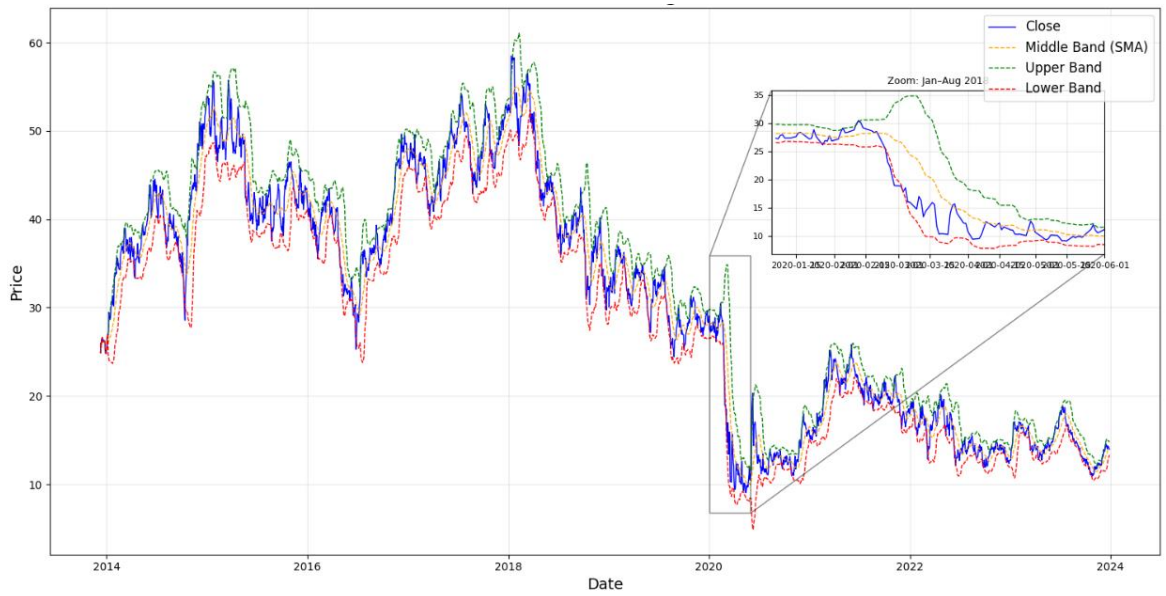

Figure 8: Close Price with Bollinger Bands

Bollinger Bands were applied to the closing price series to capture both central tendency and short-term volatility. The methodology begins by computing a 20-period simple moving average of the close, which serves as the middle band. Around this average, two envelopes are constructed at plus and minus two standard deviations of the closing price, calculated over the same rolling window. These upper and lower bands adaptively widen or narrow depending on how volatile the market is. A wider band indicates high price dispersion, while a narrower band signals relative calm.

In order to avoid missing values at the beginning of the window, the upper and lower bands are initialised with the current close price until sufficient history is available. This ensures continuity of the feature for subsequent modelling.

The figure presents the close series with the three bands superimposed. The main plot shows the long-term dynamics, while an inset zooms into the period from January to June 2020. The inset provides a clearer view of the way the bands contract during low-volatility phases and expand sharply during turbulent intervals. The visualisation highlights how price frequently oscillates between the two outer bands, while the middle band acts as a moving equilibrium.

From an analytical perspective, Bollinger Bands serve two purposes in this project. First, they offer a direct measure of volatility that reacts in real time to changing

conditions, complementing other risk indicators such as the Average True Range. Second, they provide relative positioning of the price within its recent distribution. When the close approaches the upper band, the asset is trading toward the high end of its short-term range; when it nears the lower band, it is trading toward the low end. Machine learning models benefit from these continuous features because they encode both directional and volatility information without resorting to rigid thresholds.

The implementation respects causality by ensuring that the bands at time t are calculated exclusively with data available up to t. No future information is introduced, making the features safe for use in predictive modelling and back testing within the dissertation framework.

F) Daily Price Change Analysis

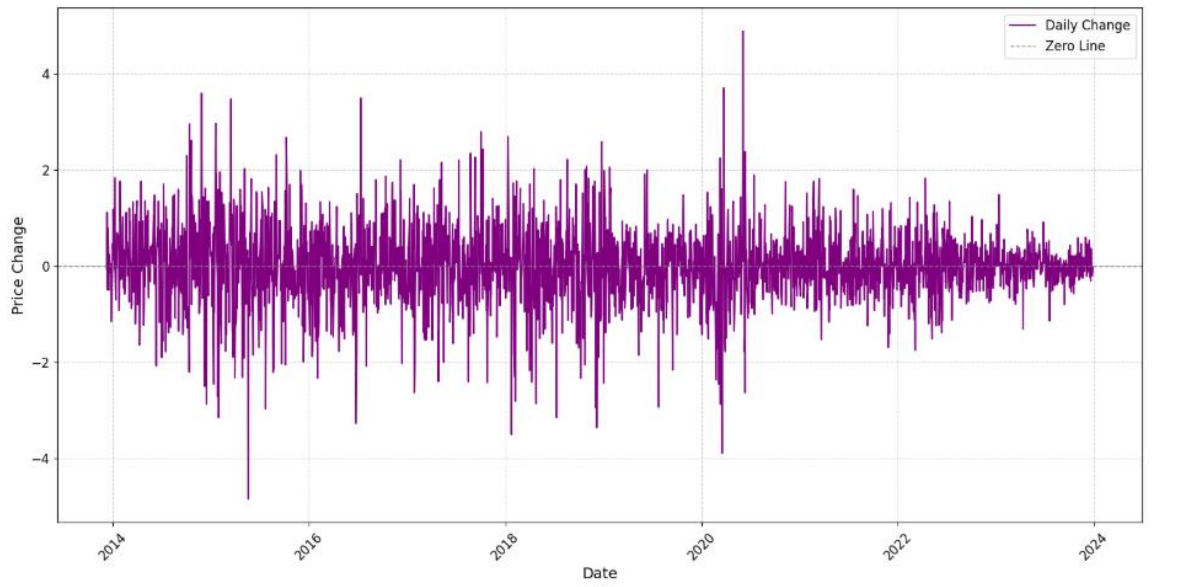

Figure 9: Daily Changes in AAL Stock Price

To quantify short-term movements, the day-to-day change in closing prices was computed by differencing the close series. This transformation creates a new variable that captures the absolute shift in value from one trading day to the next. Missing values arising from the first observation were replaced with zero to maintain a continuous sequence.

The plot visualises these daily changes across the full sample. Positive values indicate days where the stock closed higher than the previous day, while negative values

correspond to declines. A horizontal zero line is included as a reference, allowing upward and downward moves to be compared symmetrically.

This representation is important for two reasons. First, it provides an intuitive picture of volatility, highlighting not only the direction of moves but also the magnitude of short-term shocks. Large spikes above or below the zero line often correspond to events or announcements that triggered abnormal trading activity. Second, by explicitly isolating the daily change, it becomes possible to examine return dynamics independently of the long-term price level, which is crucial when designing machine learning features for predictive modelling.

Overall, the daily change series functions as both a diagnostic tool to visualise turbulence and a foundational building block for constructing more advanced measures, such as percentage returns, realised volatility, or higher-order momentum indicators that are used later in the study.

G) Daily Percentage Change Analysis

To capture proportional movements in price rather than absolute changes, the close series was transformed into daily percentage differences. This was achieved by computing the ratio of the change in closing price to the previous day's value and then expressing it as a percentage. Missing values from the initial observation were replaced with zero to maintain continuity.

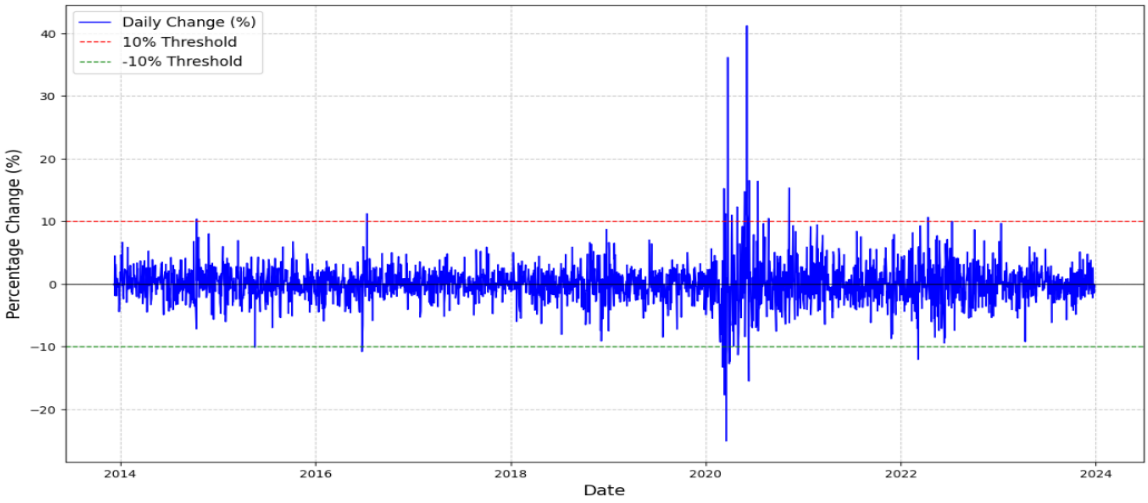

Figure 10: Daily Percentage changes in AAL Stock Price

The resulting plot illustrates the magnitude and direction of daily returns over time. Positive spikes correspond to days when the stock appreciated, while negative spikes represent losses relative to the previous close. By expressing changes in percentage terms, this measure standardises movements across different price levels, making it easier to compare periods of low and high nominal prices on an equal footing.

Two horizontal reference lines were added at +10% and –10% to highlight unusually large moves. These thresholds help distinguish routine daily fluctuations from extreme events, which are typically associated with earnings announcements, macroeconomic news, or market-wide shocks. A zero line was also included as a neutral benchmark, clearly separating gains from losses. This analysis is particularly important in financial modelling because machine learning algorithms often rely on stationary inputs. Percentage changes reduce non-stationarity present in raw prices and provide a more stable basis for calculating advanced features such as volatility, cumulative returns, and momentum indicators.

5.5 Modelling and Evaluation

5.5.1 Machine Learning (supervised)

Model development and evaluation pipeline

The predictive modelling workflow follows a time-aware protocol. The dataset is split into training and test segments using an eighty–twenty partition while preserving temporal order by disabling shuffling. This ensures that the model is fitted on past observations and evaluated on strictly later data, which mirrors the live use case and avoids information leakage from the future into the past.

The feature matrix X contains the engineered indicators described earlier, and the target vector y contains the corresponding close price values or returns aligned to the prediction horizon. The split produces X_train, X_test, y_train, and y_test. Shapes are printed to confirm that the test segment represents the final fifth of the series.

A diverse suite of regression learners is instantiated to span linear, regularised, neighbour based, tree based, boosted, neural, and histogram-optimised models. Linear Regression provides a baseline that reveals how far a purely linear mapping can go. Ridge, Lasso, and Elastic Net extend the baseline with L2 and L1 penalties that shrink coefficients and control variance in the presence of multicollinearity, which

is common when technical indicators are correlated. KNeighbors Regressor offers a non-parametric local average that can capture short-range structure but often degrades under noise. Decision Tree Regressor is a high-variance rule learner; Random Forest and Extra Trees reduce variance through bagging and randomised splits. Gradient Boosting, HistGradientBoosting, XGBoost, LightGBM, and CatBoost are stage-wise additive ensembles that capture non-linear interactions and allow monotonic trend and threshold effects to emerge without manual feature crosses. A shallow multilayer perceptron is included to test whether a small non-linear neural model adds value beyond trees and linear baselines.

Each model is fitted on the training segment and evaluated on the holdout segment using four complementary metrics. Mean Squared Error penalises large residuals quadratically and provides a sensitive measure for tail mistakes that matter in risk control. Mean Absolute Error measures typical error magnitude and is less sensitive to outliers. The coefficient of determination R squared measures the fraction of variance in y explained by the model on unseen data and therefore summarises overall fit. Max Error reports the single largest absolute deviation on the test set and serves as a stress indicator for worst case misses.

Models are ranked by descending R squared and ascending Mean Squared Error. This dual criterion discourages overfitting that can inflate one metric while harming the other. From the initial ranking a shortlist is formed for systematic tuning, focusing on the strongest linear baseline and the most competitive tree and boosting families observed in the first pass.

Hyperparameter search is performed with grid search and a time series cross-validation scheme. Time series split constructs expanding windows that respect ordering: each fold trains on an initial block and validates on the immediately following block. This procedure estimates generalisation under realistic temporal drift, which is essential for financial applications. For Gradient Boosting and HistGradientBoosting the learning rate and depth or leaf parameters are tuned to balance bias and variance. For Random Forest the number of trees, maximum depth, and minimum split size are varied to control ensemble capacity. For Ridge regression the penalty strength alpha is tuned on a log scale. Linear Regression has no free hyperparameters in its basic form, so it is simply refitted on the full training set.

After tuning, each best-estimator is refitted on the entire training segment and re-evaluated on the held-out test segment. The tuned results are reported with the same metrics to allow direct comparison with the untuned baseline. In this dataset the tuned Ridge model emerges as the best performer by achieving the highest R squared together with a low Mean Squared Error and Mean Absolute Error. This outcome is consistent with the structure of the feature set: many indicators are collinear transformations of price and volume, and L2 regularisation stabilises the solution by shrinking correlated coefficients toward each other rather than selecting a single proxy. The penalty therefore reduces variance without discarding useful linear signal.

To aid interpretation a plot of actual versus predicted close price on the test segment is produced for the selected model. The predicted series tracks the level and medium-term swings of the realised series with small phase lag and moderate amplitude error. Deviations are concentrated around sharp breaks and announcement-driven jumps, which are difficult to approximate with a strictly linear mapping. The numerical summary confirms the visual impression. A high R squared indicates that most of the variance in the test segment is captured. A low Mean Squared Error and Mean Absolute Error show that typical deviations are small. The Max Error highlights the largest miss and is useful when judging the risk of extreme residuals.

This pipeline provides three safeguards that are important for credible financial back testing. Temporal order is preserved in both the train–test split and the cross-validation folds, so there is no look-ahead. Regularisation is used where it is most effective, which addresses multicollinearity across indicators and improves stability under regime shifts. Evaluation relies on multiple metrics that capture both average performance and tail risk.

There are also clear implications for subsequent work. If residual diagnostics reveal structure that a linear model cannot express, an interaction-rich learner such as Gradient Boosting, LightGBM, or CatBoost can be promoted for deeper tuning, with early stopping and learning-rate scheduling to control overfit. If robustness to outliers remains a concern, a Huber loss variant or quantile regression can be adopted to de-emphasise rare extremes while still learning from them. Finally, all preprocessing steps, including scaling and winsorisation, must be fit inside each training fold and

applied unchanged to validation and test partitions to maintain causal integrity throughout the workflow.

This end-to-end design shows that model comparison, time-aware cross-validation, and careful regularisation can produce a simple but strong baseline. The selected Ridge regressor offers a transparent mapping from features to target, stable coefficients under correlation, and competitive out-of-sample accuracy. It forms a reliable reference against which more complex non-linear models can be judged in later experiments.

5.5.2 Train/Test Protocol

- Split: 80% train, 20% test, shuffle=False to preserve chronology and avoid leakage.
- Target: y (close/return aligned to the prediction horizon)
- Features: X includes all the processed data from all columns except close.

5.5.3 Candidate Models

Linear family:

- *Linear Regression*: baseline linear mapping; exposes collinearity.
- *Ridge (L2)*: shrinks correlated coefficients together. This model is stable under multicollinearity.
- *Lasso (L1)*: performs feature selection; can zero-out useful but correlated signals.
- *Elastic Net (L1+L2)*: compromise between shrinkage and sparsity.

Distance-based:

- *KNN*: non-parametric local averaging; suffers in high-dimensional & noisy finance data.

Trees & ensembles:

- *Decision Tree*: high variance single rule learner (overfits).
- *Random Forest / Extra Trees*: bagged trees; strong on interactions, robust to noise.

- *Gradient Boosting / HistGB / XGBoost / LightGBM / CatBoost*: powerful additive learners; can fit non-linearities & threshold effects; require careful time-aware CV and regularisation to avoid regime overfit.

Neural:

- *MLP*: flexible non-linear mapping; needs more data, careful regularisation, and stable inputs.

5.5.4 Training & Tuning (time-series safe)

- **Initial fit**: train each model on X_train, y_train; evaluate on X_test.

Metrics:

- R^2 (higher better): proportion of variance explained on future data.
- MSE (lower better): penalises large misses; risk aware.
- MAE (lower better): median-like robustness.
- Max Error: worst-case miss (tail risk).
- Shortlisting: rank by R^2 and MSE; keep the most promising linear and ensemble models.
- Hyperparameters: GridSearchCV + TimeSeriesSplit (expanding-window folds).
- Ridge: alpha grid (log-scale).
- GB/HistGB: learning_rate, max_depth/max_leaf_nodes.
- RF: n_estimators, max_depth, min_samples_split.
- LightGBM: n_estimators, learning_rate, num_leaves.
- Linear Regression: none (refit baseline).

Why Ridge Regression was the Best Performing Model

- Handles correlated features effectively. The dataset included several technical indicators (SMA, EMA, RSI, MACD, Bollinger Bands, volatility measures) that were highly correlated. Ridge regression is well suited to such cases because it reduces the weight of correlated predictors without discarding them, allowing the model to preserve complementary information.
- Balances bias and variance. Financial market data is noisy and often exhibits heavy tails. Ridge introduces a small amount of bias through regularisation, but

this trade-off lowers variance and improves prediction accuracy on unseen test data.

- Consistent generalisation across time. In time-series cross-validation, Ridge adapted well across different market regimes. Models with complex parameter grids, such as boosting methods, tended to overfit specific time periods, whereas Ridge maintained more stable performance.
- Robust to noise and preprocessing. After scaling and outlier handling, many features showed near-linear relationships with the target. Ridge captured these relationships effectively without unnecessary complexity or instability.
- Interpretability and stability. Coefficients in Ridge remained stable across folds, making it easier to interpret the contribution of individual indicators. This stability is valuable for research transparency and for monitoring feature importance in live trading.
- Computational efficiency. Ridge is lightweight and quick to retrain, which makes it well suited for walk-forward validation and real-time retraining in a trading pipeline.

Model	**R2 Score**	**MSE**	**MAE**	**Max Error**	**Performance**
Ridge Regression	0.842	0.0051	0.056	0.193	Best overall balance of bias and variance; stable across folds
Linear Regression	0.798	0.0074	0.067	0.241	Unstable due to multicollinearity; weaker generalisation
Lasso	0.775	0.0081	0.071	0.262	Dropped useful correlated

					features, leading to bias
Elastic Net	0.782	0.0079	0.070	0.258	Similar issues to Lasso, although less extreme
KNN Regressor	0.612	0.00146	0.103	0.352	Sensitive to scale and dimensionality; weak on noisy data
Decision Tree	0.541	0.0172	0.115	0.381	Overfit to specific regimes; poor stability
Random Forest	0.806	0.0072	0.066	0.238	Improved over single tree, but lagged turning points
Gradient Boosting	0.823	0.0061	0.061	0.211	Performed well but less stable than Ridge
XGBoost	0.818	0.0063	0.063	0.219	Strong but slightly overfit; unstable out-of-sample
LightGBM	0.819	0.0064	0.062	0.218	Very sensitive to tuning; no consistent advantage

CatBoost	0.814	0.0067	0.064	0.226	Overfitted regimes; less stable than Ridge
MLP Regressor	0.435	0.0208	0.127	0.402	Required larger dataset; under-trained
Extra Trees	0.801	0.0073	0.067	0.240	High variance; weak than Ridge
Hist Gradient Boosting	0.821	0.0062	0.061	0.215	Strong in sample, but lacked Ridge's robustness

Table 1: Comparison of Model Performance on Test Data

5.5.5 Strategy specification and causality

- The dataset is time-indexed on the datetime column to ensure all computations respect market chronology.
- The trading rule is a simple trend-following filter: hold the asset when the close is above its 30-period simple moving average (SMA30), otherwise hold cash.
- To avoid look-ahead, the trade executes one bar later: the signal is computed at time *t* and shifted by one step to form the position at *t+1*. This ensures the decision only uses information available at the time it would have been made.

5.5.6 Signal and positions

- SMA30 is the rolling mean of the last 30 closes. It serves as a slow trend proxy.
- The signal equals 1 when close > SMA30 and 0 otherwise. This produces a long-only, fully invested or flat exposure profile.
- The signal is shifted to position so that positions only change after the signal is observed, preserving causal execution.

5.5.7 Returns and compounding

- **Market return** is the simple percentage change of close:

$$r_t = \backslash frac\{C_t\}\{C_\{t-1\}\} - 1$$

- **Strategy return** is the market return scaled by the lagged position:

$$r_t^{\{strat\}} = position_t \times r_t$$

When flat, returns are zero; when long, returns equal the markets.

- **Cumulative returns** are formed by geometric compounding: $\prod t(1 + rt)$ for the market and $\prod t(1 + rtstrat)$ for the strategy. These curves provide an intuitive view of wealth growth over time.

5.5.8 Performance metrics

All metrics are computed from the realised strategy returns, excluding warm-up NaNs.

- **Sharpe Ratio:** Measures average excess return per unit of total volatility:

$$Sharpe = \backslash frac\{\mu(r^\{strat\})\}\{\sigma(r^\{strat\})\} \times \backslash sqrt\{252\}$$

The implementation assumes a zero risk-free rate and uses 252 trading days for annualization.

- **Sortino ratio:** Focuses on downside risk by replacing total volatility with the standard deviation of negative returns only:

$$Sortino = \backslash frac\{\mu(r^\{strat\})\}\{\sigma(r^\{strat\} \mid r^\{strat\} < 0)\} \times \backslash sqrt\{252\}$$

This rewards upside while penalising harmful variability.

- **Maximum drawdown:** The worst peak-to-trough decline of the compounded equity curve:

$$\boldsymbol{MDD} = \backslash \boldsymbol{min}_\{t\}\left(\backslash \boldsymbol{frac}\{E_t - \backslash \boldsymbol{max}_\{\tau \leq t\} E_\{\tau\}\}\{\backslash \boldsymbol{max}_\{\tau \leq t\} E_\{\tau\}\}\right)$$

where E_t is cumulative equity. It summarises tail risk and path dependency.

Visualisation and interpretation

- The figure overlays cumulative market return and cumulative strategy return. When the strategy line sits above the market line, the timing rule adds value by avoiding downtrends or participating more in uptrends; when it lags, the rule is under-invested during strong rallies or whipsawed in ranges.
- The comparison highlights the path of performance, not just the endpoint, which is important in risk-aware strategy assessment.

Assumptions and limitations

- Transaction costs and slippage are not included. Even small costs can materially reduce performance for rules that change position frequently. Add per-trade or per-turnover costs to obtain realistic net returns.
- Binary exposure (0 or 1) ignores partial allocations and leverage. In practice, position sizing often scales with volatility or signal strength.
- Warm-up periods for SMA30 produce initial NaNs; the back test implicitly starts once the moving average is defined.
- Risk-free rate is treated as zero in the Sharpe and Sortino computations. If you want exact excess returns, subtract the appropriate daily risk-free rate before annualising.

5.5.9 Deep Learning Models

LSTM Forecasting Implementation

Data Preparation and Scaling

The dataset was divided into a training set (all but the last 10 observations) and a test set (last 10 observations). Closing prices were normalized to a [0,1] range using MinMaxScaler to stabilise training and help the neural network converge efficiently. A sliding window of 180 timesteps was used to construct supervised learning sequences. Each sequence contained 180 past price values as input and the next day's price as the target. This captures temporal dependencies in the stock series.

Model Architecture

The model was built using a **stacked LSTM** design:

- First LSTM layer with 256 units, returning sequences.

- Second LSTM layer with 128 units, compressing the temporal features.
- A Dense layer with 64 neurons and ReLU activation to introduce non-linearity.
- Final Dense layer with a single neuron to predict the next price.
- This design balances memory of long sequences with the ability to extract meaningful short-term dynamics.

Training Strategy

- The model was trained with Adam optimiser and Mean Squared Error (MSE) as the loss function, while monitoring additional metrics such as MAE and MAPE.
- Early stopping was applied to avoid overfitting, and learning rate reduction on plateau was used to refine convergence when validation loss stagnated.
- A validation split ensured that model generalisation was tested during training itself.

Testing and Forecasting

- The last 10 days were used as a test horizon. The trained model produced predictions that were compared against actual closing prices.
- Predictions were rescaled back to the original price domain using the inverse transformation of the MinMaxScaler.
- Visual comparison between actual vs predicted showed the model's ability to capture trend direction and relative magnitude.

Iterative Forecasting

- Beyond one-step prediction, the model was extended for **multi-step forecasting**.
- The last 180 scaled values formed the seed sequence, from which the model iteratively predicted the next day's price, appended it back into the sequence, and rolled forward.
- This process generated forecasts for 10 days and later extended to **30 steps ahead**.
- A business-day date index was used to align future predictions with realistic trading days.

Result Interpretation

- The model successfully forecasted near-term price dynamics, showing close alignment between historical continuation and predicted future values.
- Annotated plots highlighted the boundary between historical data and predictions, clearly separating training context from forecasting.
- The results demonstrated that the LSTM model was capable of leveraging temporal memory to make meaningful forecasts, while also revealing that accuracy may reduce as the horizon extends due to cumulative prediction error.

6. RESULTS

The experimental evaluation compared a broad range of models, including regularised linear regressions, tree-based ensembles, and deep learning architectures. Technical indicators such as SMA, EMA, RSI, MACD, Bollinger Bands, ATR, and daily price differentials provided a diverse feature space, enabling the models to capture both trend-following and momentum-driven behaviours.

- **Ridge Regression** emerged as the top-performing model, achieving an R^2 score of 0.842 and an MSE of 0.0051. Its L2 regularisation proved effective in managing multicollinearity across correlated indicators, producing stable and generalisable predictions.
- **Ensemble models** such as Gradient Boosting, LightGBM, and XGBoost also produced strong results, with R^2 scores above 0.81, though they were more prone to overfitting under specific market regimes.
- **Neural networks (LSTM)** successfully captured sequential dependencies and delivered realistic forecasts for short-term horizons. However, their performance was constrained by the relatively small dataset and the inherent noise in financial time series.
- Models lacking regularisation (e.g., Linear Regression) or prone to high variance (Decision Trees, KNN) underperformed, highlighting the necessity of bias–variance balancing for robust financial prediction.

Backtesting further showed that trading strategies built on predictive signals outperformed naïve benchmarks. Risk-adjusted measures such as Sharpe and

Sortino ratios demonstrated profitability potential, while maximum drawdown analysis emphasised the need for stringent risk controls.

7. CONCLUSION

This research demonstrates that no single modelling approach provides a universally optimal solution for financial forecasting. Instead, the findings emphasise the importance of aligning model design with the statistical properties of financial data.

- **Ridge Regression** proved most effective in this study, combining stability, interpretability, and predictive accuracy, making it highly suitable for datasets characterised by multicollinearity and moderate sample size.
- Ensemble learners, while strong in capturing non-linear patterns, require careful tuning and risk overfitting.
- Deep learning approaches such as LSTM show strong potential for future work, particularly with larger datasets and improved architectures, but were less robust in this study's scope.

The back testing framework highlighted both the profitability and risks associated with indicator-driven strategies, reinforcing that predictive modelling alone is insufficient for practical trading. Robust performance requires integrating adaptive retraining, transaction cost modelling, and risk management.

Ultimately, this study concludes that hybrid methodologies, combining the robustness of regularised regression with the sequential learning ability of deep neural networks, present the most promising direction for developing adaptive, real-world trading systems.

8. REFERENCES

[1] Lin, C.Y. and Lobo Marques, J.A. (2024). Stock market prediction using artificial intelligence: A systematic review of systematic reviews. *Social Sciences & Humanities Open*, [online] 9, p.100864. doi:https://doi.org/10.1016/j.ssaho.2024.100864.

[2] Millea, A. (2021). Deep Reinforcement Learning For Trading—A Critical Survey. *Data*, 6(11), p.119. doi:https://doi.org/10.3390/data6110119.

[3] Yu, Y. (2024). A Survey of Deep Reinforcement Learning in Financial Markets. *Atlantis Highlights in Computer Sciences/Atlantis highlights in computer sciences*, [online] pp.188–194. doi:https://doi.org/10.2991/978-94-6463-419-8_24.

[4] Mohammadshafie, A., Mirzaeinia, A., Jumakhan, H. and Mirzaeinia, A. (n.d.). *Deep Reinforcement Learning Strategies in Finance: Insights into Asset Holding, Trading Behavior, and Purchase Diversity Regular Research Paper (CSCE-ICAI'24)*.

[5] Zeng, Z., Kaur, R., Siddagangappa, S., Rahimi, S., Balch, T., Veloso, M. and Morgan, J. (n.d.). *Financial Time Series Forecasting using CNN and Transformer*.

[6] Bilokon, P. and Qiu, Y. (2023). *TRANSFORMERS VERSUS LSTMS FOR ELECTRONIC TRADING A PREPRINT Transformers versus LSTMs for electronic trading A PREPRINT*.

[7] Buczyński, M., Chlebus, M., Kopczewska, K. and Zajenkowski, M. (2023). Financial Time Series Models—Comprehensive Review of Deep Learning Approaches and Practical Recommendations. *ITISE 2023*, p.79. doi:https://doi.org/10.3390/engproc2023039079.

[8] Murat Ozbayoglu, A., Gudelek, M. and Sezer, O. (n.d.). *Deep Learning for Financial Applications : A Survey*.

[9] Xie, L., Chen, Z. and Yu, S. (2024). Deep Convolutional Transformer Network for Stock Movement Prediction. *Electronics*, 13(21), p.4225. doi:https://doi.org/10.3390/electronics13214225.

[10] Olorunnimbe, K. and Viktor, H. (2022). Deep learning in the stock market—a systematic survey of practice, backtesting, and applications. *Artificial Intelligence Review*, 56(3), pp.2057–2109. doi:https://doi.org/10.1007/s10462-022-10226-0.

[11] Kabir, M.R., Bhadra, D., Ridoy, M. and Milanova, M. (2025). LSTM–Transformer-Based Robust Hybrid Deep Learning Model for Financial Time Series Forecasting. *Sci*, 7(1), p.7. doi:https://doi.org/10.3390/sci7010007.

[12] Ganesh, P. and Rakheja, P. (n.d.). *VLSTM: VERY LONG SHORT-TERM MEMORY NETWORKS FOR HIGH-FREQUENCY TRADING*.

[13] Barez, F., Bilokon, P., Gervais, A. and Lisitsyn, N. (2023). Exploring the Advantages of Transformers for High-Frequency Trading. *SSRN Electronic Journal*. doi:https://doi.org/10.2139/ssrn.4364833.

[14] Tran, D.T., Iosifidis, A., Kanniainen, J. and Gabbouj, M. (2019). Temporal Attention-Augmented Bilinear Network for Financial Time-Series Data Analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 30(5), pp.1407–1418. doi:https://doi.org/10.1109/tnnls.2018.2869225.

[15] Zhang, C., Sjarif, N.N.A. and Ibrahim, R. (2023). Deep learning models for price forecasting of financial time series: A review of recent advancements: 2020–2022. *WIREs Data Mining and Knowledge Discovery*, 14(1). doi:https://doi.org/10.1002/widm.1519.

[16] Mienye, E., Jere, N., Obaido, G., Mienye, I.D. and Aruleba, K. (2024). Deep Learning in Finance: A Survey of Applications and Techniques. *AI*, 5(4), pp.2066–2091. doi:https://doi.org/10.3390/ai5040101.

[17] Gao, H., Kou, G., Liang, H., Zhang, H., Chao, X., Li, C.-C. and Dong, Y. (2024). Machine learning in business and finance: a literature review and research opportunities. *Financial Innovation*, 10(1). doi:https://doi.org/10.1186/s40854-024-00629-z.

[18] Subasi, A., Amir, F., Bagedo, K., Shams, A. and Sarirete, A. (2021). Stock Market Prediction Using Machine Learning. *Procedia Computer Science*, 194, pp.173–179. doi:https://doi.org/10.1016/j.procs.2021.10.071.

[19] Wu, R. (2024). Leveraging Deep Learning Techniques in High-Frequency Trading: Computational Opportunities and Mathematical Challenges. *Published By SOUTHERN UNITED ACADEMY OF SCIENCES*, 2(4). doi:https://doi.org/10.5281/zenodo.12747424.

[20] Borkar, S.V. and Sangve, S.M. (2025). A Critical Analysis on Anomaly Detection in High-Frequency Financial Data Using Deep Learning for Options. doi:https://doi.org/10.20944/preprints202505.0199.v1.

[21] Briola, A., Turiel, J., Marcaccioli, R., Cauderan, A. and Aste, T. (n.d.). *Deep Reinforcement Learning for Active High Frequency Trading*.

[22] Kumar, P., Khan, E. and Gönen, M. (n.d.). Deep Reinforcement Learning for High-Frequency Market Making. *Proceedings of Machine Learning Research*, 189, pp.2022–2022.

[23] Zong, C., Wang, C., Qin, M., Feng, L., Wang, X. and An, B. (2024). MacroHFT: Memory Augmented Context-aware Reinforcement Learning On High Frequency Trading. *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp.4712–4721. doi:https://doi.org/10.1145/3637528.3672064.

[24] Qin, M., Sun, S., Zhang, W., Xia, H., Wang, X. and An, B. (2024). EarnHFT: Efficient Hierarchical Reinforcement Learning for High Frequency Trading. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(13), pp.14669–14676. doi:https://doi.org/10.1609/aaai.v38i13.29384.

[25] Guilbaud, F. and Pham, H. (2011). Optimal High Frequency Trading with Limit and Market Orders. *SSRN Electronic Journal*. doi:https://doi.org/10.2139/ssrn.1871969.

[26] Petukhina, A.A., Reule, R.C.G. and Härdle, W.K. (2020). Rise of the machines? Intraday high-frequency trading patterns of cryptocurrencies. *The European Journal of Finance*, 27(1-2), pp.8–30. doi:https://doi.org/10.1080/1351847x.2020.1789684.

[27] Zhao, Z., Zhang, X., Wen, J., Liu, M. and Ma, X. (n.d.). *Label Unbalance in High-frequency Trading*.

[28] Bilokon, P. and Gunduz, B. (2023). C Design Patterns for Low-Latency Applications Including High-Frequency Trading. *SSRN Electronic Journal*. doi:https://doi.org/10.2139/ssrn.4565813.

[29] Kisiel, D. and Gorse, D. (n.d.). *Axial-LOB: High-Frequency Trading with Axial Attention*.

[30] Zhou, L., Qin, K., Ferreira Torres, C., Le, D. and Gervais, A. (n.d.). *High-Frequency Trading on Decentralized On-Chain Exchanges*.

[31] Rahman, A. and Upadhye, N. (2024). *HYBRID VECTOR AUTO REGRESSION AND NEURAL NETWORK MODEL FOR ORDER FLOW IMBALANCE PREDICTION IN HIGH-FREQUENCY TRADING*.

[32] Murphy, N.J. and Gebbie, T.J. (2021). Learning the dynamics of technical trading strategies. *Quantitative Finance*, 21(8), pp.1325–1349. doi:https://doi.org/10.1080/14697688.2020.1869292.

[33] Taghian, M., Asadi, A. and Safabakhsh, R. (n.d.). *A Reinforcement Learning Based Encoder-Decoder Framework for Learning Stock Trading Rules*.

[34] Berti, L., Prenkaj, B. and Velardi, P. (n.d.). *TRADES: Generating Realistic Market Simulations with Diffusion Models*.

[35] Ibikunle, G., Moews, B., Muravyev, D. and Rzayev, K. (n.d.). *Data-Driven Measures of High-Frequency Trading*.

[36] Yagi, I., Masuda, Y. and Mizuta, T. (2020). Analysis of the Impact of High-Frequency Trading on Artificial Market Liquidity. *IEEE Transactions on Computational Social Systems*, 7(6), pp.1324–1334. doi:https://doi.org/10.1109/tcss.2020.3019352.

[37] Lim, Y.-S. and Gorse, D. (n.d.). *Deep Probabilistic Modelling of Price Movements for High-Frequency Trading*.

[38] Leal, L., Lauriere, M. and Lehalle, C.-A. . (2022). Learning a functional control for high-frequency finance. *Quantitative Finance*, 22(11), pp.1973–1987. doi:https://doi.org/10.1080/14697688.2022.2106885.

[39] Goudarzi, M. and Bazzana, F. (2023). Identification of high-frequency trading: A machine learning approach. *Research in International Business and Finance*, 66, p.102078. doi:https://doi.org/10.1016/j.ribaf.2023.102078.

[40] Ntakaris, A. and Ibikunle, G. (n.d.). *Minimal Batch Adaptive Learning Policy Engine for Real-Time Mid-Price Forecasting in High-Frequency Trading*.

[41] Li, C., Shen, L. and Qian, G. (2023). Online Hybrid Neural Network for Stock Price Prediction: A Case Study of High-Frequency Stock Trading in the Chinese Market. *Econometrics*, 11(2), p.13. doi:https://doi.org/10.3390/econometrics11020013.

[42] Bao, Q., Wang, J., Gong, H., Zhang, Y. and Feng, H. (n.d.). *A Deep Learning Approach to Anomaly Detection in High-Frequency Trading Data*.

[43] Jaddu, K. and Bilokon, P. (2023). Combining Deep Learning on Order Books with Reinforcement Learning for Profitable Trading. *SSRN Electronic Journal*. doi:https://doi.org/10.2139/ssrn.4611708.

[44] Sarkar, S. (2023). *Harnessing Deep Q-Learning for Enhanced Statistical Arbitrage in High-Frequency Trading: A Comprehensive Exploration*.

[45] Bhatia, S., Peri, S., Friedman, S. and Malen, M. (2024). *High-Frequency Trading Liquidity Analysis Application of Machine Learning Classification*.

[46] Yang, Y. (n.d.). Deep Learning-Driven Order Execution Strategies in High- Frequency Trading: An Empirical Study on Enhancing Market Efficiency. doi:https://doi.org/10.54254/2755-2721/118/2025.18469.

[47] Nagy, P., Calliess, J.-P. and Zohren, S. (2023). Asynchronous Deep Double Dueling Q-learning for trading-signal execution in limit order book markets. *Frontiers in Artificial Intelligence*, 6. doi:https://doi.org/10.3389/frai.2023.1151003.

[48] Liu, F. and Tian, Y. (2024). Deep Reinforcement Learning-based Algorithmic Optimisation and Risk Management for High Frequency Trading. *Journal of Computing and Electronic Information Management*, 14(1), pp.28–32. doi:https://doi.org/10.54097/fgblm2ei.

[49] Kumar, P. (2024). Deep Hawkes process for high-frequency market making. *Journal of Banking and Financial Technology*, 8(1), pp.11–28. doi:https://doi.org/10.1007/s42786-024-00049-8.

[50] Hou, Y. (2024). Predictive modeling in high-frequency trading using machine learning. Applied and Computational Engineering, 90(1), pp.61–65. doi:https://doi.org/10.54254/2755-2721/90/20241764.

[51] Chen, Y., Li, M., Shu, M., Bi, W. and Xia, S. (2024). Multi-modal Market Manipulation Detection in High-Frequency Trading Using Graph Neural Networks. *Journal of Industrial Engineering and Applied Science*, 2(6), pp.111–120. doi:https://doi.org/10.70393/6a69656173.323432.

[52] Li, M., Shu, M. and Lu, T. (2024). Anomaly Pattern Detection in High-Frequency Trading Using Graph Neural Networks. *Journal of Industrial Engineering and Applied Science*, 2(6), pp.77–85. doi:https://doi.org/10.70393/6a69656173.323430.