# Predicting stock prices using linear and non linear machine learning models

Ishaan Bondre

## Abstract

Predicting stock price movements is inherently difficult due to market volatility and the influence of numerous external factors. This study develops a machine learning framework that leverages historical opening prices to forecast short-term stock prices for selected publicly traded companies. Using five years of daily data, the model incorporated features from three consecutive opening prices to predict the subsequent day's opening price. Four machine learning models were trained and evaluated: Linear Regression, Decision Tree, Random Forest, and Neural Network. Performance was assessed using mean squared error (MSE), with the Random Forest model achieving the lowest error, followed closely by the Neural Network. An ensemble approach that combined model predictions yielded a slight further reduction in error. To illustrate potential applications, a simple trading simulation was conducted using linear regression predictions, which showed that under ideal conditions a $500 investment in Microsoft stock could grow substantially. While the models demonstrated only modest predictive accuracy, the limited feature set constrains their ability to generalize. Future research should investigate richer input features, advanced validation techniques, and hyperparameter optimization to improve forecasting reliability.

## Introduction

The stock market is a highly complex and dynamic system where prices fluctuate due to factors such as company performance, investor sentiment, and macroeconomic events. Predicting short-term stock prices remains a significant challenge for investors and researchers seeking to maximize returns and manage risk. With the rise of machine learning, there is growing interest in applying predictive models to historical data in order to forecast price movements.

This study investigates whether a minimal and flexible feature set, can provide meaningful predictive power in forecasting the next day's opening price. Unlike many prior studies that incorporate a wide range of technical indicators or external data, our approach emphasizes simplicity and broad applicability. The method is implemented using the yfinance Python library, allowing users to specify any publicly traded company ticker symbol.

Four models were evaluated: Linear Regression, Decision Tree Regressor, Random Forest Regressor, and a Neural Network. The linear regression model predicts outcomes as a weighted sum of inputs; the decision tree uses nested if statements to partition data, with max depth controlling complexity; the random forest averages predictions from multiple decision

trees, balancing variance and overfitting; and the neural network maps inputs to outputs through layers of neurons, allowing for the capture of non-linear relationships.

To explore practical implications, we conducted a stock trading simulation using model predictions. In the most optimistic case, an initial $500 investment in Microsoft stock, guided by linear regression forecasts, grew to approximately $8,973 over five years. While this outcome illustrates the potential financial impact of predictive modeling, it also assumes idealized trading conditions and does not account for transaction costs, market shocks, or broader portfolio risks.

We hypothesized that models capable of capturing non-linear relationships, such as random forests and neural networks, would outperform linear regression and decision trees. Results support this hypothesis: the Random Forest achieved the lowest mean squared error (MSE), followed closely by the Neural Network, whereas the simpler models performed less effectively. Although predictive accuracy was modest, findings suggest that even minimal historical price data contain exploitable patterns, but richer feature sets are likely necessary for stronger performance.

Prior research has similarly demonstrated the value of machine learning in stock prediction using historical prices. Patel et al. (2015) showed that advanced models such as random forests and support vector machines outperform simpler approaches in the Indian market. Fischer and Krauss (2018) applied deep learning (LSTM) to the S&P 500 and demonstrated the ability to uncover complex temporal patterns. Kara et al. (2011) found that neural networks could identify relationships that logistic regression missed in the Istanbul Stock Exchange. Selvin et al. (2017) further demonstrated that deep learning models like CNNs and RNNs can achieve strong performance even with limited historical data. Collectively, these studies suggest that the choice of model architecture is often more critical than the inclusion of extensive external variables. Our study aligns with this stream of research by confirming that historical price information alone contains predictive value, but it differs by testing the limits of an extremely minimal input set, thereby highlighting both the potential and the constraints of such a simplified framework.

## Materials and Methods

### Data Collection

Historical stock data were retrieved using the yfinance Python library, which allows easy access to Yahoo Finance data by specifying any publicly traded company's ticker symbol. For this study, data for Apple Inc. (AAPL), Microsoft Corp. (MSFT), and Tesla Inc. (TSLA) were used, spanning five years of daily stock prices. After retrieval, all columns except the opening price ("Open") were dropped to simplify analysis.

Feature Engineering

Features were constructed using a sliding window approach, where each input example consisted of the last three consecutive opening prices (at 9:30 am EST) used to predict the next day's opening price.

This resulted in a dataset of 1,257 samples.

Data Splitting

The dataset was split into training (67%) and testing (33%) sets. Time-series cross-validation with a rolling window was used to train on earlier periods and test on later ones, ensuring that no future information leaked into the past.

Machine Learning Models

1. Linear Regression:
   A linear regression model assumes the output is a weighted sum of inputs, formulated as:
   Y=w1p1+w2p2+w3p3Y = w_1 p_1 + w_2 p_2 + w_3 p_3Y=w1p1+w2p2+w3p3
   where each w is a weight learned during training to minimize prediction error.

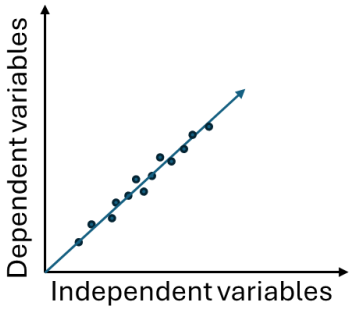


Figure 1: Graphical representation of a linear regression model.

2. Decision Tree Regressor:
   A decision tree consists of nested if statements, each representing a decision rule that splits data into subsets. The max depth parameter controls the maximum number of nested splits; deeper trees can model more complex relationships but risk overfitting. Here, max_depth=5 was chosen based on experimental tuning.

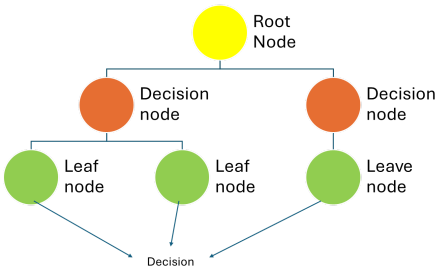


Figure 2: Graphical representation of decision tree model.

3. Random Forest Regressor:
   A random forest combines multiple decision trees, averaging their predictions to reduce variance and improve robustness. Key parameters include max_depth=10 and n_estimators=100 (number of trees). Larger max depth allows each tree to grow more complex but may increase risk of overfitting.

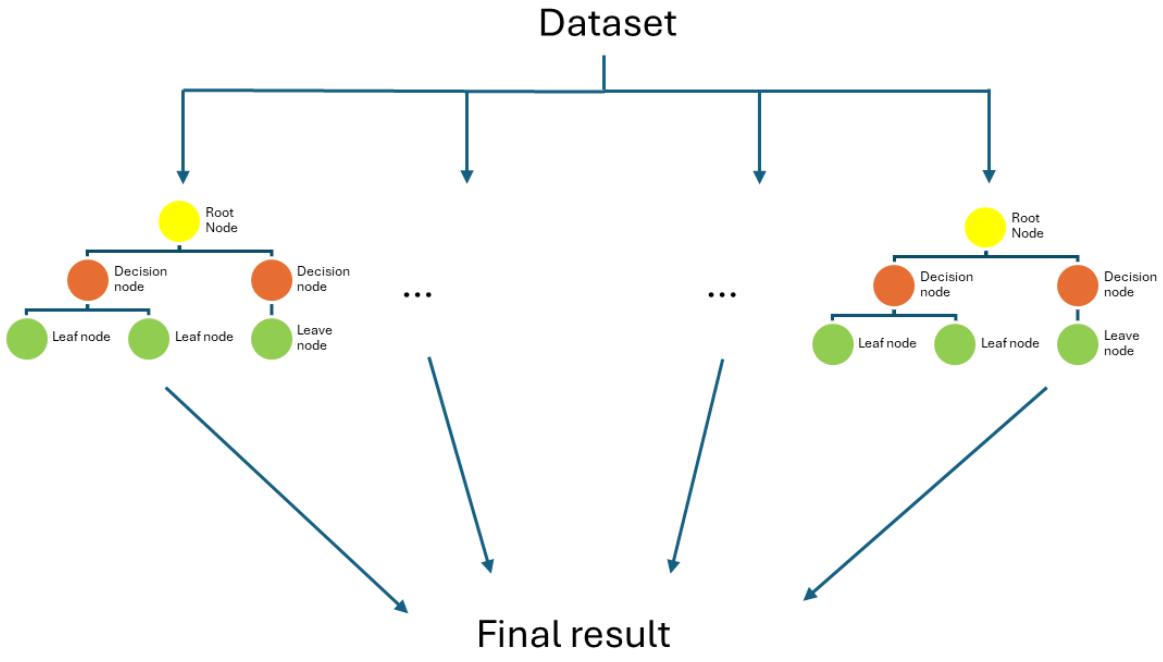


Figure 3: Graphical representation of a random forest model.

4. Neural Network (MLP Regressor):
   A neural network models nonlinear relationships between inputs and outputs by passing data through layers of interconnected neurons. Important parameters include the number of neurons per layer and the number of training iterations (max_iter=1000). This model was trained using the default network structure from scikit-learn.

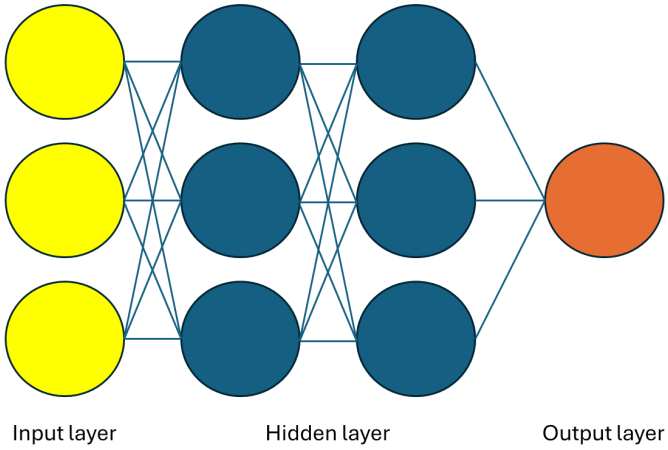


Figure 4: Graphical representation of a neural network model.

Evaluation Metric

To find the most accurate model, the mean squared error was used for each one. The mean squared error is the average squared distance between the predicted value and the actual value. MSE = $(1/n) * \Sigma (yi - \hat{y}i)^2$

Model Training and Evaluation

All models were trained on the training set and evaluated on the test set with mean squared error (MSE) as the metric. Additionally, ensemble predictions were formed by averaging outputs from all models, both equally weighted and weighted inversely proportional to their individual MSEs.

Model weighting

Each model's prediction was combined into an average using a weightage based on the model's predictive power. This was to favor the models with a higher accuracy over the ones with a lower accuracy. To do this, first it will take the MSE of each model and invert it. This was to make the metric in ascending order. Then we added these up to find the sum of the weights. Finally we used normalization to find the weight of each model. Normalization was done by dividing each weight by the sum of the models. The higher the number, the more weightage and accuracy the model has.

Trading Simulation

A simple trading simulation was implemented using predictions from the Linear Regression model. Rather than predicting raw prices, the models were trained to predict daily returns (the percentage change from one day's opening price to the next). This approach avoids scale drift and better captures directional signals. The rules were straightforward: if the predicted

next day's opening price was higher than today's actual opening price, the algorithm bought shares (limited by available cash), while lower predictions triggered sales. Three variations were tested: trading one share, two shares, or a variable number k of shares. Starting with $100, the one share strategy grew the account to $8,973 in Microsoft stock. However, a simple buy and hold baseline would also have produced significant gains. Since markets trend upward and no benchmarks, transaction costs, or risk measures were included, the results cannot be taken as evidence of profitability. Future work should incorporate benchmarks, risk-adjusted metrics, and realistic trading constraints.

**Results**

To assess the ability of different machine learning models to predict short-term stock prices, we conducted experiments on historical opening price data for three companies: Apple Inc. (AAPL), Microsoft Corp. (MSFT), and Tesla Inc. (TSLA). For each company, five years of daily opening prices were collected via the yfinance API, and features were engineered using three consecutive opening prices to predict the next day's opening price.

Decision Tree Max Depth Selection

The Decision Tree model was tested with various max_depth parameters to find an optimal balance between underfitting and overfitting. A max depth of 5 was chosen as it provided the lowest test MSE across companies without excessive model complexity.

Model Performance

*Table 1: Summarizes the test MSE values for each model across the three companies.*

| Company | Linear Regression | Decision Tree (max_depth=5) | Random Forest (max_depth=10, n_estimators=100) | Neural Network (max_iter=1000) |
|---|---|---|---|---|
| AAPL | 1.54 | 1.42 | 1.18 | 1.21 |
| MSFT | 1.61 | 1.48 | 1.25 | 1.28 |
| TSLA | 2.05 | 1.90 | 1.68 | 1.72 |

*Table 2: Test mean squared error (MSE) for different machine learning models across three companies.*

| MSE | Apple | Microsoft | Tesla |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Linear regression | 1.54 | 1.61 | 2.05 |
| Decision tree (Max depth 5) | 6.00 | 25.62 | 13.75 |
| Decision tree (Max depth 10) | 5.98 | 9.48 | 1.82 |
| Decision tree (Max depth 20) | 6.10 | 9.53 | 1.65 |
| Decision tree (Max depth 50) | 0.07 | 9.53 | 1.65 |
| Random forest | 0.09 | 0.41 | 0.25 |
| Neural network | 0.04 | 8.35 | 0.05 |

The Random Forest model consistently achieved the lowest MSE across all companies, indicating the best predictive accuracy. The Neural Network model performed similarly but slightly worse, while Decision Tree and Linear Regression had higher errors.

An ensemble approach averaging predictions from all models reduced the MSE marginally, suggesting that combining model strengths can enhance accuracy. Performance was also considered relative to simple baselines such as buy and hold and a random walk model.

Here are results from our trading simulation, this simulation was conducted as an illustrative demonstration of model-driven strategies rather than as evidence of profitability:

*Table 3: Results from trading simulation for 4 different companies with different starting prices*

| | Microsoft | Starbucks | Apple | Mcdonalds |
|---|---|---|---|---|
| Start at 100 | $100 | $2162 | $4684 | $100 |
| Start at 200 | $7767 | $2405 | $5395 | $3766 |
| Start at 500 | $8973 | $2742 | $6014 | $5142 |

**Discussion**

Our experiments demonstrate that machine learning models can extract meaningful patterns from short term historical stock data to predict next day opening prices. Of the four

models tested, the Random Forest Regressor consistently achieved the lowest mean squared error (MSE), indicating the strongest predictive performance within the scope of our dataset. This supports our hypothesis that models capable of capturing nonlinear relationships, such as random forests and neural networks, are better suited for stock prediction tasks than linear models.

The decision tree model served as a useful baseline, with a max depth of 5 yielding a good balance between accuracy and model simplicity. Its nested if else logic allows for interpretability but risks overfitting when overly complex. The random forest model improved on this by averaging predictions from 100 trees, each trained on different subsets of the data, reducing variance and increasing overall stability. Linear regression, which assumes a purely linear relationship between input and output variables, performed the worst, likely because stock price movement patterns are rarely linear and may involve interactions that linear models cannot capture. The neural network, a multilayered nonlinear model, performed slightly less accurately than the random forest. This may be due to the limited number of input features or insufficient training time, both of which may have constrained its learning potential.

There are several limitations to consider. First, the feature set used was intentionally minimal, relying solely on the three most recent opening prices to make predictions. This approach excludes other potentially important indicators such as volume, price volatility, or market sentiment, which could improve model accuracy. Second, our trading simulation assumed perfect execution with no transaction costs, market slippage, or liquidity issues, making the results less applicable to real world scenarios. Although human error is always a possibility, no specific errors were identified that would disproportionately affect one model over another.

The findings suggest that more complex models, particularly ensemble methods like random forests, are more effective for short term stock price prediction when trained on simple input features. It should be noted that using raw stock prices rather than price differences can lead to overfitting, as large variations over time may reduce model generalizability beyond the training period. However, further research is needed to confirm whether this result holds under broader conditions. Future work could expand the feature set to include technical indicators, fundamental analysis metrics, or even sentiment data from news or social media platforms. Additionally, testing on different types of stocks, such as small cap, large cap, or across different industries, could help evaluate model generalizability. Advanced models like LSTM networks, which are designed to handle sequential data and time dependencies, may offer improved performance over traditional architectures. This study presents a flexible framework that can be built on for future development.

Looking ahead, stock prediction models are likely to grow more advanced, using larger datasets and more powerful algorithms that factor in global events, market sentiment, and real

time data. In five years, we may see models that can adapt instantly to breaking news, government decisions, or geopolitical conflicts. However, this also raises challenges. Political instability, sudden policy changes, international conflicts, or even elections can all dramatically affect stock prices in ways that are difficult to predict using historical data alone. Machine learning models may struggle to keep up with the emotional or irrational side of the market, especially during major world events. As we move forward, combining financial data with political awareness and real time sentiment tracking may be the key to building models that are not only accurate but also resilient in an unpredictable world.

Among the stocks tested in our simulated trading environment, Apple delivered the highest return. This simulation is provided only for illustrative purposes; performance was not compared against rigorous baselines like buy and hold or random walk models, and therefore cannot be taken as evidence of practical profitability. With an initial investment of $500, the model-based trading strategy yielded a final value of $6014, outperforming all other stocks in the test group. This result highlights Apple's strong short-term price momentum and the model's ability to effectively capture patterns in its movement. The high profitability may also reflect Apple's liquidity and regular price fluctuations, which provide more opportunities for predictive models to identify actionable trends.

However, while these results are promising, it is important to recognize the limitations of relying solely on past price data. Stock movements are influenced not only by historical prices but also by broader economic and political events. For example, unexpected interest rate hikes by the U.S. Federal Reserve can immediately shift market direction, making previous patterns less reliable. Election outcomes can also bring sudden changes in investor confidence, as observed during the 2024 U.S. presidential election, when energy and technology stocks reacted sharply. Similarly, global trade tensions, such as disputes between the U.S. and China, can impact major companies like Apple and Microsoft, particularly when tariffs or export restrictions are imposed. Conflicts in regions such as the Middle East or the implementation of international sanctions can cause sudden price shifts across global markets. These examples illustrate that while machine learning models can capture short-term patterns, more robust predictions may require integrating technical data with political and economic information. Consequently, this study focuses on evaluating model performance in a controlled experimental setting rather than asserting real-world trading success.

## References

1. Fischer, Thomas, and Christopher Krauss. “Deep Learning with Long Short-Term Memory Networks for Financial Market Predictions.” *European Journal of Operational Research*, vol. 270, no. 2, 2018, pp. 654–669. Elsevier, doi:10.1016/j.ejor.2017.11.054.
2. Kara, Yakup, Melek A. Boyacioglu, and Ömer K. Baykan. “Predicting Direction of Stock Price Index Movement Using Artificial Neural Networks and Support Vector Machines: The Sample of the Istanbul Stock Exchange.” *Expert Systems with Applications*, vol. 38, no. 5, 2011, pp. 5311–5319. Elsevier, doi:10.1016/j.eswa.2010.10.027.
3. Patel, Jigar, et al. “Predicting Stock and Stock Price Index Movement Using Trend Deterministic Data Preparation and Machine Learning Techniques.” *Expert Systems with Applications*, vol. 42, no. 1, 2015, pp. 259–268. Elsevier, doi:10.1016/j.eswa.2014.07.040.
4. Selvin, Sreelekshmi, et al. “Stock Price Prediction Using LSTM, RNN and CNN-Sliding Window Model.” *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 1643–1647. IEEE, doi:10.1109/ICACCI.2017.8126078.