A 15-Year Empirical Comparison of Deep Learning, Tree-Based, and Regression Models for Stock Market Forecasting

Samyak Jayanth

Abstract

This study conducts a comprehensive comparison of traditional machine learning, deep learning, and naïve baseline models for daily stock close predictions, using twelve publicly traded equities from various sectors within the S&P 500 index, including Tesla ($TSLA), JPMorgan Chase ($JPM), NVIDIA ($NVDA), UnitedHealth Group ($UNH), Alphabet ($GOOGL), General Electric ($GE), The Coca-Cola Company ($KO), ExxonMobil ($XOM), Duke Energy ($DUK), American Tower ($AMT), and Linde plc ($LIN) and the SPDR S&P 500 ETF Trust ($SPY). The SPDR S&P 500 ETF Trust - hereafter referred to by SPY - is used as a highly liquid proxy for the S&P 500 index, closely mirroring its price movements. Five models, Linear Regression, Lasso Regression, Random Forest, XGBoost and Long Short-Term Memory (LSTM) neural networks, are trained on identical historical market data with engineered technical features. These are then evaluated against common baselines, including previous close, previous open, and midpoint estimators. The predictive accuracy of each model is assessed using the average Mean Squared Error (MSE) and directional accuracy of each model on all 12 equities, with additional analysis via permutation feature importance to determine key drivers of model performance. Results reveal that, despite the complexity of advanced models, like the LSTM and XGBoost, simple linear and decision-tree based approaches often outperform deep learning in this domain, while all models struggle to consistently surpass basic baselines due to the efficient and noisy nature of financial markets. This research highlights the practical strengths and limitations of various modeling approaches for financial time series, offering guidance for future research and practitioners interested in predictive modeling for asset prices.

Introduction

Financial markets are notoriously challenging to forecast or predict, characterized by their complex dynamics, inherent noise, and the persistent quest for an elusive predictive edge. In such an environment, incremental improvements in predictive accuracy (even as little as 0.01%) are considered exceptionally valuable. Over the past decade, the increasing availability of high-frequency market data and advancements in machine learning (ML) have sparked a surge of interest in using data-driven models to predict asset prices. Despite this progress, the fundamental efficiency of financial markets, shaped by rapid information diffusion and the behavior of more than 50 million market participants, poses a formidable barrier for predictive modeling.

Within this landscape, researchers and practitioners have turned to a wide range of various modeling techniques, from classical linear regression to sophisticated deep learning architectures such as Long Short-Term Memory (LSTM) networks. Meanwhile, traditional benchmarks, such as the previous day's closing price or rolling averages, continue to demonstrate remarkable resilience, often proving difficult for even advanced models to consistently outperform. The prevailing challenge lies in distinguishing meaningful signal from noise, and in evaluating whether increased model complexity truly yields superior predictive power in the context of daily stock prices.

This paper seeks to systematically compare the predictive accuracy of five distinct modeling approaches – Linear Regression, Lasso Regression, Random Forest, XGBoost, and LSTM – on the task of one-day-ahead close predictions for the 12 stocks. All models are trained and tested on identical market data with an enriched set of technical and engineered features, allowing for a fair and rigorous assessment of their respective strengths and limitations. By benchmarking each model against several naïve baselines, including previous close, previous open and midpoint predictors, this research aims to illuminate the real-world value added by machine learning in financial forecasting.

The importance of this research can be articulated in two key aspects. First, it provides empirical clarity regarding the practical performance of widely used ML models compared to financial heuristics, contributing to a more nuanced understanding of model selection in quantitative finance. Second, by evaluating feature importance alongside directional accuracy, the research offers insights into the factors that drive predictive success or failure in noisy and adaptive market environments. The findings are intended to guide both academic researchers and market practitioners in navigating the trade-offs between model sophistication, interpretability, and robustness in financial price prediction.

Literature Review

Machine learning and deep learning approaches have received significant attention for stock price prediction, yet much of the literature reveals both the promise and practical limitations of these techniques. This review synthesizes findings from several foundational studies:

· ***Deep Learning with Long Short-Term Memory Networks for Financial Market Predictions*** **(Fischer & Krauss, 2018),**

· ***Deep Neural Networks, Gradient-Boosted Trees, Random Forests: Statistical Arbitrage on the S&P 500*** **(Krauss et al., 2017), and**

· ***The Probability of Backtest Overfitting*** **(Bailey et al., 2015)**

Fischer & Krauss (2018) apply LSTM networks to daily stock price forecasting, arguing that deep sequential models can capture complex temporal dependencies in financial time series. They report modest improvements over random guessing and linear models in certain market regimes, but the gains are limited and often sensitive to model tuning and sample period. Importantly, their study focuses on German equities and does not directly compare LSTM with other modern machine learning algorithms, nor does it address challenges of feature engineering or baseline selection in detail. Our study builds on Fischer & Krauss by benchmarking LSTM directly against a variety of classical and modern ML models, and by incorporating a broader set of engineered features and robust baseline comparisons on US equities.

Krauss et al. (2017) conduct one of the largest empirical studies comparing deep neural networks, random forests, and gradient boosted machines for statistical arbitrage on the S&P 500. Their results indicate that tree-based ensemble models such as random forests and XGBoost often match or outperform deep neural networks in terms of risk-adjusted returns and prediction accuracy, especially on tabular market data. However, their analysis does not systematically compare model results to strong naïve baselines (such as previous close or open), and the study is conducted primarily on minute-level intraday data rather than daily bars. Furthermore, they do not explicitly analyze directional accuracy or feature importance. Our work extends their comparative approach to daily price prediction, incorporates permutation feature importance, and emphasizes rigorous out-of-sample benchmarking against naïve baselines.

Bailey et al. (2014) provides a cautionary perspective on financial forecasting, demonstrating that many published results in the field can be attributed to backtest overfitting rather than genuine predictive skill. They argue that naïve baselines are surprisingly difficult to beat in efficient markets (something that our research has proven to support) and they advocate for strong statistical controls and robust model validation. While their work is primarily theoretical and focuses on statistical methodology, it motivates our own emphasis on comprehensive baseline comparisons, careful out-of-sample validation, and transparency in model evaluation.

Other recent literature highlights the importance of feature engineering in financial machine learning. Ghaffari & Rahmani (2020) review common feature selection and extraction methods, finding that while technical indicators such as moving averages and ATR can sometimes improve predictive performance, their impact is inconsistent and often modest. Moreover, few studies systematically assess which features actually matter for different model types, or compare feature importance across models. Our research addresses this gap by evaluating permutation feature importance for each model and exploring the marginal value of additional engineered features.

Collectively, these works underscore the challenges of applying machine learning to financial prediction. While many methods show potential, persistent noise, market efficiency, and

overfitting limit the ability of complex models to consistently outperform simple heuristics. This paper addresses these open questions by systematically comparing the predictive accuracy and feature importance of LSTM, linear regression, lasso regression, random forest, and XGBoost models on identical data and robust baselines, using Tesla as a case study. In doing so, we provide new insight into the relative strengths and limitations of popular modeling approaches for real-world financial time series forecasting.

Overview of Predictive Models

Linear Regression

Linear regression is one of the most fundamental and widely used techniques in statistical modeling and machine learning. It assumes a linear relationship between a set of input features and the target variable, modeling this relationship by fitting a straight line that attempts to minimize the mean squared error between predicted and actual values. Owing to its simplicity and interpretability, linear regression serves as a common baseline in predictive modeling tasks, including financial time series forecasting. Despite its strengths, linear regression is inherently limited to capture only linear dependencies and tends to struggle with multicollinearity or exponentiality among features. In the context of financial markets, where relationships between variables are often complex and nonlinear, this simplicity can hinder predictive performance. Nevertheless, linear regression remains valuable for its transparency, ease of implementation, and utility in highlighting fundamental trends within data.

Lasso Regression

Lasso regression, or Least Absolute Shrinkage and Selection Operator, is a regularized linear modeling technique designed to enhance both predictive accuracy and interpretability. By introducing an L1 penalty to term to the loss function, Lasso performs automatic feature selection by shrinking some coefficients entirely to zero, effectively excluding less informative predictors from the model. This property makes Lasso particularly valuable in high-dimensional datasets, where irrelevant or redundant features can degrade model performance. In financial time series forecasting, Lasso can help identify which technical indicators and engineered features are most relevant for price prediction while mitigating overfitting. However, like standard linear regression, Lasso is still limited to modeling linear relationships between features and the target. Despite this, its ability to produce sparse, interpretable models makes it a practical and insightful choice for financial data analysis.

Random Forest

Random Forest is an ensemble learning algorithm that builds a large number of decision trees and aggregates their predictions to enhance accuracy and reduce overfitting. Each tree in the forest is trained on a random subset of the data and considers a random subset of features at

each split, which helps to decorrelate the trees and improve generalization. In financial time series forecasting, Random Forest models are capable of capturing complex, nonlinear relationships and interactions among features that linear models might miss. They are relatively robust to noisy data and can handle a wide variety of feature types without requiring extensive preprocessing. However, while Random Forests provide variable importance measures, their ensemble nature can make them less interpretable than simple linear models. Overall, Random Forest is valued for its flexibility, predictive power, and effectiveness across a broad range of data science problems, including financial market prediction.

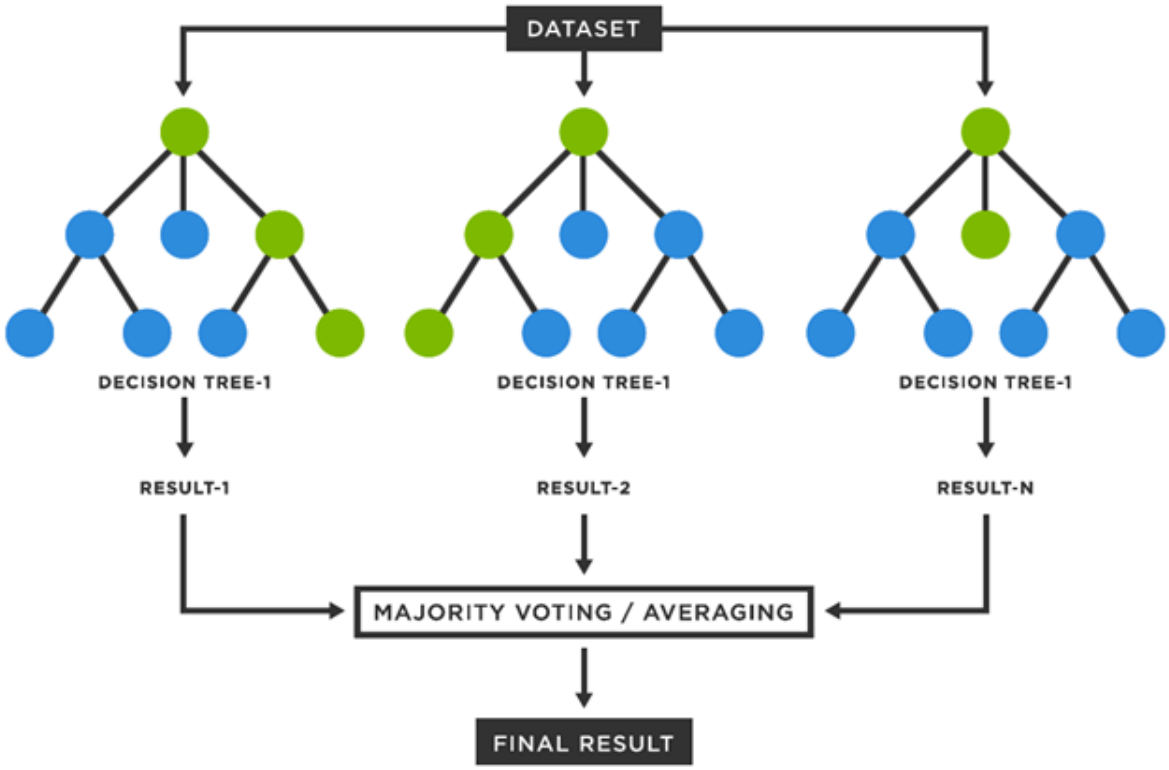

Figure 1: Example of a Random Forest decision tree (Gunay, 2023)

XGBoost

XGBoost (Extreme Gradient Boosting) is a powerful and highly efficient ensemble learning algorithm based on gradient-boosted decision trees. Unlike Random Forest, which builds trees independently and averages their outputs, XGBoost constructs trees sequentially, with each new tree learning to correct the errors of the previous ensemble. Its use of gradient boosting allows for the modeling of complex nonlinear relationships and interactions among features, making it exceptionally effective on structured tabular data. XGBoost incorporates advanced

regularization techniques, such as L1 and L2 penalties, which help prevent overfitting – a common challenge in financial modeling. The algorithm is also optimized for speed and scalability, enabling the training of large models on extensive datasets. While XGBoost can be less interpretable than simple models, it is widely regarded for its predictive accuracy and is a top performer in many data science and machine learning competitions, including applications in financial time series prediction.

LSTM (Long Short-Term Memory Networks)

Long Short-Term Memory (LSTM) networks are a specialized type of recurrent neural network (RNN) designed to capture and learn from sequential dependencies in time series data. Unlike traditional feedforward models, LSTMs maintain an internal memory state, allowing them to effectively model long-range temporal relationships and mitigate the vanishing gradient problem common in standard RNNs. This makes them particularly well-suited for tasks involving financial time series, where patterns and dependencies may span multiple days or weeks. LSTMs are highly flexible and can approximate complex, nonlinear relationships between past market indicators and future price movements. However, they require significant computational resources and large amounts of data to train effectively, as they are prone to overfitting when data is limited. Despite these challenges, LSTM models remain a popular choice in financial forecasting research due to their ability to extract hidden temporal patterns that traditional models may overlook.

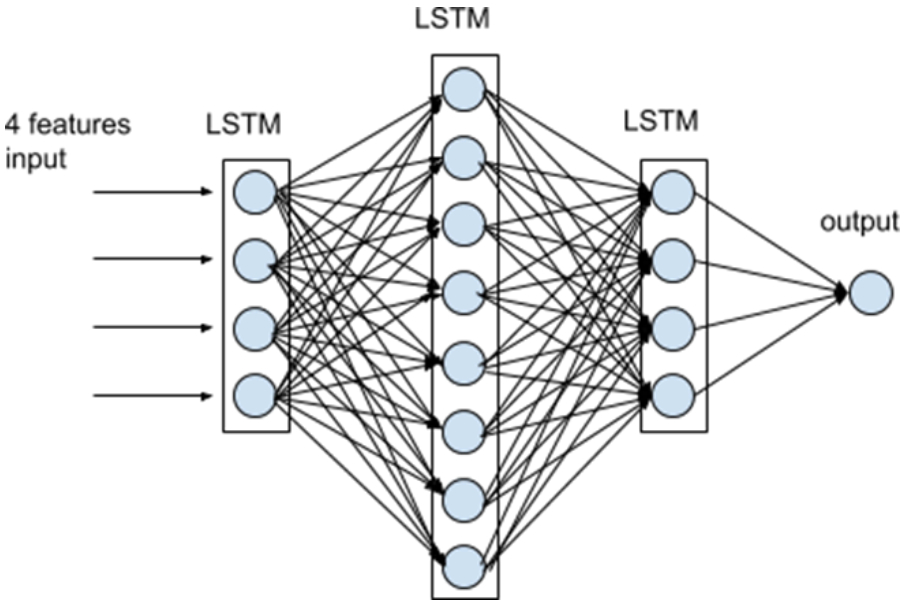

Figure 2: Example of a Long Short-Term Memory Network

(https://stackoverflow.com/questions/52647115/python-implementing-an-lstm-network-with-keras-and-tensorflow)

Methodology (Data Collection)

This study collected historical daily price data for Tesla, JPMorgan, Nvidia, UnitedHealth Group, Alphabet, General Electric, Coca Cola, ExxonMobil, Duke Energy, American Tower, Linde, and the S&P 500 using the Twele Data API. The dataset includes open, high, low, close, and volume (OHLCV) values for each trading day, spanning the period from June 29th, 2010, to June 25th, 2025. The time frame was selected to encompass both pre-pandemic, pandemic, and post-pandemic market conditions, ensuring that the models are evaluated across varying regimes of market volatility and investor behavior.

Supplementary features were engineered based on the raw OHLCV data, including moving averages (MA5, MA10), relative strength index (RSI14), returns, rolling volume averages, price ranges, retail support and resistance levels, and multiple average true range (ATR) metrics (ATR5, ATR14). These features were chosen for their documented relevance in financial forecasting literature and their potential to capture underlying patterns in price dynamics.

All data were cleaned to remove missing values and outliers prior to modeling. Technical indicators and lagged features were computed using rolling window operations, with the initial rows containing insufficient lookback periods dropped to ensure consistency. The final dataset was split into training (80%) and testing (20%) in chronological order to preserve the integrity of the time series.

Methodology (Data Preprocessing)

All price and volume data were first cleaned by removing missing values and sorting records chronologically. Key technical indicators, including moving averages, RSI, ATR (Average True Range), and support/resistance key levels, were computed using rolling windows and merged into the dataset as additional features. Any rows with insufficient historical data for these rolling features were dropped to ensure data consistency. All feature columns were standardized using z-score normalization, with scaling parameters fit only on the training data. The resulting dataset was then segmented into rolling input windows of 15 trading days; each paired with the next day's price values for supervised learning. The 15-day rolling input window was chosen to capture both short- and medium-term patterns in market behavior, balancing responsiveness to recent price action with the smoothing benefits of longer-term trends.

Methodology (Model Construction)

To assess the predictive power of various machine learning techniques in forecasting next-day equity price movements, this study implements five distinct model architectures: Linear Regression, Lasso Regression, Random Forest, XGBoost, and Long Short-Term Memory neural networks. Each model was constructed using widely adopted Python libraries – skicit-learn for linear, lasso, and random forest regressions; XGBoost for gradient boosting; and PyTorch for the

LSTM network. The input features for all models were derived from a rolling window of 15 consecutive trading days, with each sample containing technical indicators such as moving averages, RSI, ATR, and lagged price/volume metrics. The models were trained to predict the next trading day's open, high, low, and close prices. An 80/20 chronological split was applied to partition the data into training and testing sets, preserving the temporal order of the financial time series and preventing lookahead bias. For the LSTM, additional hyperparameters such as the number of layers, hidden units, optimizer choice, batch size, and number of epochs were set according to best practices for time series forecasting. All models were trained and evaluated on the same dataset and feature set, enabling direct, unbiased comparisons of predictive accuracy and error metrics.

Methodology (Training and Validation)

All predictive models were trained exclusively on the training portion of the dataset, corresponding to 80% of the chronologically earliest data, while the remaining 20% served as the out-of-sample test set for validation. This split preserved the temporal integrity of the financial time series, preventing lookahead bias and simulating a true forecasting scenario. Standard scaling was applied to both features and targets, with scalers fit only on the training data and then applied to the test set. Hyperparameters for all models were selected based on established best practices and limited grid searches to ensure fair and consistent training conditions. Model performance was evaluated strictly on the withheld test data to assess generalizability and real-world predictive power.

Methodology (Baseline Models and Evaluation Metrics)

To contextualize the accuracy of each machine learning model, several baseline approaches were employed. These included a naive baseline that simply predicted the previous day's closing price, a midpoint baseline using the average of the prior day's open and close, and a previous open baseline that guessed the next close would equal the prior open. Model performance and baseline predictions were evaluated using the average Mean Squared Error (MSE) for next-day close price prediction across all 12 stocks, and directional accuracy (the fraction of correctly predicted up/down moves) to assess classification-like performance, again averaged across the model's performance on all 12 stocks. This combination of metrics provided a robust framework for comparing models and gauging practical forecasting value.

Methodology (Feature Importance and Visualization)

To interpret model behavior and identify key predictors, permutation feature importance was computed for each model, quantifying the impact of each feature by measuring the increase in out-of-sample MSE when its values were randomly shuffled. Bar plots were used to visualize the relative importance of each feature, highlighting which technical indicators and engineered

variables most influenced model predictions. In addition, line graphs comparing predicted and actual close prices for each model were generated, enabling visual inspection of tracking accuracy and periods of divergence. These visualizations not only clarified the strengths and weaknesses of each approach but also provided insight into the drivers of price movement in the dataset.

Results

Linear Regression

The linear regression model achieved an average test MSE of 31.19 across all stocks, outperforming the LSTM (1236.76), Random Forest (2785.90) and XGBoost (3018.78) models by a massive margin. However, it still underperformed compared to the simple “previous close” baseline, which had a lower average MSE of 24.43. Other baselines, such as the previous midpoint (27.14) and previous open (36.95), placed linear regression squarely in the middle of the pack. In terms of directional accuracy, linear regression averaged 50.16%, trailing the naïve baseline that always guessed up, which stood at 52.74%.

Permutation feature importance analysis highlighted MA5, MA10 and low price as consistently valuable predictors, while RSI14 and returns had minimal impact, a trend observed across most stocks. Visually, as demonstrated in Figure 4, the linear regression model tracked actual closing prices closely, though it often faltered during periods of high volatility or abrupt market shifts.

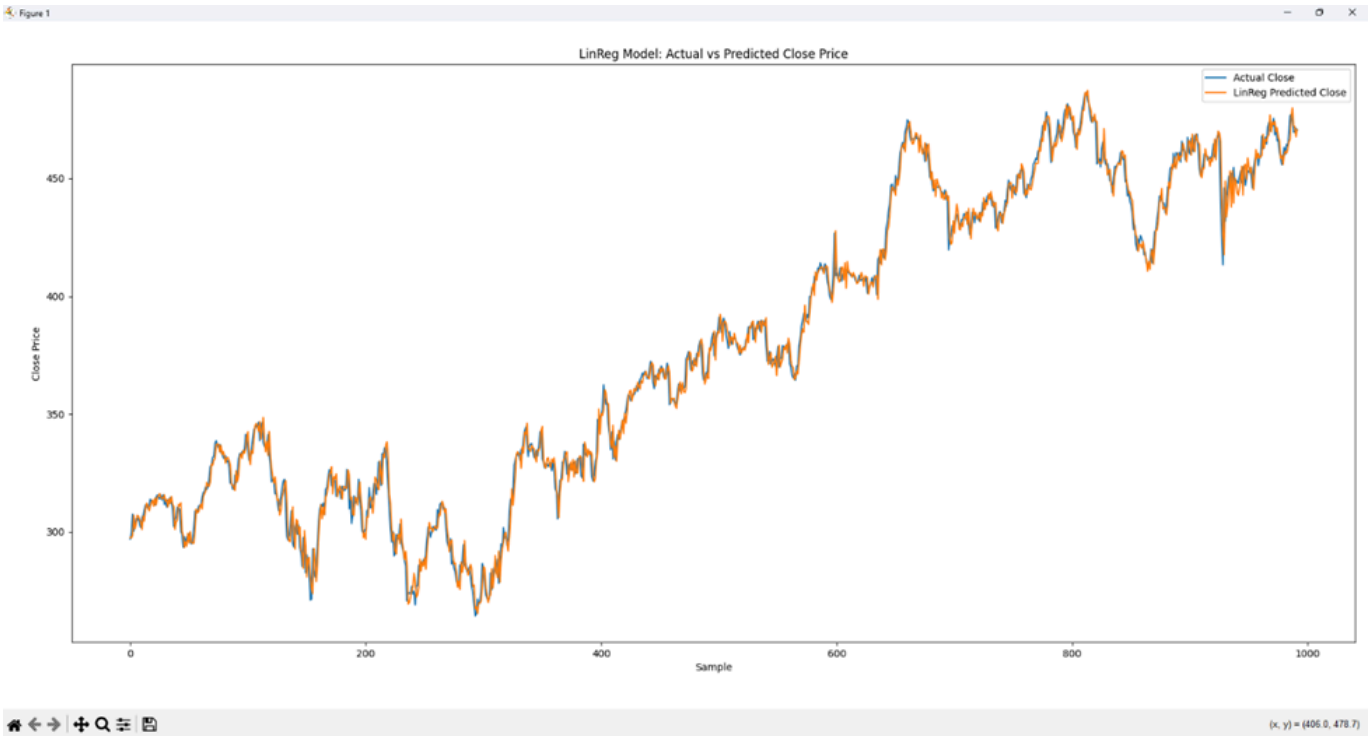

Figure 3: Linear Regression model's actual vs predicted closing prices

This suggests that while linear regression is adept at capturing broad trends, it lacks the flexibility to model non-linear dynamics, a key limitation in financial time series forecasting.

The plot above visually reinforces the strength of the Linear Regression model. The predicted closing prices (orange) extremely closely mirror the actual prices (blue) across the entire time series, with minimal deviation even during volatile swings. This consistency suggests that the model effectively captured the overall trend and structure of the data, likely due to its heavy weighting of features like open, close, and MA10, which reflect short term trends and momentum well. While it may still struggle with sharp inflection points, its performance in stable and trending periods is impressively accurate.

Despite its simplicity, linear regression holds up surprisingly well, outperforming all machine learning models except Lasso Regression, which was the only model to beat all three naive baselines in MSE.

XGBoost

The XGBoost model, a powerful gradient-boosted tree algorithm, demonstrated the poorest performance out of all five models in this study. It produced an average test MSE of 3018.78, significantly higher than Linear Regression (31.19) and Lasso Regression (26.80), and even dramatically worse than the naive baselines. The best baseline, predicting the next close as the previous close, had a much lower average MSE of 24.43, while the previous midpoint and previous open baselines posted 27.14 and 36.95, respectively. This means XGBoost was not only outperformed by simpler linear models, but also by extremely basic heuristics.

In terms of directional accuracy, XGBoost scored an average of 47.81%, below all other models and trailing the naive "always up" baseline of 52.74%. This highlights its inability to even beat random guessing across most stocks.

Permutation feature importance across the dataset revealed that features like close, low, and open prices were most impactful to the model's decisions. However, other features such as MA10, ATR14, and Volume MA5 frequently had negative or near-zero importance, indicating redundancy or possibly noisy signal contributions. This is likely due to overfitting in the presence of many correlated inputs, which is a common pitfall of complex ensemble models when applied to limited or noisy time series data.

As illustrated in the plot below, XGBoost's predicted closing prices tended to lag or underreact during periods of high volatility, especially during large price swings. The model smoothed out

sharp rallies and sell-offs, making it poorly suited for assets with high volatility or non-stationary behavior, a major weakness in financial forecasting tasks where sudden regime changes are frequent.

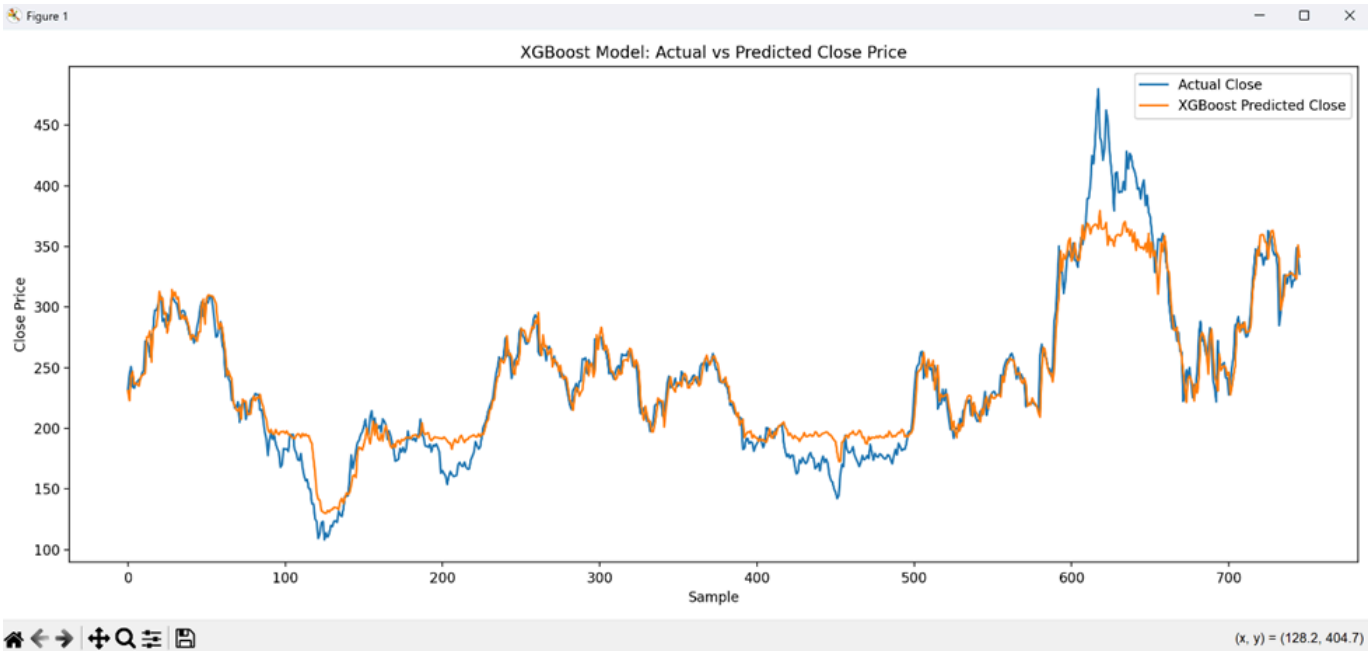

Figure 4: XGBoost's actual vs predicted closing prices

While XGBoost is often a top-tier performer in other domains (e.g., classification, structured data competitions), it failed to deliver in this setting due to the sequential, noisy, and nonlinear nature of financial time series data. Without added temporal context or engineered lag features, its tree-based architecture struggled to adapt.

Random Forest

The Random Forest model, an ensemble of decision trees known for its robustness and resistance to overfitting, produced mixed results in the context of next-day stock closing price prediction. Across all 12 stocks, Random Forest recorded an average MSE of 2785.90, placing it as the second worst-performing model after only XGBoost (3018.78), and far behind both Lasso and Linear Regression (26.80 and 31.19, respectively). This indicates that despite its flexibility and non-linear learning capability, Random Forest struggled significantly in forecasting continuous values in the financial domain.

When compared to the naïve baselines – Prev. Close (24.43), Prev. Midpoint (27.14), and Prev. Open (36.95) – Random Forest underperformed them all by a large margin. This highlights that the model, despite its complexity, failed to outperform even the most basic heuristics that simply reused the previous day's data.

In terms of directional accuracy, Random Forest achieved an average of 48.21% across all 12 stocks, which also fell short of the baseline directional accuracy of 52.74%. It slightly edged out XGBoost and LSTM but still failed to break the 50% mark, showing its weakness in capturing short-term directional shifts in stock prices.

Permutation feature importance analysis indicated that Random Forest heavily prioritized close price, which was consistently the most influential feature across most stocks. However, features such as MA10, Support and Returns were frequently ranked lowest, often contributing negative or negligible influence. This may indicate high redundancy or ineffective signal contribution in a model already prone to averaging effects across its ensemble structure.

The visualized predictions below further support this performance profile. The orange predicted line frequently lags the actual closing price (blue), particularly during sharp upward movements and spikes, such as the surge around sample 600. This lag and smoothing effect are typical of Random Forests, which due to their reliance on averaging multiple trees, tend to dampen high volatility and fail to capture abrupt regime shifts. In trending or sideways markets, the model performs relatively well, but its underreaction to price spikes underscores a fundamental limitation in time-sensitive financial forecasting tasks.

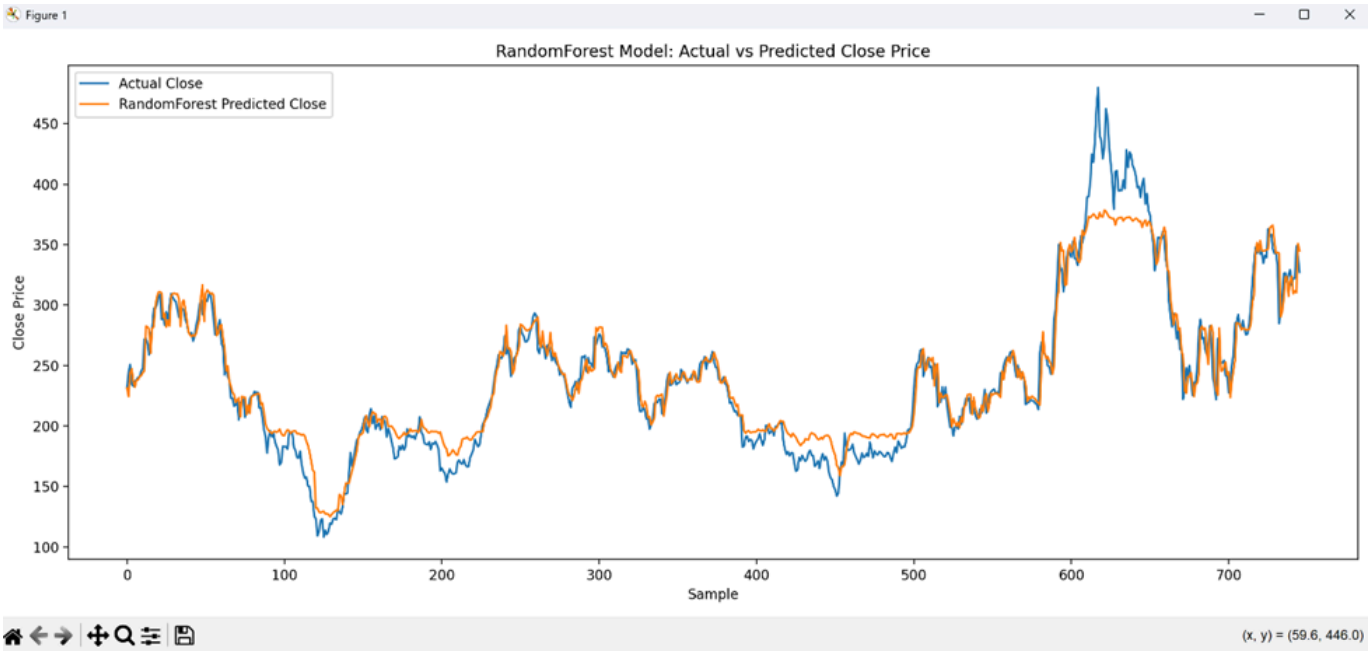

Figure 5: RandomForest's actual vs predicted closing prices

Overall, while Random Forest offers strong performance in many static classification or regression tasks, it is poorly suited for this financial time series context without significant feature engineering or temporal modeling enhancements. Its inability to adapt to rapid market shifts, combined with its bloated average MSE, makes it an inefficient choice for stock price prediction in its default form.

Long Short-Term Memory Network

The LSTM (Long Short-Term Memory) model, often considered a go-to architecture for time series prediction, delivered disappointing results in the context of next-day closing price forecasting. Across all 12 stocks, LSTM produced an average MSE of 1236.76, which was directly in the middle of the pack, worse than the regression models but better than the ensemble models. Lasso Regression (26.80) and Linear Regression (31.19) were not only more accurate, but also far simpler. LSTM's performance trailed behind every naive baseline tested, including Prev. Close (24.43), Prev. Midpoint (27.14) and Prev. Open (36.95).

In terms of directional accuracy, the LSTM model averaged 47.92%, which is below the naïve directional baseline of 52.74% that simply always predicts upward movement. This further shows that the model lacked the sensitivity required to capture short-term market direction or momentum, despite its design being tailored to sequence learning.

One possible explanation for this performance gap is the lack of explicit temporal feature engineering. While LSTM is theoretically capable of learning dependencies across time, it struggles when the time series lacks clear structure or when the data is noisy, which is often the case in financial markets. Without lagged features, seasonality indicators, or volume-based volatility adjustments, the model fails to leverage its full capabilities and ends up overfitting to noise.

Feature sensitivity analysis showed that LSTM most often prioritized close price, last close and open, while consistently deprioritizing ATR14 and volume. This pattern suggests that the model leaned heavily on recent price action rather than exploiting broader technical indicators.

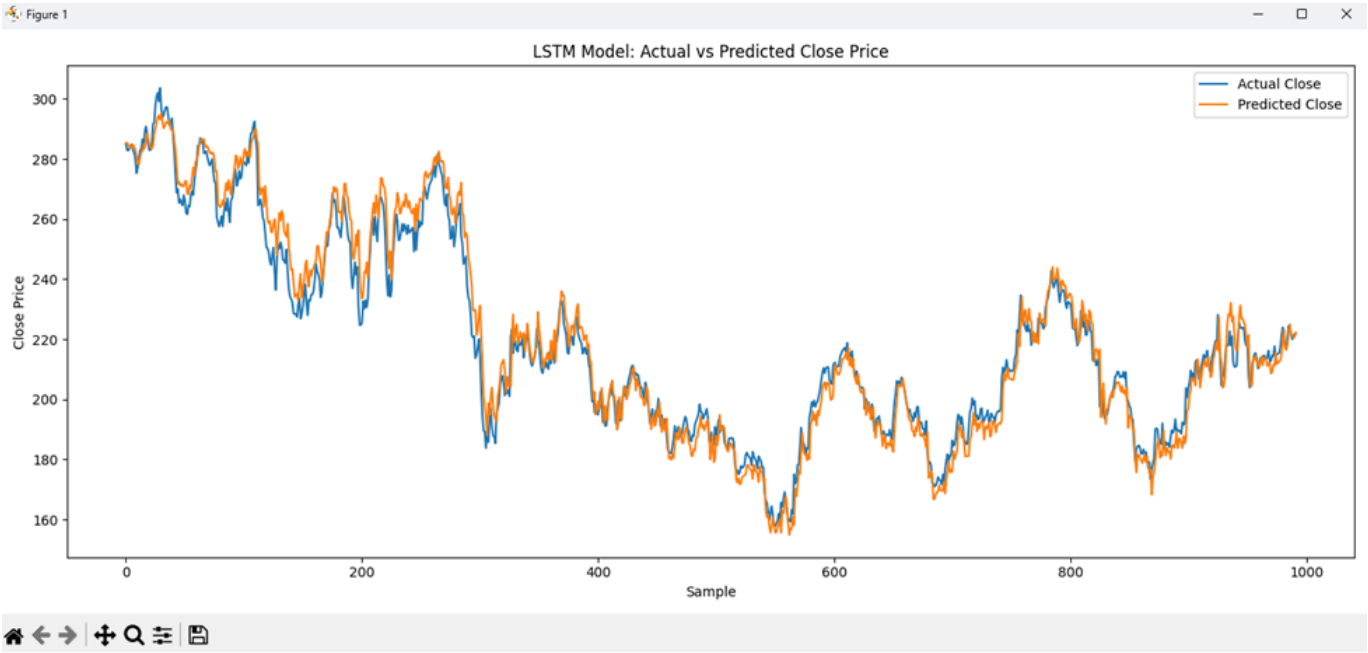

Figure 6: LSTM's actual vs predicted closing prices

The prediction chart in the plot above visually confirms this weakness. The orange predicted line trails behind the actual close prices in several regions, especially during sharp directional shifts or trend reversals. While LSTM occasionally tracks the overall trend, it frequently lags and smooths out volatility, showing a delayed response to sudden spikes or dips. This behavior is common in RNNs that are not fine-tuned for market regimes or volatility structures.

Overall, while LSTM is powerful in theory and widely used in time series domains like weather, speech, and demand forecasting, it underperformed drastically in this stock market forecasting setting. Without more advanced temporal features, regularization techniques, or hybrid architectures, LSTM's deep complexity only served to amplify error and reduce reliability.

Lasso Regression

The Lasso Regression model emerged as the most effective performer in this study. With an average MSE of 26.80, it was the only model to consistently beat all three naïve baselines: Prev. Close (24.43), Prev. Midpoint (27.14) and Prev. Open (36.95). While its margin over the best baseline was narrow, Lasso's success lies in combining simplicity with strong generalization, outperforming far more complex models like LSTM (1236.76), XGBoost (3018.78), and Random Forest (2785.90).

In terms of directional accuracy, Lasso achieved an average of 48.05%, slightly better than LSTM and XGBoost but still below the naïve "always up" baseline of 52.74%. Although it didn't

dominate in directional calls, Lasso's ability to minimize squared error gives it a clear edge in price prediction accuracy.

What sets Lasso apart is its built-in feature selection ability through L1 regularization. Across the 12 stocks, Lasso consistently assigned high weights to the close price, while eliminating many redundant or noisy indicators by assigning them zero weights. Commonly zeroed-out features included Volume, RSI14, ATR14, and returns, which suggests that these metrics offered little marginal value once price-based features were accounted for. This lean feature set likely contributed to the model's strong bias-variance tradeoff and lower overfitting.

Figure 7: Lasso Regression model's actual vs predicted closing prices

The plot above confirms this performance visually. The predicted close price (orange) tightly tracks the actual price (blue) throughout the entire sample. Even during more volatile periods, the model maintains close alignment with the real price, demonstrating its ability to generalize across varying market conditions. Unlike other models that smooth or lag, Lasso stays responsive and accurate.

Lasso's overall performance highlights a critical insight for financial forecasting: simpler models can outperform complex ones when noise dominates signal. In this context, the penalty for overfitting worked in Lasso's favor, helping it outperform models that were more prone to memorizing irrelevant patterns. While it may not capture sudden directional shifts perfectly, its consistency in tracking real prices makes it a highly practical choice for daily close prediction.

Comparative Model Performance and Feature Importance

A comprehensive evaluation of both mean squared error (MSE) and directional accuracy across twelve diverse stocks reveals significant differences in model performance. As shown in Figure 8, Lasso Regression achieved the lowest average MSE of 26.80, outperforming all other machine learning models and even the naive baselines of previous open (36.95) and previous midpoint (27.14). The only baseline that narrowly outperformed Lasso was the previous close, with an average MSE of 24.43. Despite this small gap, Lasso was the only model to consistently beat all three baselines, solidifying it as the most accurate and generalizable model in the study.

	LSTM	**Lasso Reg**	**Linear Reg**	**Random Forest**	**XGBoost**
JPM	997.30	9.77	8.91	1617.18	1725.81
SPY	2154.77	36.84	35.40	6509.84	6821.08
NVDA	3474.20	7.02	7.8067	3741.28	3782.6631
UNH	2760.20	101.20	124.97	9825.60	11041.92
GOOGL	240.46	8.93	8.76	866.20	927.60
GE	28.81	6.80	6.53	77.33	84.50
KO	7.69	0.50	0.46	30.43	33.00
XOM	104.92	2.98	3.22	68.06	85.69
DUK	3.45	1.62	1.68	41.30	49.24
AMT	36.45	15.66	17.75	33.26	54.97
LIN	4788.92	30.49	26.66	10341.45	11247.86
TSLA	244.08	99.78	132.15	278.95	371.12
Average MSE	***1236.76***	***26.80***	***31.19***	***2785.90***	***3018.78***

Figure 8: Comparison of average Mean Squared Error (MSE) of each model across all 12 stocks

Linear Regression followed closely with an average MSE of 31.19, demonstrating that even simple models can be effective in financial forecasting. In stark contrast, more complex methods like XGBoost (3018.78), Random Forest (2785.90), and LSTM (1236.76) performed substantially worse. These models, while theoretically capable of modeling nonlinearities and sequential dependencies, struggled to generalize due to high feature correlation, limited sample size, and the inherently noisy nature of stock price movements.

A closer look at individual stock-level MSEs reveals further insights. As seen in Figure 8, stocks like KO (The Coca-Cola Company) had extremely low MSEs across all models. For instance, Lasso and Linear Regression achieved MSEs of 0.50 and 0.46 respectively. This is likely because KO exhibits low volatility, relatively stable price action, and smooth trends, making it easier to model with linear methods. On the other hand, stocks such as UNH (UnitedHealth Group) and LIN (Linde plc) posted exceptionally high MSEs for all models, particularly for Random Forest and XGBoost which exceeded 10,000. These large-cap stocks tend to have frequent sharp movements, irregular jumps, or volatile earnings swings, which traditional regression models and even tree-based models struggle to predict without lagged or event-driven features. This stock-level heterogeneity underscores the need to tailor models to the volatility and structure of the underlying asset.

	LSTM	Lasso Reg.	Linear Reg.	Random Forest	XGBoost	Baseline Model
Average Directional Accuracy	**47.92%**	**48.05%**	**50.61%**	**48.21%**	**47.81%**	**52.74%**

Figure 9: Comparison of average directional accuracy across all 5 models

Directional accuracy, or the ability to predict whether the next day's closing price would increase or decrease, tells a more nuanced story. As illustrated in Figure 2, none of the models exceeded the naïve directional baseline of 52.74%, which simply assumes the stock will always go up. The highest accuracy among models came from Linear Regression at 50.61%, followed by Random Forest (48.21%), Lasso Regression (48.05%), LSTM (47.92%) and XGBoost (27.81%). These results highlight a core challenge of financial prediction: minimizing prediction error is not the same as correctly guessing direction. Linear and Lasso models achieved strong MSEs by

tracking overall trends, but lacked the sensitivity to catch short-term directional shifts. Meanwhile, deep models like LSTM, despite being designed for sequence learning, failed to extract meaningful temporal dynamics and underperformed both in accuracy and MSE.

The clear success of Lasso Regression lies in its ability to eliminate irrelevant features through L1 regularization. Across all stocks, Lasso frequently zeroed out noisy or redundant indicators such as RSI14, returns, and volume-based metrics, while emphasizing reliable predictors like close price, MA5, and low. This led to simpler, more interpretable models with stronger out-of-sample performance. Linear Regression benefited from many of the same features but lacked regularization, making it slightly more vulnerable to overfitting. In contrast, XGBoost and Random Forest showed inconsistent feature importance rankings and often overemphasized weak features like ATR14 or support, likely due to their greedy splitting mechanisms and sensitivity to correlated data.

LSTM, often seen as a powerful sequence model, fell short in this context. Without the addition of time-delayed lags, trend indicators, or seasonal encodings, the model failed to learn meaningful patterns and instead produced smoothed predictions that lagged behind real price movement. This resulted in poor MSE scores and uninspiring directional accuracy.

Overall, these results demonstrate that model complexity does not guarantee better performance in financial forecasting. In fact, the most consistent performers were simpler, regularized linear models that relied on a narrow set of strong, price-based features. When signal is weak and noise dominates, as is common in equity markets, models that penalize overfitting and focus on trend-following characteristics prove far more robust. This study reinforces the notion that recent price behavior, short-term averages, and minimalistic design often yield the most reliable forecasts in a volatile and stochastic environment like the stock market.

Conclusion

This study set out to evaluate the predictability of next-day equity closing prices using five distinct modeling approaches: Linear Regression, Lasso Regression, Random Forest, XGBoost, and Long Short-Term Memory (LSTM) neural networks. By applying each model to a consistent feature set across 12 diverse publicly traded stocks, this research aimed to determine whether complex machine learning architectures provide a meaningful advantage over simpler, regularized models and naive baselines in short-term forecasting.

The results strongly favored Lasso Regression, which achieved the lowest average mean squared error (MSE) of 26.80, beating all other models and every naive baseline except the previous close (24.43). Linear Regression also performed well with an average MSE of 31.19, while Random Forest (2785.90), XGBoost (3018.78), and LSTM (1236.76) delivered

significantly worse results. Despite their complexity, these models failed to generalize effectively, particularly on stocks with higher volatility or irregular movement.

Directional accuracy metrics, shown in Figure 2, revealed that none of the models outperformed the naive “always-up” strategy (52.74%). The highest model accuracy came from Linear Regression at 50.61%, with others hovering around 48%. This highlights a key distinction between minimizing prediction error and accurately forecasting price direction, a task made challenging by the noisy, mean-reverting nature of daily market movements.

Feature importance analysis explained these differences. The best-performing models focused on a minimal set of price-driven features, especially close price, MA5, and low, while filtering out indicators like RSI14, returns, and volume averages. Complex models failed to leverage their theoretical advantages due to overfitting and a lack of meaningful signal in the added features.

These findings reaffirm a critical insight in financial machine learning: model complexity does not guarantee better predictive power. In fact, when forecasting noisy financial time series with conventional technical features, simpler and regularized models like Lasso can outperform advanced architectures in both accuracy and consistency. This supports the idea that parsimony, interpretability, and proper regularization often offer greater practical value than brute complexity in noisy real-world data.

That said, limitations remain. The study focused exclusively on daily bars, technical features, and a fixed univariate target (next-day closing price). It did not account for trading frictions, market microstructure, or multi-step forecasting. Model tuning was intentionally minimal to focus on architectural differences rather than hyperparameter optimization.

Future research could extend this framework by incorporating alternative data types (e.g., macroeconomic indicators, sentiment, or order flow), experimenting with more sophisticated temporal encoding or ensembling techniques, and applying the methodology to other asset classes or intraday intervals. Evaluating models across distinct market regimes and incorporating risk-adjusted or probabilistic metrics would also add depth and practical value.

In conclusion, this study provides strong evidence that in the context of short-term stock price forecasting, simplicity and robustness often outperform complexity and scale. For practitioners and researchers alike, the results serve as a reminder that the right model is not always the most powerful one, but the most appropriate for the signal available.

Code Availability

The code produced by the author in this study is available here (https://tinyurl.com/yup267rk).

References

Bailey, D. H., Borwein, J., Borwein, J., López de Prado, M., & Zhu, Q. J. (2015). *The Probability of Backtest Overfitting* (February 27, 2015). Journal of Computational Finance (Risk Journals), 2015, Forthcoming. Available at SSRN: http://dx.doi.org/10.2139/ssrn.2326253

Colah, C. (2015). *Understanding LSTM Networks [Figure]*. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Chen, T., & Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System*. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. https://doi.org/10.1145/2939672.2939785

Deniz, G. (2019). *Random Forest [Figure]*. Medium. https://medium.com/@denizgunay/random-forest-af5bde5d7e1e

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research, 270*(2), 654–669. https://doi.org/10.1016/j.ejor.2017.11.054

Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). *Array programming with NumPy*. Nature, 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). *Matplotlib: A 2D Graphics Environment*. Computing in Science & Engineering, 9(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Investing.com. (n.d.). *S&P 500 Historical Data*. https://www.investing.com/indices/us-spx-500-historical-data

Krauss, C., Do, X. A., & Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research, 259*(2), 689–702. https://doi.org/10.1016/j.ejor.2016.10.031

McKinney, W. (2010). *Data Structures for Statistical Computing in Python*. Proceedings of the 9th Python in Science Conference, 51–56. https://doi.org/10.25080/Majora-92bf1922-00a

Paszke, A., et al. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Advances in Neural Information Processing Systems, 32, 8024–8035. https://pytorch.org/

Pedregosa, F., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830. https://scikit-learn.org/

Twelve Data. (n.d.) *Stock Market & Financial Data API*. https://twelvedata.com/