CS815 Assignment 2

Andrew Cheong & Panagiotis Daviotis

2025-04-11

Background & Literature Review

This analysis processes stock market data to create a comprehensive dataset with technical indicators, lagged features, and a target variable for predictive modelling.

Predicting Next-Day Stock Prices Using ML and AI: Research on machine learning (ML) and artificial intelligence (AI) for stock price prediction has advanced significantly. Early methods relied on regression models and support vector machines (SVMs) (Dronyuk et al., 2025; Nabipour et al., 2020). However, deep learning approaches, particularly Long Short-Term Memory (LSTM) networks, have since proven highly effective in capturing stock price patterns (Hijazi et al., 2025; Li, 2022).

Feature Selection and Recursive Feature Elimination (RFE): Selecting relevant features is crucial for improving prediction accuracy. Techniques like principal component analysis (PCA) and autoencoders have been explored, but Recursive Feature Elimination (RFE) has shown strong results, especially when combined with neural networks and ensemble models (Htun et al., 2023; Naik & Mohan, 2019).

Neural Networks in Stock Prediction: Advanced recurrent architectures, such as LSTMs and gated recurrent units (GRUs), effectively model both short- and long-term market trends. Ensemble learning, which combines deep learning with traditional techniques, further enhances predictive power (Padalkar et al., 2024; Singh & Malhotra, 2024).

Investment Strategies and Genetic Algorithms (GAs): Beyond prediction, AI-driven approaches guide investment strategy development. Genetic algorithms optimize trading strategies by iteratively adjusting to maximize returns. When combined with ML models, GAs enable adaptive decision-making in dynamic markets, often outperforming traditional methods (Dworkis & Huang, 2008; Kuepper, 2025).

Conclusion: This research, based on the insights from the literature, will attempt to use RFE, Neural Networks, and GAs to enhance stock price prediction and trading strategy optimization. RFE refines input data, NNs capture complex patterns, and GAs optimize trading strategies, aiming to improve accuracy and market adaptability.

Data Import

Google (GOOG) was selected due to its high liquidity and stable trends, making it ideal for algorithmic trading. Stock price data was taken starting from 2015, as a decade-long dataset captures different market cycles, including bull and bear markets. Also, data from the COVID-19 pandemic ensures that the model learns to handle extreme volatility. Columns "Date", "Open", "High", "Low", "Close", "Volume", "Adjusted" were selected as they provide essential information about the stock's daily trading activity. 'Date' acts as the index for time-series analysis, 'Open/High/Low/Close' capture the price movements within each trading day, essential for trend and volatility analysis, 'Volume' indicates market participation and liquidity and 'Adjusted' reflects price adjustments for corporate actions like splits or dividends.

Feature Engineering

Indicators are selected to capture various aspects of market behavior: **1. The RSI (14)** measures momentum and identifies overbought/oversold conditions. **2. MACD** tracks trend strength using short-term and long-term moving averages. **3. Bollinger Bands Width (BBands_Width)** quantifies volatility by measuring the width of Bollinger Bands (which uses moving averages and standard deviations to measure price volatility). **4. ATR (Average True Range)** represents market volatility over time. **5. Volume MA (5)** smooths out volume data to identify trends in trading activity. **6. Volume Ratio** compares current volume to its moving average, highlighting unusual activity. **7. Close MA (50)** represents a long-term trend indicator based on closing prices. **8. Close MA Ratio** shows how the current price compares to its long-term average. **9. The Daily Return** captures daily price changes as percentages, useful for return-based strategies. **10. Volatility (10)** reflects short-term risk.

A lookback window of 10 days captures short-term trends while maintaining computational efficiency without introducing excessive lag or overfitting.

Exploratory Data Analysis

The combined chart offers a rich, multidimensional view of Google's stock.

1. Price Trends & Volatility: There is a clear upward trend with volatility spikes during market stress highlighted by rolling volatility.

2. Trend Confirmation & Smoothing: The overlay of a 50-day moving average provides a clearer depiction of the underlying trend.

3. Return Distribution & Risk Indicators: Daily return histogram is centered around zero with fat tails, indicating rare but extreme price swings.

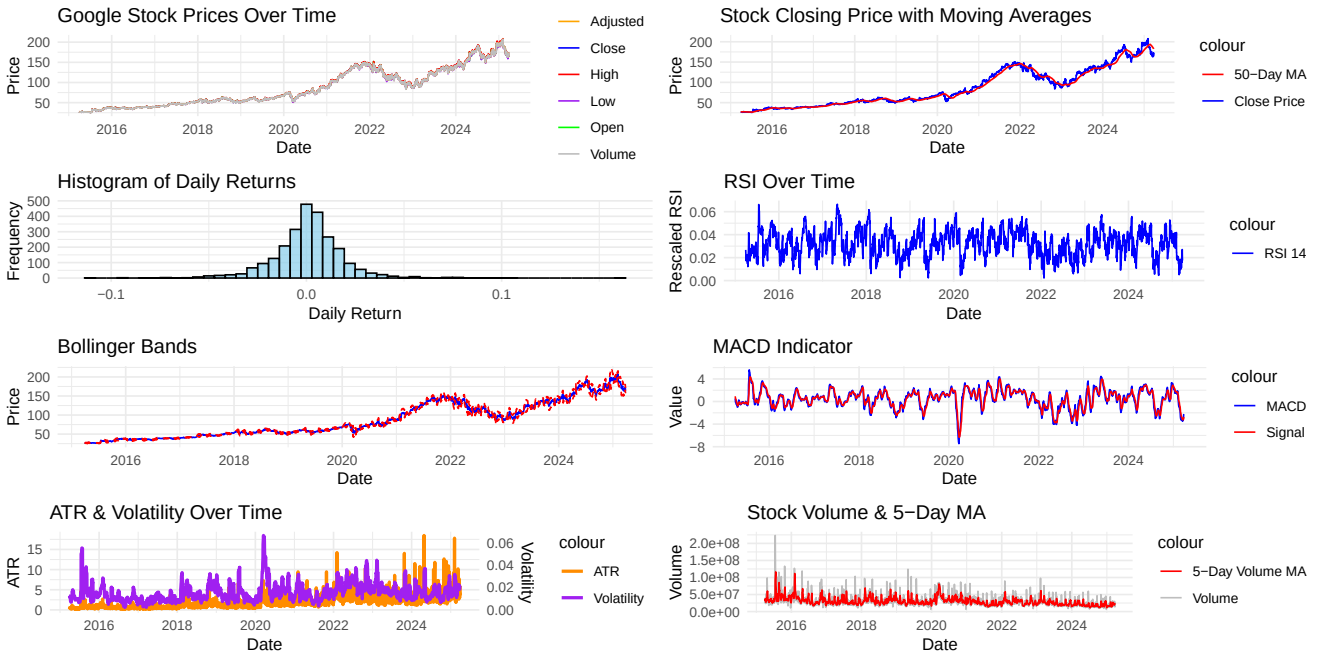
4. RSI: Periods when RSI values approach extreme thresholds (typically above 70 or below 30) implying moments when the market may be overextended and prone to reversal.

5. Volatility Bands & Trading Signals: When stock price repeatedly the bands, it suggests potential overbought or oversold conditions.

6. Trend Reversal Indications: The MACD indicator, emphasizes crossover points interpreted as bullish or bearish signals.

7. Volatility Intensity – ATR: Spikes in ATR corroborate the other volatility indicators and show periods of possible market stress.

8. Volume: Finally, the volume chart—enhanced by a 5-day moving average of volume—provides context on trading activity.



Correlation with Target Feature Plot

The chart provides a clear visual representation of how various features relate to the closing price.

1. Strong Positive Correlations: Close_MA_50, Adjusted, Open, High, Low (approx. 1 correlation – capture very similar information) confirm expected relationships since they are closely intertwined. The strong correlation of Close_MA_50 with the closing price indicates effective reflection of the underlying trend.

2. Moderate Positive Correlations: BBands_Width, and ATR have (moderate positive correlation with the closing price). This suggests that periods of larger price movements are associated with changes in closing values.

3. Weak Positive Correlations: RSI_14 (a momentum indicator), Volatility_10, Close_MA_Ratio, MACD, and Daily_Return (weak correlation) capture short-term momentum.

4. Weak Negative Correlations: Volume_Ratio, Volume, and Volume_MA_5 show an inverse correlation. The negative association of volume-related metrics may suggest that higher volumes are not necessarily aligned with higher closing prices.

Outlier Detection Plots

The chart consists of six side-by-side box plots highlighting the presence of outliers.

Close: shows that while the bulk of closing prices follow a consistent range, there are occasional higher values that could signal unique events or market spikes.

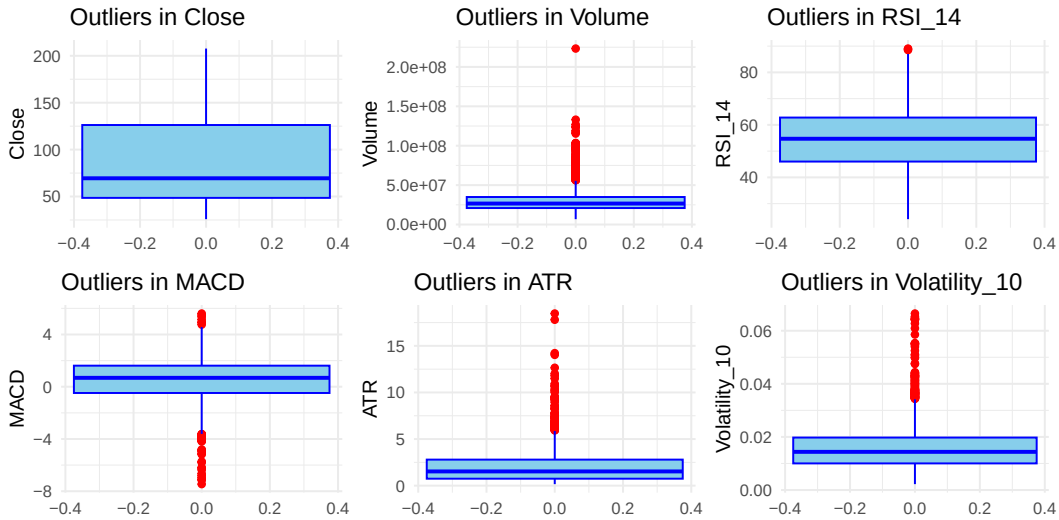
Volume: may indicate periods of intense trading activity with some outliers, potentially due to market news or significant investor actions.

RSI_14: indicates that most RSI values fall between 40 and 60, which shows a range-balanced market.

MACD: shows few outliers lie below -4, suggesting periods when bearish momentum was particularly strong.

ATR: reveals that most values remain below 5, yet several outliers reflecting periods when price swings were more pronounced.

Volatility_10: shows most values below 0.02 and some outliers up to 0.06, underscoring short-term periods of heightened volatility.



Data Splitting & Normalisation

The selected normalisation method was robust scaling (median & IQR) as it is less sensitive to outliers compared to standard scaling or min-max normalization ensuring that extreme values do not disproportionately influence the model while preserving relative feature distributions. Target variable was also normalized because neural networks (the method which will be used to make predictions of the next-day's stock price) benefit from scaled target variables since it prevents large gradients and

speeds up convergence. This is particularly relevant since financial data often has large variations in prices, so normalizing both features and targets ensures consistent scaling. After prediction, denormalisation is applied to ensure interpretability.

Feature Selection

Recursive Feature Elimination (RFE) was used for feature selection as it systematically removes less relevant features, reducing dimensionality while retaining predictive power. The model used in RFE was linear regression since it yielded the best results, indicating strong linear dependencies among features particularly relevant for financial time series data. Following this, highly corellated features were removed to reduce multicollinearity, and winsorisation for outlier clipping was applied to limit extreme values.

Neural Network

Neural Networks (and multi-layer perceptrons) are capable of forecasting directions of future price movements in financial time series. The two layers with sizes (10, 5) provide flexibility to capture complex patterns in the data while avoiding overfitting. The optimization algorithm Resilient Backpropagation (rprop+) is efficient for training neural networks with continuous outputs without manual learning rate adjustments. A convergence threshold (0.01) and stepmax (1e6) ensure thorough training without premature termination. This setup enables effective feature learning while preventing overfitting.

```
set.seed(123)
nn_model <- neuralnet(
  as.formula(paste("Target ~", paste(feature_names, collapse = " + "))), # features for target prediction
  data = train_data_norm,
  hidden = c(10, 5), # More flexible architecture
  linear.output = TRUE, # Continuous suitable for Regression Models
  threshold = 0.01, # Convergence
  stepmax = 1e6,  # Ensuring Sufficient convergence
  algorithm = "rprop+",# Resilient backpropagation
  lifesign = "full", # enable output
)
```

```
## hidden: 10, 5    thresh: 0.01    rep: 1/1    steps:    1000  min thresh: 0.0329287669312122
##                                                        2000  min thresh: 0.0190160222070112
##                                                        3000  min thresh: 0.0118419742435594
##                                                        3219  error: 0.80349  time: 6.51 secs
```

Model Evaluation

```
##   Metric      Train      Test
## 1   MAPE  0.7816265  3.497129
## 2    MDA 75.8826455 79.083665
```

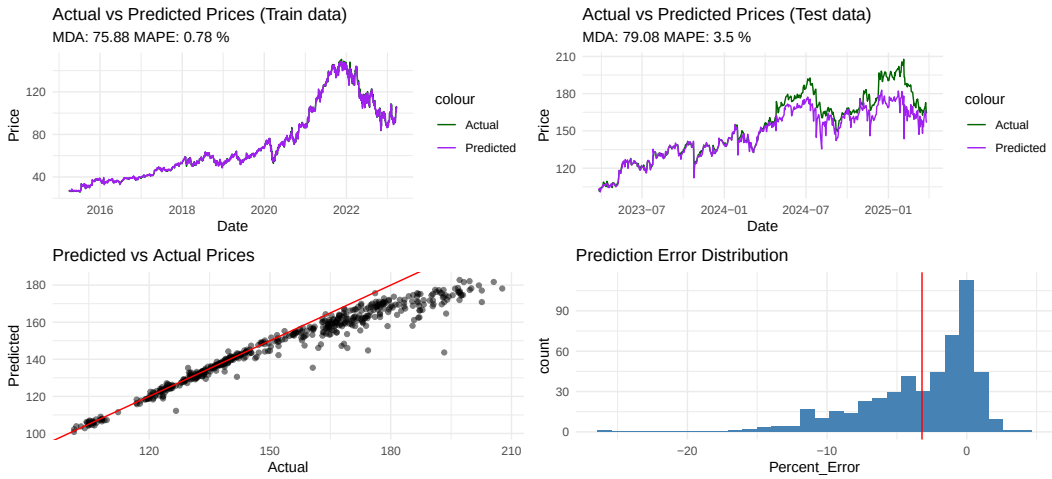
Evaluation Metrics - Why MAPE, MDA over RMSE/MSE? MAPE is preferred because it expresses errors as a percentage, making it scale-independent, which is crucial in financial forecasting and ensures consistent evaluation across assets with varying price ranges. Moreover, MDA is crucial because it complements MAPE since it assesses model's ability to predict price direction. RMSE / MSE is not used because they overemphasize large errors, which was the case on our analysis since large errors produced (April 2024) particularly due to the increase of stock price to historically-high levels not existing in trading data, so such metrics would provide a decisive evaluation of model.

Training vs. Test Set Comparison

MAPE (Relative Error): Train MAPE = 0.78% → Very low error - the model fits the training data well. Test MAPE = 3.49% → also relatively low but indicates a slight difficulty in generalization attributed to the inherent complexity of financial data, particularly to the unprecedented surge (April 2024) in stock prices, which has not been observed since 2015. While a larger dataset could enhance training and improve predictive accuracy, we have opted to maintain computational efficiency by limiting dataset expansion.

MDA (Directional Accuracy): Train MDA = 75.82% → the model correctly predicts the price movement direction 75.82% of time, also in test MDA (79.08%) there is a slight improvement in prediction accuracy showing that the model maintains its ability to predict price direction consistently, which is critical for trading strategies.

Prediction diagnostics



Actual vs Predicted Train: The low MAPE and high MDA suggest that the model is well-suited for trend forecasting, making it reliable for directional trading strategies. Some deviations between actual and predicted prices imply that short-term accuracy could be enhanced with additional feature tuning.

Actual vs Predicted Test: 1. Strong Trend Forecasting: The model effectively tracks overall price movements, making it useful for directional trading strategies. 2. Some Deviation in Absolute Accuracy: While the lines generally follow each other, occasional gaps suggest the model could be slightly refined. 3. Consistently Low Prediction Error: The low MAPE and high MDA suggest that the model is well-suited for trend-based decision-making, even if individual price points deviate slightly.

Predicted vs Actual Prices: The model effectively captures price trends, demonstrating solid predictive accuracy. There is some bias toward lower predictions compared to actual prices, which could be improved with slight adjustments.

Interpretation of Error Distribution: The concentration around 0% error suggests the model is well-calibrated and provides mostly accurate forecasts. The skewness to the left may indicate occasional under-predictions, but they are relatively rare. This distribution is favorable for financial forecasting, as predictions tend to stay close to actual values without excessive error spikes.

Trading Strategy

The trading strategy is a rule-based function that triggers buy/sell actions only when the predicted price, obtained from a neural network, crosses certain dynamic thresholds. Essentially, it checks if market conditions are favorable before executing a trade. The strategy operates in full positions (all-in or all-out): it buys as many shares as possible when a buy signal is triggered (after accounting for transaction fees), or sells all currently held shares when a sell signal occurs. This might seem risky and inflexible, but the use of adaptive actions (in the trading strategy) based on the measure of volatility mitigate the risk and maintain flexibility.

There are 3 dynamic thresholds: buy, sell, and stop-loss, all of which are proportional to recent volatility in the stock. When volatility is high, these thresholds become more conservative (i.e. they require stronger signals to trigger a trade). This mechanism helps mitigate the risk of trading during periods of uncertainty, thereby ensuring adaptive decision-making in varying market conditions. These thresholds are scaled by multiplier parameters (coefficients) of the volatility, which are evolved through a Genetic Algorithm on the training data. This allows the trading algorithm to find optimal multiplier values tailored to Google's historical stock behaviour, effectively fine-tuning the sensitivity of the thresholds to maximize the strategy's performance based on a selected fitness function (e.g. Sharpe ratio).

Finally, we scaled the predicted ratio by a factor of 1.1 when checking the buy-condition, making it easier for the buy-threshold to be crossed. This additional adjustment encourages the algorithm to execute more buy actions so long there is a price increase, regardless of volatility in the market. For example, even if the neural network were to predict no change in stock price (predicted_ratio = 1), the algorithm would still trigger a buy action with a sufficiently low threshold, because the effective prediction_ratio under the trading strategy is now 1.1 at baseline. Our rationale for manually applying this additional scaling factor was motivated by the fact that Google is widely regarded as a growth stock, which is an asset expected to appreciate significantly over time. Hence, we decided to imbue the trading algorithm with a stronger impetus to purchase Google's shares, under the assumption that this is a generally favourable strategy in the long run.

```
# Using dynamic thresholds for buying and selling decisions
trading_strategy <- function(data, predictions, params=c(1.5, 1.5, 2.5),
                             initial_cash = 10000, transaction_fee = 5) {
```

```r
# Params is a list of buy, sell, stop-loss multipliers evolved via GA
params <- as.numeric(params)

position <- 0 # out of market - no shares currently
cash <- initial_cash # starting cash
portfolio_value <- numeric(nrow(data)) # store value of portfolio (cash+shares)
trades <- data.frame(Date = as.Date(character()),
                     Action = character(),
                     Price = numeric(),
                     Shares = numeric(),
                     Cash = numeric(),
                     Portfolio_Value = numeric(),
                     stringsAsFactors = FALSE)

actual_prices <- data$Close # actual closing prices used to evaluate trading conditions
shares_held <- 0
stop_loss_level <- NA # start as NA

for (i in 10:(nrow(data) - 1)) { # Start from 10 to have enough history
  current_price <- actual_prices[i]  # closing price for current date
  predicted_ratio <- predictions[i] / current_price # ratio shows predicted upward or downward movement

  # Calculate dynamic thresholds based on recent volatility
  recent_volatility <- sd(data$Daily_Return[(i-9):i], na.rm = TRUE) # std dev of past 10 days daily returns
  buy_threshold <- 1 + recent_volatility * params[1]
  sell_threshold <- 1 - recent_volatility * params[2]

  # Update stop loss (trailing at recent low - multiplier*ATR)
  atr_value <- data$ATR[i] # ATR for the current day to reflect price volatility.
  current_low <- min(data$Low[(i-5):i]) # lowest price over the last 5 days.
  stop_loss_level <- current_low - params[3] * atr_value # stop-loss below current low, adjusted by ATR*multiplier

  # Buy condition - if not owning any shares and the predicted ratio exceeds buy threshold
  if (position == 0 && (1.1*predicted_ratio) > buy_threshold) {
    max_shares <- floor((cash - transaction_fee) / current_price) # max no. of shares that can be purchased
    if (max_shares > 0) {
      shares_held <- max_shares # updates to the number of shares bought
      cash <- cash - (shares_held * current_price) - transaction_fee # reduce by share cost and transaction fee
      position <- 1 # switching to 1 indicating holding of shares
      new_trade <- data.frame(
        Date=data$Date[i], # Trade Execution Date
        Action="BUY", Price=current_price, Shares=shares_held, Cash=cash,
        Portfolio_Value = cash + shares_held * current_price)
      trades <- rbind(trades, new_trade)
    }
  }
  # Sell condition (either through sell-signal or stop-loss)
  else if (position == 1 &&
           (predicted_ratio < sell_threshold || current_price < stop_loss_level)) {
    cash <- cash + (shares_held * current_price) - transaction_fee
    position <- 0
    sell_reason <- ifelse(current_price < stop_loss_level, "STOP_LOSS", "PREDICTION")
    new_trade <- data.frame(
      Date=data$Date[i], Action=paste("SELL", sell_reason),
      Price=current_price, Shares=shares_held, Cash=cash,
      Portfolio_Value=cash)
    trades <- rbind(trades, new_trade)
    shares_held <- 0
    stop_loss_level <- NA
  }
  portfolio_value[i] <- cash + (shares_held * current_price)
}
```

```
  # Final sell if still holding position
  if (position == 1) {
    final_price <- actual_prices[nrow(data)]
    cash <- cash + (shares_held * final_price) - transaction_fee
    new_trade <- data.frame(
      Date=data$Date[nrow(data)], Action="SELL FINAL", Price=final_price,
      Shares=shares_held, Cash=cash, Portfolio_Value = cash)
    trades <- rbind(trades, new_trade)
    shares_held <- 0
  }
  portfolio_value[nrow(data)] <- cash

  return(list(
    portfolio_value = portfolio_value,
    trades = trades,
    final_cash = cash,
    metrics = list(
      return_pct = ((cash / initial_cash)^(252 / nrow(data)) - 1) * 100,
      sharpe_ratio = mean(diff(log(portfolio_value[portfolio_value > 0]))) /
        sd(diff(log(portfolio_value[portfolio_value > 0]))) * sqrt(252)
    )))
}
```

Fitness Function & Genetic Algorithm

The fitness function used for the genetic algorithm aims to maximise the Sharpe ratio. This approach looks to maximise the capital gains while remaining risk-averse, ensuring that the genetic algorithm avoids overfitting to overly-aggressive high-risk and high-return trades. Based on the GA plot, a solution that maximised the fitness function was reached very quickly.

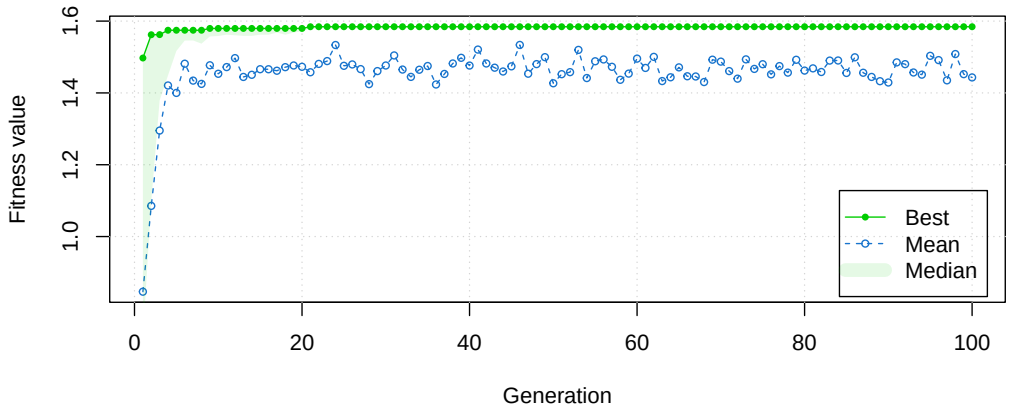
The genetic algorithm chromosome comprises 3 real values representing the buy/sell/stop-loss multipliers respectively. The upper and lower bounds for values were set to give the algorithm room to explore various reasonable threshold levels within these ranges. The population size of 100 gives sufficient diversity for candidate solutions. This GA configuration effectively achieves a balance between exploration (80% crossover, 40% mutation) and exploitation (keepBest=True, elitism=2) for robust portfolio search, with a slight emphasis on exploration. Meanwhile, gareal_blxCrossover encourages controlled exploration by sampling values slightly beyond the parents' values.

```
fitness_function <- function(params, data, predictions) {
  buy_mult  <- params[1]
  sell_mult <- params[2]
  stop_mult <- params[3]

  result <- trading_strategy(
    data, predictions,
    params = list(
      buy=buy_mult, sell=sell_mult, stop=stop_mult))
  return(result$metrics$sharpe_ratio) # Return Sharpe ratio as fitness
}

# Genetic algorithm
GA <- ga(
  type = "real-valued",
  fitness = function(params) fitness_function(params, train_data, train_pred_denorm),
  lower = c(0.1, 0.1, 1.0), # Min values for buy, sell, stop-loss multipliers
  upper = c(3.0, 3.0, 5.0), # Max values
  popSize = 100,
  maxiter = 100,
  crossover = gareal_blxCrossover,
  selection = gareal_tourSelection,
  keepBest = TRUE,
  pcrossover = 0.8,
  pmutation = 0.4,
  elitism = 2,
```

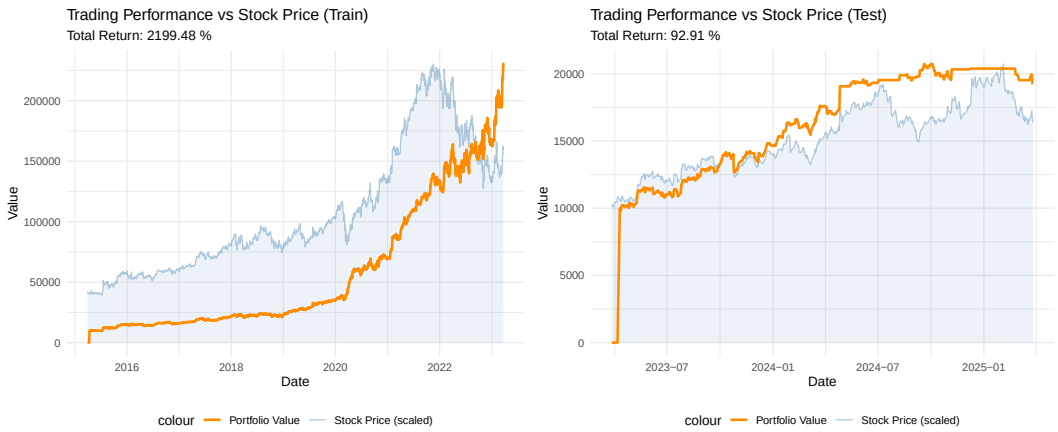
```
  seed = 42,
  parallel = TRUE
)
plot(GA) # plot evolution of solution using GA
```



The algorithm quickly improves within the first few generations and reaches a plateau around 1.6, indicating convergence. The average fitness value across the population each generation fluctuates initially but stabilizes around 1.4, suggesting the algorithm is refining solutions. Overall, the model finds good solutions early, but after a few generations, improvements slow down, while the plateaus of fitness suggests consistent optimal solutions are being found.

Performance Evaluation (Train & Test)

```
##  buy_mult sell_mult stop_mult
## 2.5109765 0.4228134 2.6973576
```




```
##                   Metric     Train     Test
## 1     Initial Capital ($)  10000.00 10000.00
## 2       Final Capital ($) 229947.73 19291.22
## 3   Annualised Return (%)     48.10    38.98
## 4 Annualised Sharpe Ratio      1.58     1.71
```

Training data performance: The results on the training data with the evolved multipliers are largely very successful, seeing steady growth while being able to dump its stocks before major dips in 2020 and 2022. Despite an entire year of falling stock price in 2022, the trading algorithm was still able to execute trades that allowed it to rally through this period. This yielded a total return of 2,199.48% (48.1% annualised) at the end of the training period, with a Sharpe ratio of 1.58. The multipliers evolved through the genetic algorithm are 2.5126 (buy), 0.4233 (sell), and 2.8153 (stop-loss). It can be quantitatively observed that the buy multiplier greatly exceeds the sell multiplier by approximately 5 times, meaning the buy threshold is much higher while the sell threshold is much lower. Effectively, the algorithm is more cautious to buy into the stock and quicker to

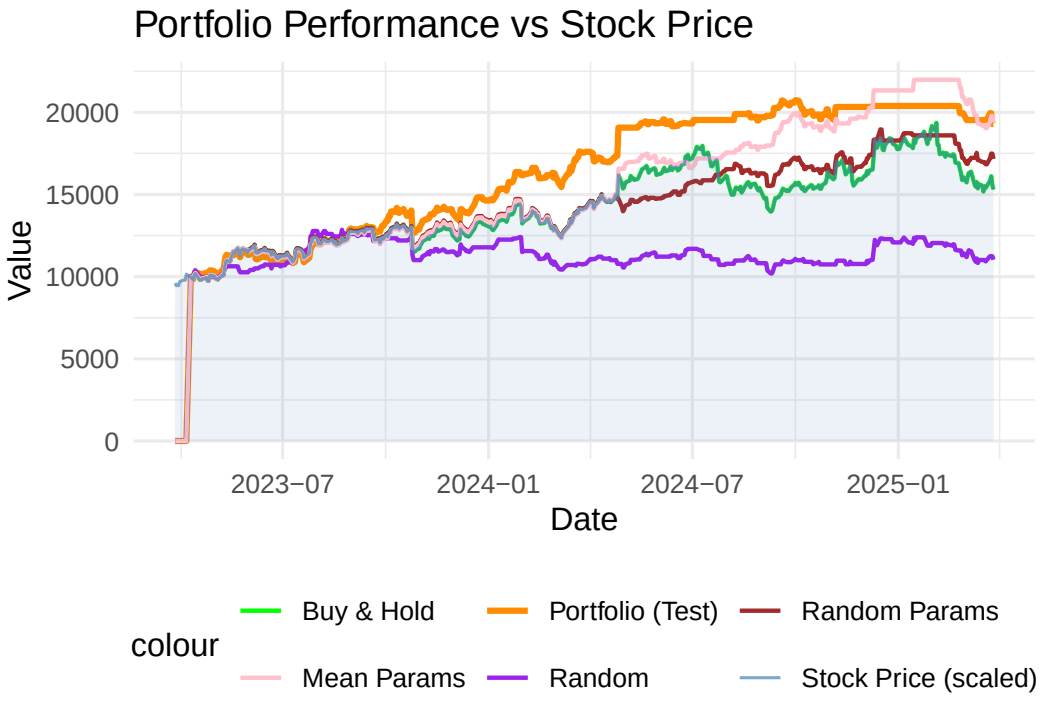
exit. This might indicate that price drops for Google's stock tend to be very sudden, and thus it favours trading strategies with lower sell-thresholds.

Test data performance: The combination of the previously trained neural network price-prediction model and genetically-evolved trading strategy (referred to as the "trading algorithm" from hereon) was used for backtesting on the test data. The results were just as successful, attaining total returns of 92.91% (38.98% annualised) and an annualised Sharpe ratio of 1.71. Given an initial capital of US$10,000, the algorithm has managed to make a realised gain of US$9,291.22 over the backtesting period. Observing the plotted performance, it starts with a sharp increase, followed by fluctuations that somewhat aligns with the stock's performance, but maintains an upward trend that eventually outperforms the stock itself. The algorithm also managed to anticipate a huge price drop around July 2024 and successfully divested from it. From the portfolio performance plotted on the test data, there was a period of unusually low activity between 6 Nov 2024 and 24 Feb 2025 wherein minimal stock purchases were made, as evidenced by the flatline during those periods and the gap in the trade history. When compared against the neural network prediction accuracy plot, it can be observed that it coincides with the period where the neural network was underpredicting Google's stock price. More precisely, the period of the underpredictions had started much earlier around April 2024. This is further corroborated by the scatterplot comparing predicted and actual prices, which shows that the model tends to underpredict stock prices when it rose above $150 a share, which also happened around April 2024. This is likely due to how the model has not been trained on prices more than $150 a share, given the fact that the maximum closing price in the training dataset is $150.71 on 18 Nov 2021, and would struggle to extrapolate its predictive ability to data above this price. As such, it tends to output price predictions that are closer to the range of the data it had been trained on. This underprediction effectively caused the algorithm to adopt a more pessimistic outlook of the future prices, which made it more difficult for the prediction ratio to exceed the buy threshold to execute stock purchases. If the neural network's predictions had closely matched the actual prices, this might have led to a strong enough buy signal to trigger a stock purchase. This observation has thus underscored the importance of an accurate price prediction model, as this would have a strong effect on the execution of the trade under the genetically-evolved trading rules.

Training vs. Test Performance: The profitability on test dataset remains strong despite the drop of 10% from train data which are mainly attributed to the unseen prices. The Sharpe ratio slightly improves in testing ($1.71 > 1.58$), which is positive—indicating that despite lower profitability, the model maintains good risk-adjusted returns. This suggests efficient trade execution even under different conditions.

Comparisons against other approaches

```
##                   Metric Buy_Hold   Random Random_Param Mean_Param
## 1     Initial Capital ($) 10000.00 10000.00     10000.00   10000.00
## 2       Final Capital ($) 15303.09 11030.47     17162.79   19404.42
## 3   Annualised Return (%)    23.76     5.04        31.08      39.39
## 4 Annualised Sharpe Ratio     0.78     0.25         1.20       1.48
```



1. Buy and hold: This strategy serves as a reasonable baseline for our trading algorithm, since it only makes sense to implement an active trading algorithm if it outperforms absolute passive investment. At the very least, this is a reasonable strategy that demonstrates consistent long-term profitability. The 23.76% return and 0.78 Sharpe ratio from buying and holding are significantly lower than the algorithm's 38.98% annualised return and 1.71 Sharpe ratio. The trading algorithm's higher return compared to the buy-and-hold strategy is to be expected, since a sell signal is triggered when it anticipates a dip that exceeds the sell threshold, thereby cutting its losses and re-entering when it anticipates a price upswing. The trading algorithm's much higher Sharpe ratio could also be due to how it experiences less volatility. When the trader is not holding any shares and the stock price is volatile, the buy threshold increases proportional to this volatility and makes it less likely for the trading algorithm to buy the stocks. As a result, the trading algorithm does not experience these periods of high volatility as compared to the trader that holds it throughout these periods.
2. Random buy/sell/hold actions: Another benchmark to use would be the performance of a trader that executes random trade actions from day to day. The results are extremely poor, ostensibly due to a lack of refined decision-making, with a 5.04% annualised return and Sharpe ratio of 0.25. This highlights the need for a robust and rational trading algorithm to be employed if one were to trade actively in the stock market, as an irrational or random trading algorithm could be demonstrably worse than a passive investing strategy.
3. Trading strategy with random buy/sell/stop-loss multipliers: This strategy makes use of the neural network and trading rules from our trading algorithm. However, the buy/sell/stop-loss threshold multipliers are randomly-generated (within the lower and upper bounds defined in the GA) rather than evolved using a GA. The result looks somewhat decent due to the bounds defined in the GA, with an annualised return of 31.08% and Sharpe ratio of 1.20, but still fall short of the GA-evolved multipliers since the randomly-generated multipliers are not guaranteed to be optimal.
4. Trading strategy with mean buy/sell/stop-loss multipliers: Similar to the previous comparison except that the buy/sell/stop-loss threshold multipliers were arbitrarily selected by picking the median values of the lower and upper limits defined in the genetic algorithm. Although this trading strategy yielded a higher annualised return of 39.39% than the trading algorithm, the Sharpe ratio was lower at 1.48. This is not unusual, given that the genetic algorithm aimed to maximise the Sharpe ratio fitness function rather than just returns alone.

From the combined plot, it can be more visually understood how each trading strategy performed relative to one another, as well as against the stock itself. The buy-and-hold strategy effectively serves as a proxy for the performance of the stock. From the plot, it (green line) shows a steady increase in value, indicating consistent growth. It can be observed that only the 3 trading strategies aided by the price prediction model were able to outperform the stock itself. Of these 3, the one that used the median multipliers (pink line) obtained the highest returns, albeit only marginally higher than the trading algorithm used. Yet, its Sharpe ratio was still lower than the trading algorithm's, which means that the trading rules used were not optimised to minimise risk in its strategy. Meanwhile, the random multiplier strategy (red line) exhibits volatility with periods of upward momentum.

Conclusion

The methodology employed in this analysis is designed to optimize stock price prediction and trading strategy effectiveness. By carefully selecting indicators, employing robust feature engineering, designing an adaptive neural network, and implementing a dynamic trading strategy through a GA , the model provides a sophisticated framework for stock market forecasting and automated trading.

Suggestions for Future Research

Future research could explore hybrid feature selection techniques, combining RFE with deep learning-based methods like autoencoders for enhanced input optimization. Alternative neural network architectures such as transformers or attention-based models could further improve prediction accuracy. Reinforcement learning could be integrated with GA to dynamically adapt trading strategies based on market conditions. Investigating multi-objective optimization in GA could balance profitability with risk management more effectively. Finally, testing the approach across different market regimes and asset classes would validate its robustness and generalisability.

Relative Contributions

Andrew Cheong - 50%

Panagiotis Daviotis - 50%

References

1. Allen, F., & Karjalainen, R. (1999). Using genetic algorithms to find technical trading rules. Journal of Financial Economics, 51(2), 245–271. https://doi.org/10.1016/S0304-405X(98)00052-X
2. Dronyuk, I., Klapchuk, M., & Singh, D. S. P. (2025). Stock market values prediction using deep neural networks. In Z. Hu, F. Yanovsky, I. Dychka, & M. He (Eds.), Advances in Computer Science for Engineering and Education VII. ICCSEEA 2024 (Vol. 242). Springer, Cham. https://doi.org/10.1007/978-3-031-84228-3_20
3. Dworkis, M., & Huang, D. (2008). Genetic algorithms and investment strategy development. Wharton Research Scholars, 87. http://repository.upenn.edu/wharton_research_scholars/87
4. Hijazi, O., Tikito, K., & Ouazzani-Touhami, K. (2025). A systematic review of artificial intelligence models applied to prediction in the stock market. In S. Assoul, B. El Bhiri, R. Saidi, & E. D. Yves Frederic (Eds.), Communication and Information Technologies through the Lens of Innovation. ICATH 2023. Advances in Science, Technology & Innovation (pp. XX–XX). Springer, Cham. https://doi.org/10.1007/978-3-031-74470-9_30
5. Hryshko, A., & Downs, T. (2004). System for foreign exchange trading using genetic algorithms and reinforcement learning. International Journal of Systems Science, 35(13–14), 763–774. https://doi.org/10.1080/00207720412331303697
6. Htun, H. H., Biehl, M., & Petkov, N. (2023). Survey of feature selection and extraction techniques for stock market prediction. Financial Innovation, 9, 26. https://doi.org/10.1186/s40854-022-00441-7
7. Kuepper, J. (2025, January 12). Using genetic algorithms to forecast financial markets. Investopedia. https://www.investopedia.com/articles/financial-theory/11/using-genetic-algorithms-forecast-financial-markets.asp
8. Nabipour, M., Nayyeri, P., Jabani, H., Shahab, S. S., & Mosavi, A. (2020). Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data: A comparative analysis. IEEE Access, 8, 210640–210655. https://doi.org/10.1109/ACCESS.2020.3015966
9. Naik, N., & Mohan, B. R. (2019). Optimal feature selection of technical indicators and stock prediction using machine learning techniques. In A. Somani, S. Ramakrishna, A. Chaudhary, C. Choudhary, & B. Agarwal (Eds.), Emerging technologies in computer engineering: Microservices in big data analytics. ICETCE 2019. Communications in Computer and Information Science (Vol. 985). Springer, Singapore. https://doi.org/10.1007/978-981-13-8300-7_22
10. Padalkar, S. P., Bere, S. S., & Hanchate, D. B. (2024). Prediction of stocks and stock price using artificial intelligence. International Journal of Creative Research Thoughts, 12(5). https://www.ijcrt.org
11. Singh, H., & Malhotra, M. (2024). Stock market and securities index prediction using artificial intelligence: A systematic review. Multidisciplinary Reviews, 7, Article 2024060. https://doi.org/10.31893/multirev.2024060
12. Treleaven, P., Galas, M., & Lalchand, V. (2013). Algorithmic trading review. Communications of the ACM, 56(11), 76–85. https://doi.org/10.1145/2500117
13. Li, X. (2022). Application of RNN and LSTM architectures for stock price forecasting. In Proceedings of the 5th International Conference on Economic Management and Green Development (pp. 196–200). AEPS. https://doi.org/10.1007/978-981-19-0564-3_20