Multiforest Trading Algorithm: A Novel Framework for Equity Price Prediction Using Disrupted Time-Series Data

Jaideep Padhi * and Clayton Greenberg

Posted Date: 14 April 2025

doi: 10.20944/preprints202504.1082.v1

Keywords: stock price prediction; machine learning; time-series forecasting; binary classification; disrupted time-series data

 doi:10.20944/preprints202504.1082.v1

Article

Multiforest Trading Algorithm: A Novel Framework for Equity Price Prediction Using Disrupted Time-Series Data

Jaideep Padhi * and Clayton Greenberg

Dover-Sherborn Regional High School; cgreenbe@alumni.princeton.edu

* Correspondence: jaideepsaipadhi@gmail.com

Abstract: The application of artificial intelligence in stock price forecasting is an important area of research at the intersection of finance and computer science, with machine learning techniques aimed at predicting future price movements, seeking consistent and profitable financial outcomes. However, due to the volatile nature of financial markets, generating predictions regarding an equity's future performance is challenging due to the complex and diverse factors that influence stock price dynamics, particularly concerning intraday movements. Prior research primarily focuses on hyperparameter optimization, feature engineering, and hybridization, often overlooking fundamental modifications to model and data architecture. Altering model and data architecture during model creation can significantly enhance model performance under real-time market conditions, to a greater extent than the aforementioned methods. Time-series forecasting in equity prices consists of two dimensions: magnitude and direction. Current algorithms used for stock price prediction have reduced efficiency as they attempt to forecast both dimensions with a single model. This paper introduces the Multi- forest model, a novel approach to stock price prediction that implements a bilayer machine learning algorithm combining sequential binary classification processes and regression processes to increase prediction accuracy. Although the classification process disrupts the continuity of the time-series data, the regressor effectively gen- erates valid predictions, dispelling notions that a complete time-series is required for accurate predictions. The Multiforest Trading Algorithm(MTA) demonstrates effectiveness during temporal deployments, providing success rates of 93.4%, 94.1%, and 84.0% for April 2024, June 2024, and October 2024, respectively, months dif- fering greatly in volatility and overall performance. When compared to models currently implemented in stock price prediction, the MTA outperformed all by a minimum margin of 15%, providing consistent results and exhibiting cautionary behavior when faced with volatile market conditions. Regarding profitability, the algorithm produced profit factors of 28.8, 31.0, and 15.0 for each respective month in the temporal deployments, indicating a projected profitability between 5-7 times greater than that of current algorithms.

Keywords: stock price prediction; machine learning; time-series forecasting; binary classification; disrupted time-series data

1. Introduction and Background

Time-series forecasting using machine learning techniques has applications not only in equity price prediction but in other financial markets. However, for research purposes, the stock market is a viable starting point due to its high trading volume, volatility, and speculation compared to other financial markets. Machine learning models commonly used in stock price prediction include the Long Short-Term Memory (LSTM) network, the Support Vector Machine (SVM) model, and the K-Nearest Neighbors (KNN) model. Although effective in training and testing, these models are less

 doi:10.20944/preprints202504.1082.v1

efficient in real-time execution, primarily due their reliance on a single model to predict both dimensions of price movement. Recent studies have aimed to im- prove these models through surface-level techniques, which marginally improve model perfor- mance. Hybridization between machine learning models is a frequently used optimization tech- nique, with the LSTM-ARIMA model [1] being a notable example applied to stock price pre- diction. As evaluated by S. Kulshreshtha and Vijayalakhsmi A., the LSTM-ARIMA hybrid algorithm [2] outperforms baseline models in effectiveness. However, the model demonstrates a Mean Absolute Percentage Error (MAPE) of 2.8% and requires 10 years of historical price data to gen- erate predictions. Therefore, for intraday price movements, such models may not accurately pre- dict future closing prices and are unsuitable for integration with trading algorithms. Despite its limitations, this hybrid model approach outperforms standalone models. A study by Prashant Pilla and Raj Mekonen predicting S&P 500 Index [3] prices using a standalone LSTM network [4] reports a MAPE of 6.4%. Although the numerical metric is not explicitly provided in the paper, the MAPE is approximated based on the Mean Absolute Error(MAE) and the index's average val- uation during the prediction period (2015-2019). Another study by H. Y. Kim and C. H. Won integrates a Generalized Autoregressive Conditional Heteroskedasticity(GARCH) model and an LSTM network to predict stock price volatility [5]. The logic behind this combination is that the GARCH model accurately captures the statistical properties of stock returns while the LSTM net- work incorporates these statistics as well as historical price data to generate predictions. The hybrid model, named GEW-LSTM, produced a Mean Absolute Error(MAE) of 0.0107 when pre- dicting future volatility of the KOSPI 200 Index [6], 37.2% less than the MAE produced by the E-DFN [5] model, the best-performing model at the time. The model also produced a Heterosce- dasticity Adjusted Mean Absolute Error(HMAE) and Heteroscedasticity Adjusted Mean Squared Error(HMSE) which are 24.8% and 48.0% lower than those of that produced by the E-DFN mo- del, respectively. Hybrid models exhibit greater predictive power than baseline models, and the evidence provided by both prior studies was the catalyst for this study's exploration of hybridi- zation and bilayer algorithms.

2. Methods

2.1. Software Architecture

The Random Forest Regressor(RFR) and the Random Forest Classifier(RFC) are well- suited for daily equity price prediction due to their ability to: a) handle non-linearity in feature variables and b) mitigate overfitting through ensemble decision trees. The Multiforest model integrates a combination of both a Random Forest Classifier and a Random Forest Regressor. After training, the Random Forest Classifier categorizes each trading session based on its 09:30 AM Open price and 09:30 AM Close(10:30 AM Open) price as either a *good* or *bad* session. By supplying the classifier with the 09:30 AM Open price and 09:30 AM Close(10:30 AM Open) price of the current session, the model predicts whether the session will be *good* or *bad*. The classified sessions are then separated into two datasets each serving as the respective train- ing/testing dataset for its corresponding regression model. Specifically, the *Good Sessions* re- gressor trains and tests on the *Good Sessions* dataset, and the *Bad Sessions* regressor utilizes the *Bad Sessions* dataset. Following the classifier's prediction, the scaled and transformed feature variables are provided to the corresponding to generate the final closing price prediction. If the model predicts the current session to be a *good* session, then the *Good Sessions* regression model is employed to predict the closing price of the next *good* session, which it treats as the current session. If the model classifies the current session as a *bad* session, the *Bad Sessions* regression model is applied to predict the closing price of the next *bad* session. A detailed workflow map of the Multiforest model architecture appears in Figure 1 for reference.

 doi:10.20944/preprints202504.1082.v1

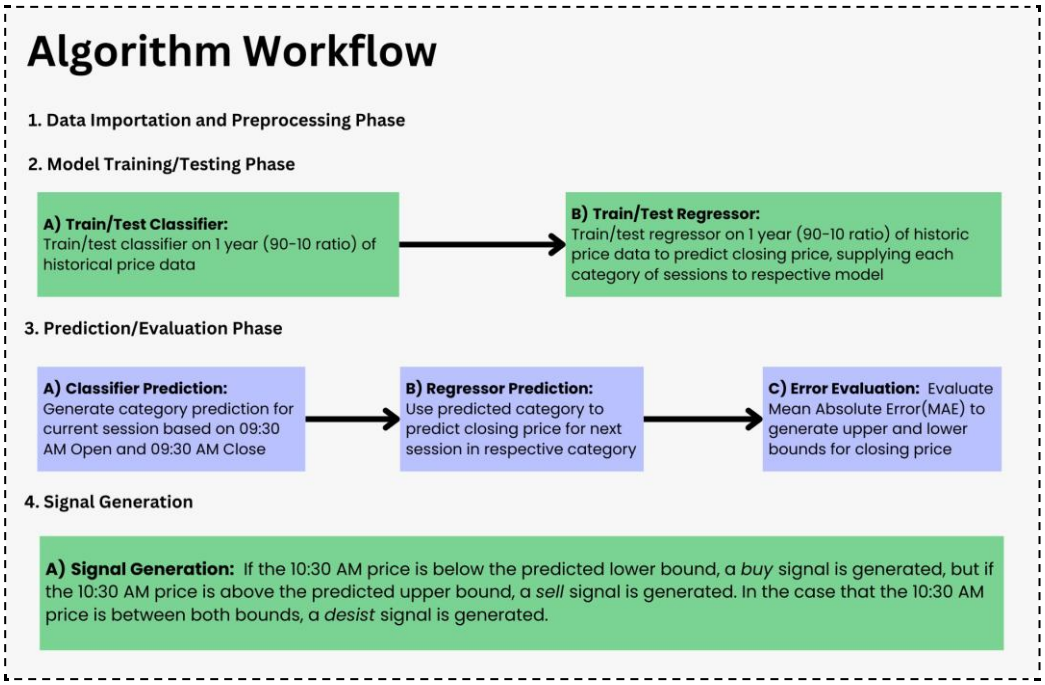


Figure 1. Algorithm workflow for the MTA implementing a series of binary classification and ensemble regression **1.** A depiction of training and testing processes involved in model creation **2.** An outline of prediction processes in the algorithm as well as its usage of errors generated in defining bounds **3.** The rules governing trade signal generation.

2.2. Model Training/Testing Dataset

The Multiforest model trains on one year of historical price data for the security being evaluated. For each execution, two datasets are imported via the YFinance API [7]: one containing hourly price data and the other containing daily price data. The daily price dataset includes a security's Open and Close prices for each trading session, while the hourly dataset captures trading data at hourly intervals beginning at the market open of 14:30 PM GMT (09:30 AM EST). The hourly dataset filters for the 14:30 PM GMT entry, collecting the session's 09:30 AM EST Open price and the 10:30 AM EST Open price of the security. In both datasets, the High, Low, and Volume columns are omitted since they do not contribute to the model's training or pre- dictions. After filtering, the data is divided, with 90% used for training purposes (225 sessions), and 10% for testing purposes (25 sessions). The securities used to evaluate the MTA are all constituents of the Standard & Poor's 500(S&P 500) Index. For each security in the index, price data is imported, filtered, and transformed similar to the Multiforest model.

2.3. Data Transformation

The imported price data undergoes transformation for classifier training by comparing the 4:00 PM EST closing price with the 9:30 AM EST opening price for each session in the dataset. Trading sessions are classified as *good* sessions if the closing price exceeds the opening price and as *bad* sessions if the closing price falls below the opening price. The data is then split into a 90–10% distribution for training and testing purposes, respectively, before being scaled for processing by the classifier and the regressor.

2.4. Stationarity

During real-time implementation of the Multiforest model, the initial version, which forecasts an equity's closing price, was effective for a substantial majority of equities in the S&P 500 but generated anomalous predictions for a small subset of securities. Identifying these anomalous errors

 doi:10.20944/preprints202504.1082.v1

within the prediction dataset is straightforward, as the predicted price range deviates from the actual price by a substantial percentage that is uncommon during intraday price move- ments. However, when implementing the Multiforest model in real-time scenarios or integrating it into a trading algorithm, consistency in predictions is prioritized. A potential hypothesis regarding these anomalous predictions is the influence of non-stationary data on the regressor's perform- ance, as regression models typically require stationary data to generate accurate predictions.

Stock price data is inherently non-stationary due to several factors including: a) long-term directional trends affecting its price movements, b) seasonality in price data, and c) changes in vola- tility. The non-stationarity of the data is further exacerbated when divided into separate datasets, as periods with continuous sequences of *good* or *bad* sessions are entirely placed into one dataset, generating gaps in the other dataset, thus disrupting the time-series data provided to both models. The differences variation of the Multiforest model mitigates the effects of non-stationarity by training, testing, and predicting upon the difference factor between the 09:30 AM Open price and the 04:00 PM Close price, rather than the actual closing price. The differences variation is implemented by generating interaction terms within the dataset. A new column, *variable difference*, is generated by determining the difference factor between the 09:30 AM and 10:30 AM Open prices, serving as the feature variable for the model. Another column, *target difference*, is calculated as the difference factor between the 09:30 AM Open price and 04:00 PM Close price, serv- ing as the target variable. This approach reduces non-stationarity in the datasets by removing trends, seasonality, and stabilizing variance.

2.5. Trading Algorithm Logic

The logic governing the MTA is as follows: subsequent to the Multiforest model pre- dicting the current session's closing price, the Mean Absolute Deviation(MAD) is added to and subtracted from the predicted price to determine the predicted upper error bound and lower error bounds. The errors generated by the RFR during testing follow a normal distribution, a trend con- sistent across all predicted securities. Figure 2 provides a Q-Q plot visualization for a randomly sel- ected sample of four securities from the S&P 500 index, illustrating how the errors align with a theoretical normal distribution.

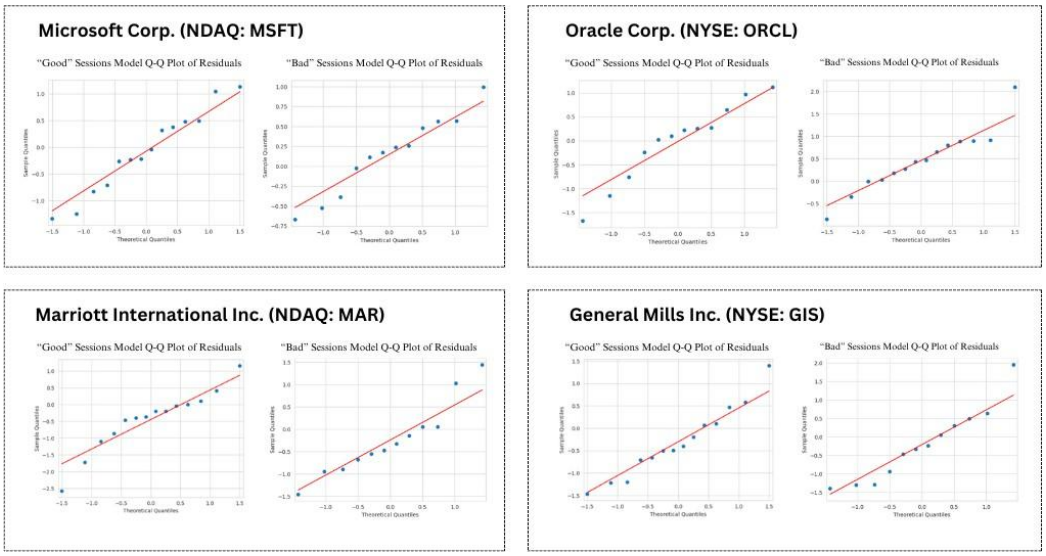


Figure 2. A) A layout of four Q-Q plot graphs of stocks [NDAQ: MSFT; NYSE:ORCL; NDAQ:MAR; NYSE: GIS], each depicting a normal distribution observed in error data, the basis of the trading algorithm **B)** errors were recalculated as percentage error to address distortion in the Q-Q plot due to long-term directional price movement frequently observed in equities.

In all four cases, the distribution of data points across the theoretical quantile lines in the Q-Q plot indicates that the error data follows a normal distribution. Increased spread from the reference

 doi:10.20944/preprints202504.1082.v1

line at the extremes is due to a few extreme data points recorded in the error data; how- ever, the majority of error data produced during predictions adheres to a normal distribution. The adherence of the model's error data to a standard distribution demonstrates stability in its per- formance and allows for the implementation of the empirical rule [8]. While the empirical rule serves as the basis for the MTA logic, the algorithm utilizes the MAD rather than Standard Dev- iation (STD) due to its relative robustness during temporal deployments when evaluating the up- per and lower bounds for the predicted closing price. For a normal distribution of data points, the typical relationship between MAD and STD is a 4:5 ratio [9]. As a result, using MAD as the basis of the algorithm generates bounds of reduced deviation relative to the predicted price, while still yielding more favorable trade signal datasets than those based on STD. The algorithm has three variations based on the number of MAD multiples used in predicted error bound calculation: *MAD_1* (1 multiple of the MAD determines predicted error bounds), *MAD_2* (2 multiples of the MAD determine predicted error bounds), and *MAD_3* (3 multiples of the MAD determine pre- dicted error bounds). When implementing the three variations of the algorithm in real-time exec- ution, it was observed that increasing the number of MAD multiples led to a decrease in the num- ber of signals generated by the algorithm. For example, in real-time implementation, the *MAD_1* variation typically generates over 150 signals per session, covering approximately 30-40% of the S&P 500 Index. The *MAD_2* variation, however, typically generates fewer than 40 signals per session (8-10% of the index), while the *MAD_3* variation generates approximately 10 signals per session(2-3% of the index). Increasing the number of multiples of the MAD used to determine error bounds filters out lower-confidence signals, leaving only signals that best represent the algorithm's confidence in their potential profitability. For this reason, the *MAD_3* variation was selected for the study, as it most accurately reflected the algorithm's confidence while still providing sufficient monthly data for research purposes.

3. Results and Discussion

3.1. Temporal Deployment of the Trading Algorithm

The differences variation of the MTA was deployed across three prior trading months to evaluate its performance over a substantial timeframe. The selected trading windows were April 2024, June 2024, and October 2024. The month of April 2024 represented a downward market with the S&P 500 index declining 4.0% overall. In contrast, the month of June 2024 represented an advancing market with the S&P 500 index increasing by 3.6% overall. The month of October 2024, however, represented a neutral market, declining only 0.9% over the month, the lowest absolute monthly percentage change observed in the 2024 fiscal year. The contrasting market performance during these months, alongside data importation limitations, informed the selection of these trading periods for evaluation. Figure 3A below presents the overall performance statistics for signals generated by the differences variation of the algorithm.

Differences Variation **(A)**	Total Num. of Trades	Num. of Profitable Trades	Num. of Loss Trades	Profit Factor
Apr. 2024	230	169	61	4.7
Jun. 2024	119	85	34	3.7
Oct. 2024	187	124	63	2.4

Original Algorithm **(B)**	Total Num. of Trades	Num. of Profitable Trades	Num. of Loss Trades	Profit Factor
Apr 2024	378	353	25	28.8
Jun 2024	374	352	22	31.0
Oct 2024	237	199	38	15.0

Figure 3. **A)** A table displaying performance statistics of the differences variation from the temporal deployment across the three months used in the study **B)** A display of collected performance statistics of the original version of the algorithm for the same period.

 doi:10.20944/preprints202504.1082.v1

The CBOE Volatility Index(VIX) [10] averaged 16.1 per day in April 2024, reflecting an increase of 17.7% from the previous month's average volatility of 13.7. In June 2024, the average volatility was 12.7, marking the least volatile month of the 2024 financial year. Although the S&P 500 Index experienced the lowest cumulative change in October 2024, the market experienced a significant increase in volatility, with the average daily VIX being 19.4 for the month. Thus, the varying number of signals generated by the algorithm can be attributed to market fluctuations and overall volatility levels. During the higher-volatility months of April 2024 and October 2024, the model recommended more trades compared to June 2024, a month of low volatility in the markets. The success rates for trade recommendations during these periods were 73.5% (April 2024), 71.4% (June 2024), and 66.3% (October 2024). This reflects an approximate 70% success rate for potential future trades using the differences variation of the algorithm. Collected performance statistics for the original version of the MTA are displayed in Figure 3B above. The original version of the MTA, which predicts closing prices rather than difference factors, was deployed across the same three timeframes: April 2024, June 2024, and October 2024. The success rates of trade rec- ommendations during these periods were 93.4% (April 2024), 94.1% (June 2024), and 84.0% (October 2024). This models an approximate 90% success rate for potential future trades by the original version of the algorithm. Therefore, this version of the Multiforest algorithm is better suited for real-time application than the differences variation due to its higher accuracy rate.

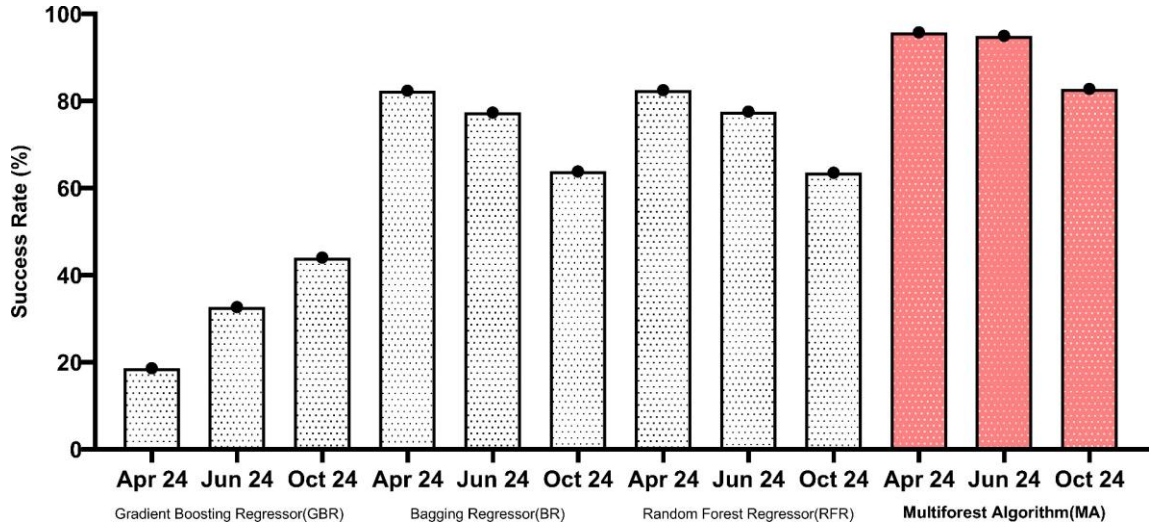
Based on deployment results, the differences variation of the model generated an average of 8.25 trade signals per trading session and the original version of the model presented 15.22 trade signals per trading session. Combining these averages as well as the success rates from the evaluation, the algorithm predicts an average of 5.78 winning trades and 2.48 losing trades per trading session for the differences variation, compared to an average of 13.70 winning trades and 1.52 losing trades per trading session for the original algorithm. When accounting for outlier predictions, typically limited to two signals per trading session, the original algorithm remains more profitable than the variation. The inclusion of these two trades as losses adjusts the averages to 13.70 winning and 3.52 losing trades per session. Considering the most unfavorable scenario in which all outlier predictions are losses, the adjusted win rate for the original version is 79.6%, still exceeding the success rate of the differences variation, and therefore serves as the basis for continued research.

3.2. Comparative Performance Analysis

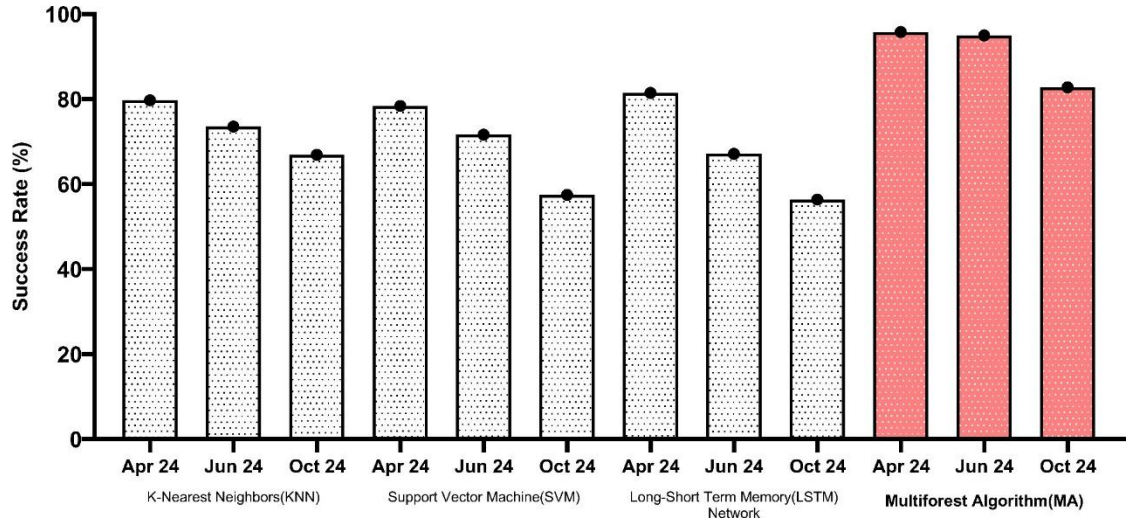
The Multiforest model was benchmarked against standalone regression models and artificial neural networks to evaluate its performance relative to other models frequently utilized in time-series forecasting. However, the Mean Absolute Error(MAE) produced by all comparison models for each predicted equity were of substantial nature. Consequently, the 10:30 AM price rarely lies outside the defined interval given by the predicted error bounds. Thus, the previous trading algorithm logic implemented in the MTA logic was ineffective. All evaluations provided in this paper are not on the basis of tuned hyperparameters, but rather on the default hyperparameter configuration provided during model creation. It was also observed that hyperparameter tuning did not significantly impact the Multiforest model performance during temporal deployments. Primarily, the predictive performance of the comparison models in comparison to the Multiforest algorithm was at question, and therefore the algorithm logic for each comparison model was reduced in complexity as a result. Specifically, the upper and lower predicted error bounds were excluded in the algorithm structure, and the predicted price was directly compared to the 10:30 AM price. In cases where the predicted price was greater than the 10:30 AM price, a *buy* signal was issued, and a *sell* signal was issued otherwise. As a result, the algorithm generates trade signals for all stocks in the index, omitting the selection process used by the Multiforest algorithm. Figure 4A,B presents the success rates for each comparison model relative to the MTA when deployed on three randomly chosen trading sessions from each of the months of April 2024, June 2024, and October 2024 – the same time periods used in the study. The chosen sessions were April 3, April 17, and April 24 (April 2024); June 12, June 21, and June 27(June 2024); and October 4, October 14, and October 29 (October 2024). Due to the architectural

 doi:10.20944/preprints202504.1082.v1

constraints of the neural network models implemented in the study, the usage of both the 09:30 AM Open price and the 10:30 AM Open price as feature variables was not feasible. Therefore, pre- dictions for the neural network models were generated solely based on the 09:30 AM Open price.



(A)



(B)

Figure 4. **A:** A visualization of results of the samples evaluated during temporal deployments for comparison regression models relative to the results of the MTA. **B: A)** A visualization of the results of samples evaluated during temporal deployments for comparison neural network models in relation to the results of the MTA.

The Multiforest Algorithm is a more effective approach for stock price prediction than all evaluated models as the MTA outperforms all comparison models when evaluated based on success rate. The algorithm is also successful in adapting to the effects of consistent volatility, producing fewer signals when faced with higher volatility markets. During October 2024, a month of higher volatility in financial markets, the algorithm generated only 64 signals for the sample days

evaluated. This contrasts significantly with the number of trades generated by the algorithm for the sample days evaluated from June 2024, a month of low volatility in financial markets, as there were 220 signals generated for this period.

 doi:10.20944/preprints202504.1082.v1

4. Conclusions

This paper introduces the Multiforest Model, a bilayer machine learning algorithm that integrates binary classification and regression processes to predict intraday price movements for equities. Although the algorithm's classification technique disrupts the continuity of time-series data, it is more effective than other regression and neural network models that rely on a complete time-series. While prior research focuses on hyperparameter optimization, feature engineering, and hybridization to improve prediction accuracy, this study explores a classification –before– regression architecture as a method for generating more accurate predictions. The classifier pre- dicts the direction of intraday movement, while the regressor predicts the magnitude of move- ment. The Multiforest model's performance was evaluated against six machine learning models commonly used for stock price prediction: the Random Forest Regressor(RFR), the Gradient Boosting Regressor(GBR), the Bagging Regressor(BR), the LSTM network, the AutoRegressive Integrated Moving Average(ARIMA) model, and the Convolutional Neural Network(CNN) model. Results from temporal deployments demonstrate that the Multiforest model is more effi- cient and profitable in comparison to many algorithms currently in use. It should be noted that, in its current state, the Multiforest model cannot be directly integrated with trading mechanisms for real-time order executions, as the current algorithm lacks API integrations with major trading platforms. The results presented in this paper are based on a specific view of financial markets, emphasizing the model's applicability solely to the S&P 500 Index. However, the Multiforest model's usability extends beyond the stock market, encompassing financial instruments such as currency pairs, exchange-traded funds (ETFs), and commodities. The Multiforest model is not recommended for derivative instruments such as futures contracts, options contracts, and swaps due to time-based factors such as theta-decay and extreme levels of volatility frequently observed in these instruments. In conclusion, the Multiforest algorithm is an effective approach to stock price prediction, accurately predicting the future direction of trading signals for the remainder of the session in many cases, delivering desirable and consistent returns.

5. Extensions

5.1. Weightage of Trades

In its current state, the MTA selects stocks from the S&P 500 Index for the current trading session that it predicts will make a definitive shift in the predicted direction. However, the trading algorithm does not prioritize trade signals, assigning equal weighting to each security. To optimize this aspect of the algorithm, a numerical index score can be calculated at the time of prediction, allowing the algorithm to assign greater or lesser weighting to specific signals. The index score is calculated by evaluating the geometric mean of the predicted minimum profit per- centage(MPP) and the predicted expected profit percentage(EPP). The portion of the daily trading volume allocated to each security in the portfolio is calculated by totalling the index scores for all securities and calculating the percentage composition of the total for the equity in question. Figure 5 illustrates an example containing weighted, ranked predictions from the trading session of January 28th, 2025.

	Stock	Predicted Price	Lower Price	Upper Price	Category	10:30AM_Price	Signal	Minimum Profit Percentage	Expected Profit Percentage	Index Score
2	ABT	124.702740	122.235918	127.169563	bad	129.210007	Sell	1.579169	3.488326	2.347052
131	DVA	171.602804	167.602130	175.603478	bad	178.212494	Sell	1.463992	3.708881	2.330187
0	MMM	146.260562	144.200371	148.320754	bad	150.339996	Sell	1.343117	2.713472	1.909060
134	DE	470.807998	461.895907	479.720089	bad	483.700012	Sell	0.822808	2.665291	1.480886
130	DRI	189.447719	185.858820	193.036618	bad	194.175003	Sell	0.586268	2.434548	1.194695
380	RTX	126.660806	124.576276	128.745337	bad	129.559998	Sell	0.628790	2.237721	1.186194
67	BSX	100.920202	99.924483	101.915921	bad	102.529999	Sell	0.598925	1.570074	0.969720
161	ETR	76.240052	75.537572	76.942531	bad	77.320000	Sell	0.488190	1.396725	0.825752
240	PODD	277.786205	273.482034	282.090375	good	282.690002	Sell	0.212115	1.734691	0.606592
412	SYF	66.592797	65.751698	67.433897	good	65.599998	Buy	0.231249	1.513413	0.591588

 doi:10.20944/preprints202504.1082.v1

Figure 5. Captured output from model execution in which the index score of all trade signals are calculated for weightage purposes.

As shown in Figure 5, the algorithm evaluates the geometric mean of each signal's projected minimum profit percentage and its expected profit percentage. Doing so categorizes trades with higher projected profitability as favorable choices for potential trades. The index score serves as a mechanism to evaluate the algorithm's confidence in the potential returns of a trade. The system calculates the index score for each trade signal, and, using the data from Jan 28th, 2025 as an example, allocates the portfolio trading volume as shown in Figure 6.

	0	1	2	3	4	5	6	7	8	9
Stock	ABT	DVA	MMM	DE	DRI	RTX	BSX	ETR	PODD	SYF
Trading Volume Percentage(%)	17.46	17.33	14.2	11.02	8.89	8.82	7.22	6.14	4.52	4.4

Figure 6. Captured output from weightage of trades execution in which the index scores for all signals were totalled and each trade percentage was in direct proportion to the signal's index score relative to other signals predicted.

5.2. Profit Thresholds and Drawdown Control

The current version of the trading algorithm does not execute real-time orders but pre- dicts the stock's daily performance direction and closing price, with the latter based on the pre- diction category of the former. Therefore, when evaluating the algorithm's performance from a monetary perspective, implementing profit thresholds and drawdown control would enhance the

model. Ideally, for a *buy* signal, the profit threshold would align with the algorithm's upper error bound, and the drawdown control point would be set at one multiple of the MAD below the lower error bound.

5.3. Parallelization for Runtime Optimization

During real-time evaluations, the algorithm required significant runtime to generate predictions, taking an average of 11 minutes to produce trade signals for the current session. Due to second-by-second price fluctuations and the dynamic nature of markets, runtime must be opti- mized and reduced to secure trades closest to the 10:30 AM price of the equity. Parallelization is a runtime optimization technique that enables the algorithm to fully utilize resources of the host machine. Process-based parallelization, implemented using the built-in *multiprocessing* library, distributes processes across the CPU cores of the host machine, maximizing computational re- sources to reduce runtime. Though less effective for this use case than process-based parallel- ization, thread-based parallelization uses the *threading* module to optimize computational re- sources. The Global Interpreter Lock(GIL) [11] is the primary hindrance to implementing thread

based parallelization in the algorithm. Only one thread can have possession of the GIL at a given time, limiting the potential of multiple cores available and is therefore an unsuitable approach for CPU-intensive tasks as required by the algorithm.

6. Supplementary Information

6.1. Abbreviations, Terms & Definition(s)

Minimum Profit Percentage(MPP)*:* The percentage difference between the 10:30 AM price of a security and either the lower or upper predicted bound, based on the directional signal given by the Multiforest model.

Expected Profit Percentage(EPP)*:* The percentage difference between the 10:30 AM price of a security and the actual predicted price provided by the Multiforest model.

 doi:10.20944/preprints202504.1082.v1

Standard & Poor's 500 Index(S&P 500): An abbreviation for the Standard & Poor's 500 Index, a collection of the top 500 largest corporations traded on stock exchanges in the United States, ranked by market capitalization.

Global Interpreter Lock(GIL): A mechanism used by the CPython interpreter to ensure thread safety by permitting only one thread to execute Python bytecode at a given time

*1_MAD Variation(*MAD_1*)*: A manipulation of the Random Forest Regressor error data in which 1 multiple of the Mean Absolute Error(MAD) are both added and subtracted from the predicted price to determine the upper and lower error bounds of the closing price

*2_MAD Variation(*MAD_2*)*: A manipulation of the Random Forest Regressor error data in which 2 multiples of the Mean Absolute Error(MAD) are both added and subtracted from the predicted price to determine the upper and lower error bounds of the closing price

*3_MAD Variation(*MAD_3*)*: A manipulation of the Random Forest Regressor error data in which 3 multiples of the Mean Absolute Error(MAD) are both added and subtracted from the predicted price to determine the upper and lower error bounds of the closing price

6.2. Disclaimer

The results outlined in this paper do not guarantee future outcomes, and the algorithm is presented for academic and educational purposes only. Financial markets inherently involve volatility and uncertainty that artificial intelligence may not detect during training and execution but will nonetheless influence model predictions. Therefore, the model may not always be accurate for financial decision-making. This model does not provide certified financial advice, and indi- viduals and third-parties seeking financial advice should consult a financial advisor. The authors affirm that this model is not a substitute for financial advice and are not liable for any financial decisions made by individuals or third parties based on the model's prediction or statistics.

6.3. Data Limitations & Availability

The primary limitation when importing data via the YFinance API is the API restriction on hourly data, allowing only two years of prior data to be downloaded relative to the present date. While there are no such limitations for daily price data importation, only two years of daily price data can be used due to the hourly dataset restriction. Counterintuitive to generally con- ceived notions, providing only one year of price data to the model for training and testing was more beneficial than providing two years of data when evaluating training and testing metrics.

One possible explanation is that, due to the dynamic nature of financial markets, machine learning models train more accurately when provided only with recent data rather than a mix of recent and marginally outdated data.

6.4. Model Limitations

The primary limitation of the Multiforest Trading Algorithm is its inability to detect sudden price shifts, particularly those in relation to earnings reports [12]. Sudden fluctuations in eq- uity price observed in trading sessions following earnings reports are more pronounced than those

observed during regular trading sessions. As a result, the algorithm may interpret these irregular price movements as indicators of either undervaluation or overvaluation for the current session, issuing *buy* or *sell* signals accordingly. The algorithm does not acknowledge these reports nor considers the outlined results when generating predictions, and therefore any produced signals may not be representative of any upsurges in purchase or sale volume throughout the remainder of the session. Consistent volatility, while less detrimental than the sudden volatility following earnings reports, also reduces the algorithm's performance. Performance evaluations for October 2024, for both the original version and the differences variation, show that consistent volatility reduces the algorithm's performance, as the ratio of profitable trades to losing trades for both versions is lowest in October 2024. The 2024 Presidential Elections [13], held on November 6th, increased overall

 doi:10.20944/preprints202504.1082.v1

volatility due to tensions over potential changes in foreign and domestic economic policies. As a result, consistent volatility, though lesser in magnitude than the volatility experienced during earnings reports anticipation, is nonetheless detrimental to the algorithm's performance when evaluated across a significant timeframe.

6.5. Author Contributions

Jaideep Padhi, an advisee of Dr. Clayton Greenberg, served as the primary researcher for the project. JP was involved in code generation, data analysis, data interpretation, and writing while CG was primarily involved in administration and advising. Both authors were involved in infrastructure conceptualization of the algorithm and interpretation of research results. Both authors edited, revised, read and approved the final manuscript.

6.6. Infrastructure Specifications

Package Dependencies: NumPy(v1.26.4), Pandas(v2.2.2), scikit-learn(v1.6.1), yfinance(v0.2.52)
CPU Specifications: (2) Intel Xeon single-core vCPUs with hyper-threading, 2.20Ghz, 56MB cache Platform Specifications: x86_64 Linux OS 6.1.85+ with glibc 2.3.5 Python Version: 3.11.11

References

1. K. Kashif and R. Ślepaczuk, "LSTM-ARIMA as a Hybrid Approach in Algorithmic Investment Strategies," *arXiv preprint arXiv:2406.18206*, 2024. [Online]. Available: https://arxiv.org/abs/2406.18206. [Accessed: Jan. 11, 2025].
2. S. Kulshreshtha and A. Vijayalakshmi, "An ARIMA-LSTM Hybrid Model for Stock Market Prediction Using Live Data," *J. Eng. Sci. Technol. Rev.*, vol. 13, no. 4, pp. 117–123, 2020. [Online]. Available: https://www.researchgate.net/publication/343992007_An_ARIMA-_LSTM_Hybrid_Model_for_Stock_Market_Prediction_Using_Live_Data. [Accessed: Jan. 17, 2025].
3. S&P Dow Jones Indices, "S&P 500 Index," *S&P Global*, [Online]. Available: https://www.spglobal.com/spdji/en/indices/equity/sp-500/. [Accessed: Jan. 21, 2025].
4. P. Pilla and R. Mekonen, "Forecasting S&P 500 Using LSTM Models," *arXiv preprint arXiv:2501.17366*, Jan. 2025. [Online]. Available: https://arxiv.org/abs/2501.17366. [Accessed: Jan. 23, 2025].
5. Kim, H., & Won, C. (2018). *Forecasting the Volatility of Stock Price Index: A Hybrid Model Integrating LSTM with Multiple GARCH-Type Models*. *Computational Economics*, 51(1), 29–49. https://doi.org/10.1007/s10614-017-9704-y [Accessed: Jan 27, 2025]
6. Korea Exchange, "KOSPI 200 Index," *Korea Exchange Official Website*, [Online]. Available: https://global.krx.co.kr. [Accessed: Jan. 28, 2025].
7. R. Roussi, "yfinance: Yahoo! Finance market data downloader," *GitHub Repository*, 2018. [Online]. Available: https://github.com/ranaroussi/yfinance. [Accessed: Jan. 31, 2025].
8. R. E. Walpole, R. H. Myers, S. L. Myers, and K. E. Ye, *Probability and Statistics for Engineers and Scientists*, 9th ed. Boston, MA, USA: Pearson, 2011.
9. J. F. Kenney and E. S. Keeping, *Mathematics of Statistics, Part One*, 3rd ed. Princeton, NJ: Van Nostrand, 1962.
10. Chicago Board Options Exchange, "CBOE Volatility Index (VIX)," *Cboe Global Markets*, [Online]. Available: https://www.cboe.com/tradable_products/vix/. [Accessed: Feb. 3, 2025].
11. Tony J. R. L. McMullin and Peter D. Zeitz, "Parallelism in Python: The Global Interpreter Lock (GIL)," *Journal of Computing Science and Engineering*, vol. 19, no. 4, pp. 122-134, 2019.
12. Medya, S., Manish, K., & Jain, R. (2022). *An Exploratory Study of Stock Price Movements from Earnings Calls*. arXiv. Available: https://arxiv.org/abs/2203.12460 [Accessed: Feb. 5, 2025].
13. SIFMA, "Markets and Elections," *SIFMA Insights*, Oct. 2024. [Online]. Available: https://www.sifma.org/wp-content/uploads/2024/10/SIFMA-Insights-Markets-Elections-10-2024.pdf. [Accessed: Feb. 5, 2025].

 doi:10.20944/preprints202504.1082.v1