Exploring the Extent of the Predictive Power of A Random Forest Regression Model in Predicting Same-Day Closing Stock Prices for Microsoft Corporation (NASDAQ: MSFT)

Rajit Gupta
International School of Kuala Lumpur.

Abstract

This study will investigate the predictive power of a Random Forest Regression model when trained to use basic input features – open, high, low, and volume live-data for a stock – to predict the stock's same-day closing price. Unlike the majority of literature, this model doesn't use any lagged or technical features which is purposefully done to test the predictive power of a minimalist model. In this paper, the Random Forest Regression model was trained on 4 years worth of data for the Microsoft Corporation stock (NASDAQ: MSFT) from the start of 2015 to the end of 2018. It was then tested to predict the same-day closing price for a forecasted 10 days using aforementioned features and achieved an R^2 value of approximately 0.89 and a low MSE and MAE of approximately 0.36 and 0.421, respectively. Despite the limitations of not accounting for exogenous factors or not having any technical features, this study still shows the strong predictive power of a simple model that can be easily applied in intra-day trading.

Keywords: *Random Forest Regression, Intra-Day Trading, Same-Day Closing Price Prediction*

Introduction

Stock price prediction has always remained a field of exploration and fascination for traders and those in finance and a longstanding goal. In recent years, there has been much more research in this field and increasing applications of machine learning models in forecasting stock prices and movement. While most of the research focuses on forecasting future, specifically next day, prices of movements based on current data, there is a relatively unexplored field of same-day price prediction, which this research paper will focus on. Specifically, in this paper I will explore the use of a Random Forest Regression model to predict the same-day closing price for the stock of Microsoft Corporation (NASDAQ: MSFT).

There are 2 types of Random Forest models, regression and classification. For this paper, I will be looking at a Random Forest regression model. These models are ensemble learning algorithms (Breiman, 2001) which combine multiple individual models to produce more robust predictions.

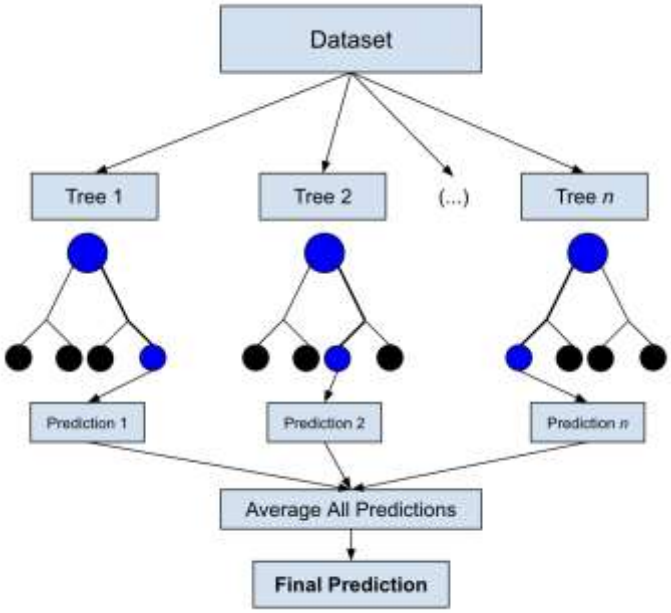

Figure 1: Visual Representation of Random Forest Regression Models

The figure above, created by me, shows how Random Forest Regression Models generate their predictions. This algorithm is made of countless individual decision trees – "forest" of decision trees – that are all fed random subsets of the data – referred to as a bootstrap sample – which is drawn with replacement (IBM, 2025). Within each tree, there is another layer of randomness from the feature randomization. This means "when making a split, each tree only considers a random subset of features (Baladram, 2024)." Each tree then splits until it grows to a minimum sample size and then makes its predictions using the features and dataset specific to the tree. For a regression model, as in this study, the final prediction generated by the algorithm is an average of the prediction from each individual tree. Averaging helps to reduce variance, leads to less errors, making the result more stable, more accurate, less noisy, and makes the model less likely to overfit (IBM, 2025). Their robustness and minimal processing requirements have made them a popular model used in financial forecasting. As seen in the paper by Krauss, Do, and Huck (2017), Random Forest models performed better than other kinds of models, such as deep neural networks, making them one of the most optimal models to use when predicting stock prices. Furthermore, Random Forest models use a variety of hyperparameters that can be tuned as required by the research and the dataset to optimize the model. In this paper, I focused on the following 4 key hyperparameters: 'n_estimators', 'max_depth', 'min_samples_split', and 'min_samples_leaf'. Firstly, 'n_estimators' determines the number of decision trees in a model. With a higher number of decision trees, a model tends to perform better due to more bootstrapped samples which lead to increased stability and reduced variance (Breiman, 2001). Secondly, the parameter 'max_depth' controls the depth of each tree," which is the maximum number of splits until the terminal node (Probst et al., 2019)." Next, the 'min_samples_split' parameter controls the "minimum number of samples required to split a node (Fraj, 2017)." According to Probst et al. (2019), this hyperparameter is important in preventing the model from over-segmenting the data and fitting to small fluctuations in the dataset. Lastly, the 'min_samples_leaf' parameter determines the number of samples required to be at every leaf node within a tree. These hyperparameters are vital in ensuring that the model performs to the best capability, and although there are many more hyperparameters that can be tuned, I chose to focus on these ones for the purposes of this study.

There is extensive literature in this field that goes into exploring the use of models such as Random Forest as well as other machine learning models to try to forecast and predict stock prices and movement. Random Forest models are also easy to train and deploy and are robust models (Krauss et al., 2017). Furthermore, the study from Wu (2022) also shows the supremacy of the Random Forest model when compared to XGBoost and Simple Decision Trees. In this paper, Wu shows that the Random Forest model outperforms both aforementioned models, highlighting the superior predictive power of the algorithm. Most papers tend to focus on using advanced lagged,

engineered, or technical features when developing such models. Hybrid learning models which combine different algorithms and incorporate technical indicators tend to outperform basic models, especially during volatile periods (Omar et al., 2022). However, there are some studies such as Lubis and Samsudin (2025) that have minimized lagged features to develop the Random Forest model. In this paper, they used basic input features – open, high, low, and volume – to predict the next day opening prices. This paper achieved strong results in the predictive power of the model showing that minimalist Random Forest models can also be used for future day forecasting.

In this research paper, I aim to explore the predictive power of a Random Forest Regression model when predicting the same-day closing price for the MSFT stock using basic input features. These features will be the opening price, high, low, and volume of the stock; these features are chosen due to the fact that they are available throughout the trading day and are easily accessible and interpretable. The focus of this study is to evaluate how far a minimalist model can go in terms of predicting stock price behavior. The purpose of this research is for intra-day trading where the applications of such a model can be implemented to execute trades towards the end of the day based on forecasted predictions. The remainder of this paper is structured as follows. Firstly, I will explain my methodology on how I built my Random Forest Regression model to predict same-day closing prices for MSFT. Following that, I will show my results and provide an analysis on the results of the 10-day predictions and trends that I notice in the data set. Subsequently, I will discuss my findings, exploring the strengths, limitations, implications, and future research potential. Finally, I will end with a conclusion to wrap up the key findings and themes in this study.

Methodology

In this paper, I built a Random Forest model using Python to predict closing prices for the Microsoft Corporation stock (NASDAQ: MSFT). I used 4 years' worth of the Open, High, Low, Close, and Volume (OHLCV) data for MSFT from January 2nd, 2015 (first trading day of 2015) to December 31st, 2018 (last trading day of 2018) to train and test the model. I imported the data from Yahoo Finance into the model. As for the features that were used in the model to predict the closing price, I utilized the opening price of the stock, highest price, lowest price, as well as the volume of stocks. Since I am exploring the intra-day predictive power of the model, I utilized these features since they are easily accessible throughout the trading day, can be monitored, used to predict closing prices at any point in the day, and are simple and easily accessible data. The target variable of the predictions is the same-day closing price, or as the model reads it, the next closing price. Unlike most other time-series forecast models, my target variable was predicted for the same day, not a future-day forecasting. Most other models in this area use features from today with a combination of lagged features to predict features or prices for the next day. This means that the features remain temporally aligned with the target value, allowing the model to analyze and predict intra-day stock price behavior.

After acquiring the data required for the training and testing part of the model, I conducted some preprocessing and cleaning up of the data. I started by dropping any rows with NaN values as a precautionary measure, but this wasn't necessary as the data was already formatted by Yahoo Finance in the following table:

	Date	Open	High	Low	Close	Volume
0	1/2/2015 16:00:00	46.66	47.42	46.54	46.76	27913852
1	1/5/2015 16:00:00	46.39	46.73	46.25	46.33	39673865
2	1/6/2015 16:00:00	46.38	46.75	45.54	45.65	36447854
3	1/7/2015 16:00:00	45.98	46.46	45.49	46.23	29114061
4	1/8/2015 16:00:00	46.75	47.75	46.72	47.59	29645202
...	...	...	...	...	...	...
1001	12/24/2018 13:00:00	97.68	97.97	93.98	94.13	43935192
1002	12/26/2018 16:00:00	95.14	100.69	93.96	100.56	51634793
1003	12/27/2018 16:00:00	99.30	101.19	96.40	101.18	49498509
1004	12/28/2018 16:00:00	102.09	102.41	99.52	100.39	38169312
1005	12/31/2018 16:00:00	101.29	102.40	100.44	101.57	33173765

1006 rows × 6 columns

Table 1: Microsoft Corp. Stock (NASDAQ: MSFT) OHLCV Data in Python

Continuing with the preprocessing, I dropped the "Date" and "Close" columns from my feature matrix as they wouldn't be used to predict the closing price. I also separated the target column "Close" into a different response variable, *y*.

After this, I moved on to training and testing the model. I used an 80/20 random split with 80% of the data used for training and the remaining 20% being used for testing the model. During this stage of the model, another argument was added of "random-state = 42" which would help ensure reproducibility in the model. This way the data points that were used in the training and test sets would stay the same each time the script ran. For training, each of the decision trees are trained on a bootstrap sample. The 20% testing set is only used after the training for the model is complete where it now has a strong understanding of the relationship between the features and target variable. To test the model after training it, I have to fit the training sets to a random forest regression model. For this, I accessed the Scikit Learn library in Python and fitted my training sets to a default Random Forest Regression model. As mentioned in the introduction, Random Forest models have a multitude of hyperparameters to tune, however I only chose to tune the few referenced earlier. The default Random Forest Regressor from the Scikit Learn library sets those parameters as the following: 'n_estimators' = 100, 'max_depth' = none, 'min_samples_split' = 2, and 'min_samples_leaf' = 1. After fitting the set, I can use it to predict for the training set by using the input features from each row of the data set. To measure the efficacy of the model on the training set, I used 2 key metrics: R^2 and Mean Squared Error (MSE). The R^2 value for my training set was approximately 0.99914 while the MSE was approximately 0.38138. Although the R^2 value is quite high and the MSE value remains decently low, I decided to improve the model. To do so, I used the GridSearchCV function from the Scikit Learn library. This function extensively searches over a predefined grid of hyperparameters and their respective values. This practice aligns with current literature where cross-validated hyperparameter tuning is shown to be optimal for a model when predicting on out-of-sample data (Gu et al., 2020). For this model, I chose to search through the following values: 'n_estimators': [100, 200, 300], 'max_depth': [None, 10, 20], 'min_samples_split': [2, 5, 10], and 'min_samples_leaf': [1, 2, 4]. This function also conducts a k-fold cross-validation, 3-fold specifically for my model. This is an important step during hyperparameter tuning as it prevents overfitting of the model and ensures that the model will be able to generalize well. After the search was complete, I obtained the optimal combination of hyperparameters from my grid for the model, which was the following: 'n_estimators' = 300, 'max_depth' = None, 'min_samples_leaf' = 1, and 'min_samples_split' = 2. With the optimal combination of hyperparameters, I re-trained the new model on the original training sets and used it to predict the test set again. Evaluating it on the same metrics, I received slightly better results, with a higher R^2 value of approximately 0.99917 and a lower MSE of 0.37053, an indication that this was definitely a better model than a default Random Forest Regression model from the library.

With this updated model, I then made predictions for the next closing prices. I started by making predictions for the last trading day of 2018 (which was already in the data set) and then inputting the necessary input features to make closing price predictions for the following 9 days. This way I am able to simulate how this model can be used for intra-day trading as the model can be used towards the end of the trading day when the markets settle for the day.

Results and Analysis

After predicting 10 days worth of closing prices by inputting the necessary features, I generated the following table on Google Sheets. Table 2 shows a side-by-side comparison of the predicted closing price of the MSFT stock versus the actual closing price on that day. It also shows some basic statistics with the squared error of each trading day and the absolute error.

Dates	Predicted Closing Price	Real Closing Price	Squared Error	Absolute Error
December 31, 2018	101.39	101.57	0.0324	0.18
January 2, 2019	100.65	101.12	0.2209	0.47
January 3, 2019	98.6	97.4	1.44	1.2
January 4, 2019	100.69	101.93	1.5376	1.24
January 7, 2019	101.57	102.06	0.2401	0.49
January 8, 2019	102.81	102.8	0.0001	0.01
January 9, 2019	104.53	104.27	0.0676	0.26
January 10, 2019	103.49	103.6	0.0121	0.11
January 11, 2019	102.82	102.8	0.0004	0.02
January 14, 2019	101.82	102.05	0.0529	0.23

Table 2: Comparison of predicted vs. actual closing prices of MSFT and error comparison

Based on this table, I also used some other statistics to assess the predictive power of the model. Firstly, the R^2 value is approximately 0.89065. This means that approximately 89.1% of the variance in the closing price of MSFT can be accounted for or predicted by the model and the selected input features. This R^2 value indicates a strong performance by the model when it came to actually predicting closing prices, especially considering that the model used basic trading data with no lagged or derived features. Due to the high volatility of stock markets and the qualitative features that drive variations in stock prices, the R^2 value of this model is quite high. Secondly, another statistic shows that the MSE of this model when predicting was 0.36041. This means that most of the predictions were actually quite close to the actual closing price. When comparing this with our MSE from the training model, it's actually slightly lower by approximately 0.01. MSE significantly penalizes larger errors in the predictions and looking at the table we notice that there are only 2 significant days – January 3 and 4 – that have significantly large squared errors. These 2 days also have the largest absolute errors, making them the outliers in our data set and the worst predictions. Lastly, the mean absolute error (MAE) shows that on average the predictions from the model differ by approximately $0.421 from the actual closing price of the day. Looking at the average closing price throughout the week, this is only about a 0.42% error margin.

From Table 2, it is observable that the model performed best on January 8, 11, and 10 with absolute errors of $0.01 to $0.11. However, the worst performing days of the model were January 3 and 4 with absolute errors above $1.2. This outlying effect in accuracy could be attributed to either the unexpected volatility of markets, or

the large changes in price during this time. These 2 days had the highest change in actual stock prices from the predicted 10 days. Due to this, I then predicted that there is a possible correlation between either the change in actual closing prices (due to external factors and sentiments) and the model error or the % changes in input features and the absolute error. Firstly, I tested out looking at changes in the input features across these 10 days, shown in the table below:

Dates	Open Price	High Price	Low Price	Volume	% Change in Open	% Change in High	% Change in Low	% Change in Volume
December 31, 2018	101.29	102.4	100.44	33173800	—	—	—	—
January 2, 2019	99.55	101.75	98.94	35239300	1.7178	0.6348	1.4934	6.2263
January 3, 2019	100.1	100.19	97.2	42579100	0.5525	1.5332	1.7586	20.8285
January 4, 2019	99.72	102.51	98.93	44060600	0.3796	2.3156	1.7798	3.4794
January 7, 2019	101.64	103.27	100.98	35656100	1.9254	0.7414	2.0722	19.0749
January 8, 2019	103.04	103.97	101.71	31514400	1.3774	0.6778	0.7229	11.6157
January 9, 2019	103.86	104.88	103.24	32280800	0.7958	0.8753	1.5043	2.4319
January 10, 2019	103.22	103.75	102.38	30067600	0.6162	1.0774	0.8330	6.8561
January 11, 2019	103.19	103.44	101.64	28314200	0.0291	0.2988	0.7228	5.8315
January 14, 2019	101.9	102.87	101.26	28437100	1.2501	0.5510	0.3739	0.4341

Table 3: Input Features and Respective Percent Changes for Predicted Days

For January 3, looking at Table 3, the only significant change in the input features is a ~21% change in stock volume, while for January 4, there are only moderate changes in high and low stock prices. Thus, from Table 3, there is no specific correlation to be observed between changes in input features and the significant errors in the model. Thus, I moved on to my second hypothesis – the possibility that the errors could be correlated to changes in the actual closing price due to external non-quantitative market features. To explore this, I created a scatterplot, using the Logger Pro application, of the absolute changes in actual closing prices of MSFT and the absolute error, as seen below:

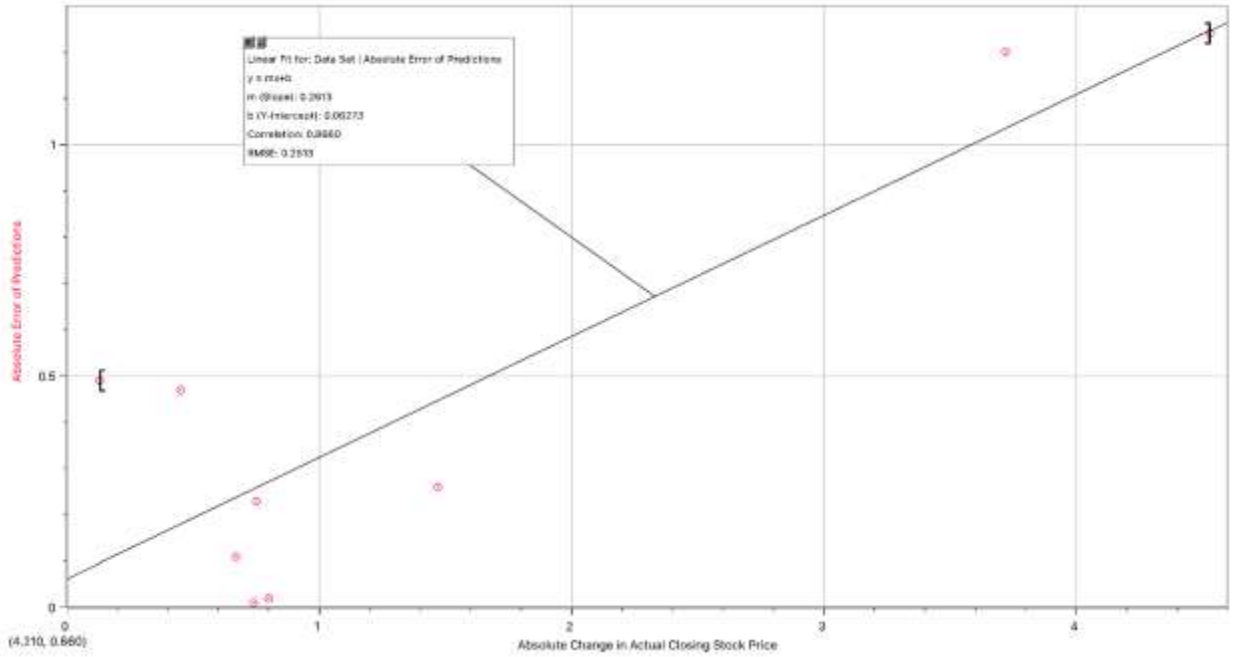

Figure 2: Scatterplot of Absolute Error and Absolute Change in Actual Closing Stock Price

Looking at Figure 2, we notice a moderately strong positive relationship between the absolute change in the actual closing price of MSFT and the absolute error in the predictions of the model. With an R^2 value of approximately 0.866, this indicates a strong positive relationship between the 2 variables. This suggests that a limitation of this model can be that the model performs worse on days where the stock experiences large movements. This can be due to a variety of external market factors, investor sentiment and macroeconomic policies and features that affect the company and/or the industry. This represents the limitations of this basic Random Forest Regression model as stock movement depends on a plethora of features not captured by the basic trading data I used as my input features.

Discussion

The results of this study show that a Random Forest Regression model, when trained with basic intra-day trading features – Open, High, Low, and Volume – can predict, with high accuracy, the closing stock price of MSFT on the same day. This model shows a moderately strong potential for real-world applications, particularly for traders looking to make late-session decisions during the trading day. The key findings of this study show that the basic Random Forest Regression model in my paper was able to predict closing prices for new days with approximately an 0.891 R^2 value and a low MAE of $0.421.

Although this model performed well with intra-day closing price predictions, there are various limitations that need to be addressed. The first limitation is the restricted feature set. In this study, I deliberately used a basic set of features that are readily available throughout the trading day to train the model and preserve simplicity, interpretability, and accessibility. However, the immediate limitation is the lack of exogenous variables. Stock price prediction has been a long-standing problem for traders due to the great number of factors that affect the stock market, the unpredictability, and the volatility. A few of these factors include investor sentiment, geopolitical news, macroeconomic policies, and real-world events. The absence of these factors limits the adaptability and the predictive power of the model. It restricts the model to only being efficient in stable settings. This limitation could be seen in our data set of predictions where the model exhibited poor predictions with significantly higher errors on January 3 and 4. This suggests that real-time features aren't sufficient features to explain stock price movement in all situations. This model intentionally remains simple for the purposes of this study, and thus this limitation cannot be addressed in the scope of this paper, but simply acknowledged. Another

limitation is the fact that the model is designed to predict the same day's closing price and not conduct future forecasting. Due to this feature of the model, it remains minimally generalizable to other fields in finance and limits the applications of this model. Due to no lagged features, the model fails to analyze temporal relationships and patterns. Unlike other models such as ARIMA or LSTM, Random Forest models don't have a memory and can't remember past values unless explicitly fed into the model as features, in the form of lagged features. Due to this limitation, the model can't learn patterns on momentum, volatility, and reversals in the stock price. I've acknowledged this limitation and understand that it hinders the generalizability of the model, however, it is beyond the scope of this study as I intend to simply address the predictive power of the model given real-time features for intra-day trading purposes. Another limitation in my methodology is the limited search of features conducted. To truly find the best possible Random Forest Regression model possible, the GridSearchCV function can be expanded to try countless more possibilities with many of the other hyperparameters as well. I addressed this limitation by only focusing on tuning some of the most relevant hyperparameters that are important to a model like the one made in this study.

Notwithstanding these limitations, this study still has several strengths as well. Firstly, the 4 real-time input features – Open, High, Low, and Volume – are easily accessible, interpretable, and available to traders throughout the trading day. By not including lagged or engineered features in the model, I am able to maintain a high degree of simplicity and interpretability in the model. This deliberate use of minimal and simple input features also mitigates the risk of data leakage. Furthermore, by avoiding these complexities, my model is also less susceptible to overfitting. Secondly, a strength of the methodological design of the model is the temporal alignment between the features and the target variable. Unlike traditional models which shift the target variable to be in the future, the model predicts the closing price for the same day. In this way it prevents any implicit lookahead bias with the data, allowing for more accurate predictions and applications in the current day. In this way, my model also differentiates from traditional time-series models in the literature around this field, and explores a rather niche sub-topic of intra-day trading predictions. Another key strength is the relatively strong out-of-sample performance seen with the high R^2 value from predictions. Moreover, as seen earlier in Table 2, the model wasn't sensitive to the wide distribution of input features and changes in them, showing that the model performed technically strong.

The findings from this study have meaningful implications for both real-world applications and research regarding this field. In real-world scenarios, this model shows that it has a moderately strong predictive power and can be applied in intra-day trading, portfolio execution, and trading system designs. Potentially, models like these could be embedded within trading systems and applications to display real-time forecasted closing prices as the input features continue to change. In this way, the model can help traders maximize their benefits and optimize their trading strategies. Due to the simple input features used, this model can also be easily updated and used for predictions as well as trained on a regular basis to keep improving its accuracy and predictive power. The implications of this model also extend into compliance policies regarding human-AI collaboration in finance. Models like this remain easily explainable and interpretable, making them well-suited for use as financial regulators increasingly move toward explainable AI. This contrasts with many other deep-learning models that may seem accurate, however, can raise red flags over auditability and reproducibility.

In comparison to other literature around this field, this study both builds on and provides new insights on the use of machine learning for predicting stock movement. By avoiding the complexities of lagged features and extensively built models, this study builds on a principle that algorithmic models in finance depend more on temporal structure, predictive objective, and alignment between input features (Gu, Kelly, and Xiu, 2020). In keeping the simplicity of the model, this study differs from a majority of existing literature which focuses on the potential of complex deep learning models such as LSTM and CNNs which are better able to capture trends and

temporal dependencies in stock pricing (Fischer and Krauss, 2018). In this way, this study contrasts other literature by showing how a simple yet well-structured model is also able to deliver a competitive performance with a relatively strong predictive power with much more interpretability and lower operational complexities.

There is a lot of research that has been done in regards to this field, looking at how AI models and other sorts of time-series models can predict stock movement and pricing, and there remains a vast future for this field as well. Future research can go in a variety of directions such as, but not limited to: looking at combining different kinds of models, using more lagged and derived features to increase the predictive power, expanding the model to predict beyond the stock market and into foreign currency or commodities. There has also been progress in research of combining such technical algorithms with other features that affect the markets such as using deep learning models with sentiment analysis or macroeconomic policy and news analysis to determine a more realistic and accurate prediction of stock movement. This is crucial as current limitations in this field are in regards to the lack of ability in incorporating qualitative features and real-world news to predict stock prices, however, further research in this area could revolutionize the way that models are developed and their predictive power.

In conclusion, the evidence presented in this study shows that a basic Random Forest Regression model is able to predict closing prices with a strong accuracy. The model demonstrates that even without the use of advanced and technical features, the predictive power of the model remains strong. The metrics obtained on the 10-day out-of-sample predicting period – an R^2 value of 0.89, MAE of $0.421 and an MSE of 0.38 – suggest that basic input features still provide a strong explanatory power for intra-day trading and price behavior of the MSFT stock. The application of this model can be used by traders for end-of-day trades and making predictions when the input features start to settle towards the end of the trading day. The interpretability of this model makes it accessible and easy to use and provides a basic insight on the predictive power of Random Forest Regression models.

Conclusion

This study investigated the predictive power of a Random Forest Regression model in forecasting the same-day closing price of Microsoft Corporation stock (NASDAQ: MSFT). I used basic, real-time input features that are available throughout the day to traders, namely the opening price, high, low, and volume. Using the model built, across a 10-day prediction period, it achieved an R^2 value of approximately 0.89, a low MSE and MAE of approximately 0.36 and 0.421, respectively. This paper indicates that Random Forest Regression models are able to achieve strong results and indicate a high predictive power even when trained on minimal, non-lagged, and non-technical features. However, this study also found some of the limitations of this model. The most important limitation of not only Random Forest models, but machine learning algorithms in this field as a whole, is the inability to incorporate exogenous factors that affect stock markets such as, but not limited to investor sentiment, geopolitical events, and macroeconomics policies. Another limitation of this study is the use of simple input features and no lagged or derived features. Nonetheless, this limitation was accepted as the scope of this paper is to explore a minimalist Random Forest Regression model. Ultimately, this study contributes to the increasing literature around intra-day trading and shows a strong predictive power of a basic Random Forest Regression model in predicting the same-day closing price.

References

Baladram, Samy. "Random Forest, Explained: A Visual Guide with Code Examples." *Medium*, TDS Archive, 30 Nov. 2024, medium.com/data-science/random-forest-explained-a-visual-guide-with-code-examples-9f736a6e1b3c.

Breiman, Leo. "Random forests." *Machine Learning*, vol. 45, no. 1, Oct. 2001, pp. 5–32, https://doi.org/10.1023/a:1010933404324.

Fraj, Mohtadi Ben. "In Depth: Parameter Tuning for Random Forest." *Medium*, All things AI, 21 Dec. 2017, medium.com/all-things-ai/in-depth-parameter-tuning-for-random-forest-d67bb7e920d.

Gane, Muhammad Turner. "Next Day Stock Price Prediction of the S&P 500 Using Random Forest." *Medium*, Medium, 8 Sept. 2024, medium.com/@m.turnergane/next-day-stock-price-prediction-of-the-s-p-500-using-random-forest-f6b5bca0e55f.

Gu, Shihao, et al. "Empirical asset pricing via machine learning." *The Review of Financial Studies*, vol. 33, no. 5, 26 Feb. 2020, pp. 2223–2273, https://doi.org/10.1093/rfs/hhaa009.

IBM. "What Is Random Forest?" *IBM*, 4 June 2025, www.ibm.com/think/topics/random-forest#:~:text=Random%20forest%20is%20a%20commonly,both%20classification%20and%20regression%20problems.

Krauss, Christopher, et al. "Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500." *European Journal of Operational Research*, vol. 259, no. 2, June 2017, pp. 689–702, https://doi.org/10.1016/j.ejor.2016.10.031.

Lubis, Muhammad Amsari, and Samsudin Samsudin. "Using the random forest method in predicting stock price movements." *Journal of Dinda : Data Science, Information Technology, and Data Analytics*, vol. 5, no. 1, 7 Feb. 2025, pp. 28–35, https://doi.org/10.20895/dinda.v5i1.1765.

Omar, Abdullah Bin, et al. "Stock market forecasting using the random forest and deep neural network models before and during the COVID-19 period." *Frontiers in Environmental Science*, vol. 10, 25 July 2022, https://doi.org/10.3389/fenvs.2022.917047.

Probst, Philipp, et al. "Hyperparameters and tuning strategies for Random Forest." *WIREs Data Mining and Knowledge Discovery*, vol. 9, no. 3, 28 Jan. 2019, https://doi.org/10.1002/widm.1301.

Wu, Yifan. "Stock price prediction based on simple decision tree random forest and xgboost." *BCP Business & Management*, vol. 38, 2 Mar. 2023, pp. 3383–3388, https://doi.org/10.54691/bcpbm.v38i.4311.