A Comparative Analysis of Regressor Machine Learning Models in Forecasting SPDR S&P 500 ETF Trust (SPY) Movements

Puranjay Haldankar & Seungwoo Lee

Abstract

Stock market forecasting remains one of the most challenging tasks in finance due to the market's high volatility, complex dynamics, and sensitivity to external events. This study investigates the short-term predictive performance of five regressor models on the S&P 500 ETF Index (SPY). There are two experimental setups for evaluating models: one incorporating technical indicators and one excluding them. The aim is to determine whether technical indicators enhance prediction accuracy and to identify which model is most effective for short-term forecasting. Also, the model's performance is assessed using MAE, RMSE, R2 score, and directional accuracy. The results showcase that the R2 score is a poor indicator in short-term financial datasets. Directional accuracy with technical indicators ranged from 51-54%, a result better than randomly guessing. Without technical indicators, models ranged from 45-48%, highlighting the importance of technical indicators in predictions.

1. Introduction

The SPDR S&P 500 ETF Trust (SPY) is one of the most widely traded exchange-traded funds in the world, designed to track the performance of the S&P 500 index. Launched in 1993, it was the first ETF listed in the United States and remains the largest by assets under management. SPY holds shares of all 500 companies in the index, providing investors with broad exposure to the U.S. equity market in a single, liquid security. Because of its size, liquidity, and close correlation with the S&P 500, SPY has become a benchmark instrument for institutional investors, traders, and researchers seeking to measure or replicate the performance of the U.S. stock market.

To predict possible price movements, traders often use technical analysis to find trends or patterns in data, such as predicting the S&P 500's SPDR ETF (SPY) value during its opening, closing, highs, and lows [1]. In this research, we will be comparing five different regressor models - Simple Linear, Gradient Boosting, Random Forest, XGBoost, and Prophet - using the historical 2014 to 2018 SPY data for training and testing for 2019 predictions.

Furthermore, the research aims to fine-tune and evaluate each of the five models, that is, adding feature engineering and comparing it to its baseline model. To find which model performs the best overall, and give insight into the question of whether adding features to a model makes a notable difference in predicting the stock market.

2. Background Information and Literature Review

2.1 Linear Regressor

The linear regression model is one of the fundamental models that most prediction models derive from. It is a simple model that assumes a linear relationship between the inputs (independent variables) and the outputs (dependent variables). The basis of the linear regression formula is:

$$y = \beta 0 + \beta 1x1 + \beta 2x2 + \cdots + \beta nxn + \epsilon$$

Where:

- y = predicted value
- β = coefficients
- x = input features
- ϵ = residual value

β is often referred to as a regression coefficient and defines the slope of the input to the next step, and when combined with all of the *x* and β, it equals the overall slope of the data points [2]. And, ϵ, the residual value, indicates the extent to which the average value of the data deviates from the linear relationship. This model is high-speed to train and interpret, which is why most use it as a baseline model. However, the downsides of linear regression are its linear relationship, which is often not realistic in real data.

Furthermore, there are usually assumptions that must be satisfied by the data for the regression model to be totally valid. Some of these assumptions are that for each value of *x* and *y*, observations, which were without measurement error, and the relationships between *y* and each of *x,* should be linear in the parameters of the specific functional form chosen [3].

These assumptions are why linear regression models became a baseline model, as they often cannot accurately represent the data values, and are frequently not used when there are extreme outliers. It is important to note that the above model is a simple linear regression model, and other forms of linear regression models exist that are more complex and contain more notable features. Though this research does not examine other linear regressions, such as multiple linear regression, it is for the sake of comparing a basic model to a more complex one to show an apparent difference. Therefore, in this paper, we are using linear regression on our data as the standard against other regression models in our analysis.

Key Components of Simple Linear Regressor

Independent Variable (*x*): The single predictor used to estimate the target variable.

Dependent Variable (*y*): The outcome or target variable.

Model Coefficients (β_0, β_1):

- β_0 (intercept) is the predicted value of y when x = 0.
- β_1 (slope) represents the change in y for a one-unit change in x.
- Equation: $y = \beta_0 + \beta_1 x + \epsilon$

Residual (ϵ): The difference between the predicted and actual value (error term).

Loss Function (Usually MSE): Mean Squared Error (MSE) = average of squared residuals.

Advantages of Simple Linear Regressor

1. The simplicity and interpretability of the linear regressor are easy for anyone with a basic understanding of algebra to understand.
2. A simple linear regressor is computationally efficient and can train extremely efficiently.
3. It works best with linear trending data compared to data with more outliers.

Limitations of the Simple Linear Regressor

1. It can only capture linear relationships and fails in environments where the data is highly variable.
2. It is assumption-heavy for it to be valid.
3. It is outlier sensitive, where too many of them can corrupt its accuracy.

2.2 Gradient Boosting Regressor

The gradient boosting regression model is the foundation of more popular models like XGBoost, which will be covered later in the study. It is an ensemble model that builds trees sequentially, where each new tree learns to fix the errors made by its predecessors. This model originated from Freund and Schapire's work on weighted iterative classification, where they created an algorithm that used gradient descent [4]. Gradient boosting is named as it is because of the use of gradient descent (an optimization algorithm used to minimize the loss function) to change parameters to reduce errors.

It is also highly customizable to the particular needs of the application and learns to respect the different loss functions [5]. In simple words, gradient boosting is a learning model that combines the outputs of predictors to produce a robust prediction. It is one of the best-performing models in practice because of its algorithm, and it can handle non-linear relationships and optimizations well. However, the downsides of this model are that it is slow to train, can overfit if not tuned properly, and is complex to interpret. In this study, we will use gradient boosting to compare its results primarily with its most similar model, XGBoost, to gain a deeper understanding of what truly makes one model predict the stock market more accurately.

Key Components of Gradient Boosting

Ensemble of Weak Learners (Decision Trees): Each tree corrects the errors of the previous one.

Additive Learning Process: Starts with a base prediction. Adds new trees trained on residual errors:

$$Fm(x) = Fm-1(x) + \eta \cdot hm(x)$$

Where η is the learning rate, and hm(x) is the m-th tree.

Learning Rate (η): Controls the contribution of each tree. The smaller the learning rate, the more conservative and accurate, but it needs more trees.

Subsampling (Stochastic Gradient Boosting): Randomly selects a subset of data at each step, which reduces variance and improves generalization.

Advantages of Gradient Boosting

1. Gradient boosting can capture non-linear patterns well, and it is exceptional at modeling complex interactions between values.
2. It can handle missing data and outliers successfully using tree-based learners and is not affected by outliers, unlike linear models.
3. It can also contain feature importance, indicating which variables are more influential on its predictions, and can help guide to a more accurate overall prediction.

Limitations of Gradient Boosting

1. Gradient boosting takes much longer to train compared to a linear regressor because it trains hundreds of decision trees at a time.
2. Gradient boosting is likely to overfit the data if there are too many training decision trees.
3. It is necessary to have a variety of parameters to obtain more accurate readings, which, without them, leads to overfitting the data.

2.3 Random Forest Model

Random Forest is a further refinement of classification and regression from the Cox models and CART models (6). A Cox model is patterned after a linear regression, and its outputs are presented as coefficients, which can be easily transformed. It is a popular machine learning model in clinical and medical fields because of its ability to predict the overall risk of the outcome accurately. A CART model is straightforward, non-parametric, and used in either classification or regression. A CART's output is referred to as a "decision tree," most commonly because of the way each of its predictors splits into how a tree's branches are grown. It selects the optimal decision based on recursive partitioning (a method for constructing binary trees by iteratively splitting data into subgroups based on optimal splitting rules) until there is no longer further discrimination [6].

The Random Forest uses the ideas from the Cox and CART models to create a large number of decision trees, and the outputs from these trees combine into a voting system for classifying problems or averaging the regressions. RF uses randomization in two ways: the first is bootstrap sampling (a resampling technique used to estimate the sampling distribution of a statistic by repeatedly drawing random samples with replacement from an original sample), and the second is at the decision nodes, where it selects a certain number of predictors (decision trees). Then the algorithm tests all possible thresholds for all selected variables and chooses the variable-threshold combination that results in the best split. This algorithm reduced the number of variables needed to obtain the optimal prediction. There are many applications for when the random forest is useful, typically as a medical decision support tool and prediction model for stock market prices to make a profit as investors [7]. In this study, we will compare all five regressor models to the random forest model, which also predicts stock market prices.

Key Components of RF

1. Bootstrap Aggregating: Random sampling with replacement ensures diverse training sets for each tree.
2. Ensemble Averaging: The results of all the decision trees are averaged to give a final output.
3. Decision Tree: Built upon multiple individual decision trees. Each tree is trained on a random subset of the data and makes its prediction. The final result is based on the aggregation or average of all trees.
4. Tolerance to Missing Values: The ensemble can still perform well when some input features are missing, as not all trees rely on the same features.
5. Random Feature Selection: At each node split within a tree, only a random subset of features is considered. This feature creates more randomness and prevents overfitting.

Advantages of RF

1. Random Forest can model complex and non-linear interactions between features without requiring explicit feature engineering, which is helpful in unpredictable financial data.
2. By averaging the predictions of many decision trees, Random Forest reduces the risk of overfitting, especially compared to single decision trees.
3. RF provides measures of feature importance, allowing researchers to identify which variables (e.g., past price, volume, technical indicators) are most influential in prediction.

Limitations of RF

1. Not ideal for sequential data, RF does not inherently account for temporal dependencies in time series data, making it less suited than models like LSTM for capturing trends over time unless time-based features are carefully engineered.
2. Training many decision trees with large datasets can become resource-intensive and slow, especially when dealing with large stock datasets or numerous hyperparameter combinations.
3. Less interpretable compared to linear models, RF is more interpretable than deep learning models; however, it still functions as a “black box” model, making its decision process harder to explain compared to linear regression.
4. Random forest regression cannot interpolate data, and its maximum and minimum values are bound to its highest and lowest training set data, limiting the accuracy of its predictions.

Random Forest Classifier vs Random Forest Regressor Model

Deriving from the same random forest model, it is essential to distinguish the differences between the Random Forest Classifier (RFC) and the Random Forest Regressor (RFR). The primary differences lie in the outputs of each model and its corresponding prediction strategy. The outputs for the RFC are labeled as “yes” or “no”; meanwhile, the RFR outputs a numerical value. RFC also differs from RFR by its prediction strategy, where it aggregates the majority vote of the decision trees to come to a desired outcome, while RFR uses averaging all of the decision trees to come to a prediction [8]. For example, in a scenario of students’ test results, RFC will predict if the majority passes or fails, while RFR will try to predict the exact score of the sample of students. In this research, we will compare the results of RFR with and without features for a comprehensive analysis of the data.

2.4 Prophet Forecasting Model (PFM)

The Prophet forecasting model (PFM) developed by Meta allows for forecasting a desired outcome over time [9]. PFM can predict accurately without unnecessarily extensive complex parameters like LSTM [9]. This feature increases the functionality of this model to be used simply by people without too much knowledge of machine learning. PFM can also perform even if there are missing values or extreme outliers, allowing for the overall prediction of the model to function more easily than other models. Prophet is ten times easier and learns faster from training than other ARIMA (Autoregressive Integrated Moving Average) models, also used to forecast and analyze data [10].

PFM was specifically designed and developed to be accessible for experts and non-experts in the machine learning and forecasting world [11]. The main differentiator from other traditional models is the clear way to identify patterns, specifically patterns, holidays, and trends, with sudden changes seen after training the model. However, the Prophet’s downside from other models is that it is not helpful for high-frequency data in finance. Therefore, it would not be able to do poorly on short-term volatile patterns. Despite this, PFM is another strong, effective model for mid-term to long-term forecasts, and in this study, we will include PFM to compare its SPY forecasts to the other four regressor models.

Key Components of PFM

1. PFM takes a few seconds to fit the model with tunable parameters. Regularly models time series of datasets with trends, seasonality, and holidays. The following formula represents PFM:
    a. $y\ (t) = g\ (t) + s\ (t) + h\ (t) + \epsilon t$
    b. This is the generalized prediction equation, and it uses the model PFM’s trends, seasonality, and holidays (10).
2. Trend: Prophet captures the underlying direction of the market, whether the SPY is generally trending upward or downward over time, by fitting either a piecewise linear or logistic growth curve. For stock data, the linear model is most commonly used. It can also detect structural breaks in trends, such as major crashes or rebounds, through automatically selected change points.
3. Seasonality: Prophet models capture recurring patterns, for example, weekly seasonality of Monday dips and Friday rallies, or yearly patterns like surges in trading during holiday

time. Prophet uses these periodic effects to match the cyclical market behaviors over the years.
4. Change Points: The model automatically detects change points, which are moments where the trend shifts sharply. Users can also manually specify them if needed.

Advantages of PFM

1. Prophet has a built-in seasonality detection that automatically detects and models daily, weekly, and yearly seasonality, which is helpful for markets that exhibit varying behavior across months, weeks, or even days (e.g., holiday effects, end-of-month volatility).
2. Fast and straightforward to implement with only a few lines of code, users can build and deploy a Prophet model without deep knowledge of time series theory or tuning complex parameters.
3. Prophet is quick to identify outliers and has robust data handling skills, such as in real-world situations where stock data may be incomplete or contain sudden spikes or crashes. It can still generate stable forecasts without requiring major preprocessing.

Limitations of PFM

1. Prophet is not suitable for intraday forecasting and is optimized for daily or longer timeframes. It performs poorly on minute-by-minute or hourly stock data, which is often needed in active trading.
2. Prophet models components (trend, seasonality, holidays) as being linearly added together, which can oversimplify complex non-linear interactions in daily stock price movements.
3. Prophet has limited feature integration and does not support external variables as flexibly as models like XGBoost. You cannot easily include factors such as trading volume, sentiment scores, or macroeconomic indicators.

2.5 Extreme Gradient Boosting (XGBoost) Prediction Model

XGBoost changed the way gradient boosting operated initially. In XGBoost, individual trees are created using multiple cores, and data is organized to minimize the lookup times [12]. The idea of boosting started with the idea of a weak learner becoming a better learner with features. First introduced as the AdaBoost algorithm, the weak learners (decision trees) were given more weights, and the strong learners were given less weight, thereby preventing the prediction model from overlearning the sample data. With this algorithm, by repeatedly adjusting the weight, a function of gradient boosting was created.

Gradient boosting is divided into three different steps. First is a proper differentiable loss function (a function used to measure how far off a prediction is from the result), then a weak learner (regression trees) makes predictions, and finally an additive model adds all of the projections up and puts it through the loss function and keeps repeating the process until a proper optimized value of loss function is reached [12]. XGBoost is similar to having a gradient boosting as its core, but it has the additional process of a multi-threaded approach, which uses the CPU core of a machine to lead to greater speed and performance than a regular simple gradient boosting (12). XGBoost is a solid model to predict the stock market, and the use of gradient boosting will improve the directional accuracy of its results, so the model does not overfit like other traditional models.

Key Components of XGBoost

1. Gradient Boosting: Used in regression and classification problems to build strong predictive models by combining many weak models to correct their errors.
2. Sparse Aware Implementation: Involving automatic handling of missing data values to support parallelization of decision predictions [12].
3. Tree Pruning: Unlike some models that grow the decision trees, XGBoost uses depth-wise pruning, which builds the trees entirely and removes the splits that don't reduce the loss and improve the model's simplicity.
4. Parallel and Distributed Computing: XGBoost can train weak learners in parallel, making it faster than simple gradient boosting models

Advantages of XGBoost

1. XGBoost uses optimized gradient boosting (a machine learning technique that builds predictions by combining multiple weak learners, usually decision trees) with parallel processing and efficient memory usage, providing high performance and speed compared to traditional models.
2. XGBoost uses regularizations (L1: Lasso Penalty, adding a penalty equal to the absolute value of coefficients which is helpful in feature selection & L2: Ridge Penalty, adds penalty equal to the square of the coefficients which reduces model complexity without eliminating essential features) to help prevent overfitting data, which means the model is memorizing the training data too much, eventually relying on it when performing on testing data.
3. XGBoost has high accuracy and often outperforms other algorithms in structured data tasks because of its capability to handle complex patterns.

Limitations of XGBoost

1. XGBoost is a black-box model (a model that is not easy to understand how the model made the decision), making it hard to interpret and falling behind simpler models like decision trees.
2. XGBoost requires careful feature tuning to perform optimally; otherwise, overfitting and poor data might result.
3. XGBoost performs best on structured and table-like data, and it is not ideal for sparse and unstructured data like audio and images. Although in this study, we are comparing XGBoost on numerical values from the SPY, this will not limit the ability to produce poor results.

2.6 Features

Other than open, close, high, low, and volume for all the data points of the SPY, additional features are required for more accurate prediction. The existence of only these features is insufficient in capturing complex patterns and daily discrepancies related to the stock market. Therefore, additional features must be derived and engineered from the pre-existing ones to enhance the predictive power of these models. These features will be used in all five of the regressors we will be testing. Results without and with feature engineering. These features include the following.

2.6.1 Simple Moving Averages

$$S.M.A = \frac{\Sigma A}{n}$$

Where *A* are the stock prices and *n* is the number of periods.

Simple Moving Averages allow for the model to gauge long-term trends in prices better, smoothening out short-term fluctuations.

2.6.2 Exponential Moving Averages

$$E.M.A = \frac{2}{N+1} \times p_t + (1 - \frac{2}{N+1})\times E.M.A_{t-1}$$

Where,
p_t Is the price at the given time
$E.M.A_{t-1}$ is the Exponential Moving Average of the previous time period
N is the time period

Exponential Moving Averages are similar to Simple Moving Averages. However, they give more weight to recent prices due to their recursive nature.
The first *E.M.A.* that is referred to by this recursion is simply the *S.M.A* of the stock prices at that given point in time.

2.6.3 Relative Strength Index

RSI identifies overbought and oversold shares. Overbought shares are usually shares that have risen too quickly due to excessive buying and hence signify that the stock may be due for a correction or pullback (RSI > 70). Similarly, Oversold shares are usually shares that have fallen too quickly due to excessive selling and thus might be due for a rebound (RSI < 30).

$$RSI = 100 - \frac{100}{1+\frac{avg\ gain}{avg\ loss}}$$

The average loss and average gain are calculated over a 14-day period.

2.6.4 Moving Average Convergence Divergence (MACD)

It is a financial indicator used to understand potential buy/sell signals. MACD is derived by comparing the *E.M.A.* values of two prices at different time points. Usually, the MACD is calculated by subtracting the 26-period *EMA* from the 12-period *EMA.*

2.7 Additional Features

- MACD Signal: The signal line of the MACD (Moving Average Convergence Divergence) indicator, typically a 9-day EMA of the MACD line. Used to identify potential buy/sell signals when it crosses the MACD line.
- BB Upper: Upper Bollinger Band, calculated as the moving average plus two standard deviations. Indicates overbought levels.
- BB Middle: Middle Bollinger Band, which is simply the moving average (usually 20-day). Acts as a trend baseline.
- BB Lower: Lower Bollinger Band, calculated as the moving average minus two standard deviations. Indicates oversold levels.
- Lagged Return 1D: Previous day’s return (percentage change from the day before). Captures short-term momentum.
- Close 1d: Closing price of the previous day. Useful for price-level context.
- Close 2d: Closing price from two days ago. Helps in modeling short-term trends.
- Close 3d: Closing price from three days ago. Adds more context to recent price movements.
- Return 3d: Return over the last three trading days. Captures short-term performance.
- Return 7d: Return over the last seven trading days. Measures weekly trend strength.
- Volume Change: Change in trading volume compared to the previous day. Indicates unusual market interest or activity.
- Volatility 7d: Rolling standard deviation of returns over the past 7 days. Measures short-term price fluctuation.
- Day Of Week: Numeric representation of the day of the week (0=Monday, ..., 4=Friday). Used to detect weekday seasonality or patterns.
- Month: Numeric month (1=January, ..., 12=December). Helps identify monthly trends or seasonal effects.

2.8 Metrics of Accuracy

Various metrics are employed to quantify the deviation between the predicted and actual values in dollar terms ($). These also include directional accuracy, which is the prediction accuracy of the gain/loss direction- another critical metric.

2.8.1 Mean Average Error

Mean average error is simply the absolute percentage prediction errors in dollar terms ($), providing insights into the absolute accuracy of the model across all the predictions and actual stock prices.

2.8.2 Root Mean Square Error

Root Mean Square Error follows a similar principle to Mean Average Error, except it punishes larger errors more than minor errors. Root Mean Square Error is the square root of the absolute error. As a result, greater errors are punished more severely. For example, the Absolute error 1 v/s 0.1 will have different weightages across root mean square error and mean average error.

2.8.3 R2 Score

The R-squared (R^2) or coefficient of determination is a statistical measure that represents the proportion of variance in the dependent variable that is explained by the independent variable(s) in a regression model. In simpler terms, it indicates how well the model fits the data, with higher values indicating a better fit.

2.8.4 Confusion Matrix

A Confusion Matrix is a table that summarizes the classification performance of any given model. It divides predicted data into true positives and negatives as well as false positives and negatives. While not entirely applicable for all the models tested, with stock market data, classification should be done based on directional accuracy, i.e., true or false prediction of an upward or downward trend.

3. Methodology

3.1 Data Collection

Historical stock data for SPY was sourced using the yfinance Python package. The training dataset spans from 1 January 2014 to 31 December 2018, while the testing dataset includes data from 1 January 2019 to 31 December 2019. Each entry contains daily Open, High, Low, Close, Volume, and Date information.

3.2 Feature Engineering

To improve the predictive performance of models, a plethora of features were created using the pre-existing basic information that was imported using the yfinance Python package. These engineered features were selected based on their widespread use in technical analysis and their ability to provide diverse signals related to momentum, trend, volatility, and time structure, which are critical for capturing the non-linear behavior of the financial market. The data captured across these features was not standardized in the final data collection.

3.3 Standardization

Standardization is a data preprocessing technique used to rescale numerical features so that they have a mean of 0 and a standard deviation of 1. This transformation ensures that all features contribute equally to the learning process, especially in models that are sensitive to the magnitude of input values. All the features in the training dataset were standardized.

3.4 Data Training & Testing

The regressor models were trained on data from January 2014 to December 2018 and tested on data from January 2019 to December 2019. The following datasets were chosen to simulate a real-world setup, wherein models are often trained to predict future prices and price movements based on significant historical data.

4. Results

Table 1: Regression Models with Technical Indicators

Model	MAE	RMSE	R^2 Score	Directional Accuracy	Confusion Matrix
Random Forest Regressor	10.8505	14.3005	-0.3475	54.35%	TP: 74 FP: 46 TN: 51 FN: 59
Prophet (Meta)	11.2047	14.9070	-0.0277	54.62%	TP: 82 FP: 47 TN: 54 FN: 66
XGB Regressor	11.3988	14.9603	-0.4748	53.91%	TP: 73 FP: 46 TN: 51 FN: 60
Gradient Boosting Regressor	11.6802	15.3718	-0.5349	53.68%	TP: 69 FP: 42 TN: 55 FN: 65

Simple Linear Regressor	1.5618	2.0688	0.9722	51.08%	TP: 83 FP: 62 TN: 35 FN: 51

Table 2: Regression Models without Technical Indicators

Model	MAE	RMSE	R² Score	Directional Accuracy	Confusion Matrix
Random Forest Regressor	9.5332	13.3086	0.1808	45.78	TP: 43 FP: 29 TN: 72 FN: 105
Prophet (Meta)	16.8988	20.1956	-0.8422	49.20%	TP: 76 FP: 55 TN: 47 FN: 72
XGBRegressor	9.9552	13.6746	0.1352	47.79%	TP: 42 FP: 24 TN: 77 FN: 106
Gradient Boosting Regressor	2.2326	2.8269	0.9630	51.00%	TP: 88 FP: 62 TN: 39 FN: 60
Simple Linear Regressor	9.3510	13.0489	0.2125	46.59%	TP: 41 FP: 26 TN: 75 FN: 107

5. Analysis

Classification Performance

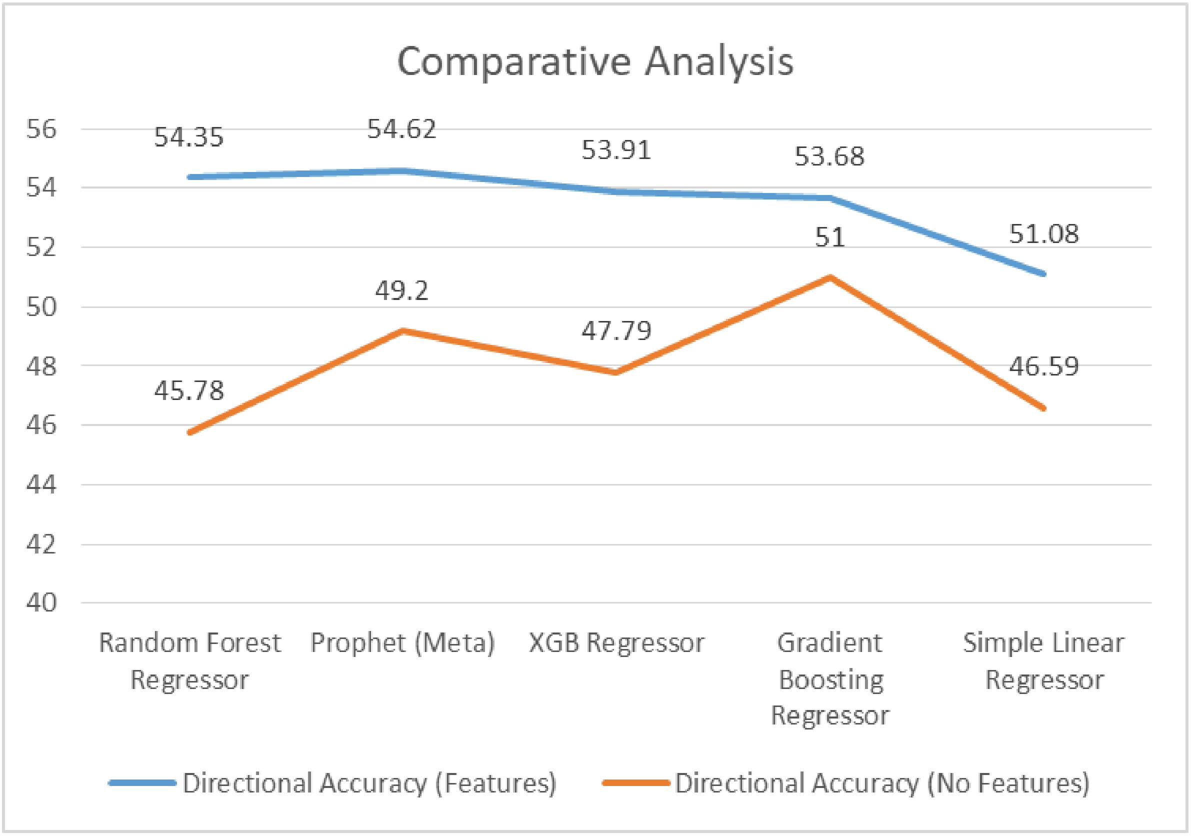


Figure 1: Directional Accuracy (%) across Features v/s No Features

The comparative analysis in Graph 1 shows the directional accuracy of five regression models- Random Forest, Prophet, XGBoost, Gradient Boosting, and Simple Linear Regression- across two scenarios: with engineered technical features (blue line) and with only closing price data (orange line).

Across all models, the inclusion of features yielded a better directional accuracy (Prophet being the highest (54.62%), slightly outperforming Random Forest (54.35%) as well as XGBRegressor

(53.91%), followed by Gradient Boosting Regressor (53.68%) and Simple Linear Regression (51.08%).

When using only closing prices, Gradient Boosting achieved the highest accuracy (51.00%), followed by Prophet (49.20%). However, the absence of features reduced performance significantly in most models, with XGBoost showing a drop from 53.91% to 47.79% and Random Forest from 54.35% to 45.78%.

Therefore, the use of features improves the directional accuracy of each of the models. Moreover, the directional accuracy without features is more volatile compared to models with added technical features, which depicts consistency in predictions when using additional features. The overall accuracy of this simulation remains lower than expected because of the limitations in the dataset, which was purposely designed to ensure fair and consistent conditions across all the models, truly capturing the impact of features of each model.

From Graph 1, the Random Forest regressor has the highest improvement in direction accuracy with additional features. Starting at its base model accuracy of 45.78% it reached one of the highest in the study of 54.35% through feature engineering. That is nearly a 10% jump from its baseline prediction result to its results when additional features have been added. This showcases knowing how to use feature engineering in prediction models can improve their results significantly, and proves how it is worth the time to add them.

Overall, Gradient Boosting Regressor remains the most consistent model across both simulation conditions (53.68% and 51%), followed by Prophet, which performs reasonably well across both setups (54.62% and 49.3%). This portrays the importance of features that depend on the model itself. Models arranged according to their significance on features (ascending) are as follows: Gradient Boosting Regressor (2.68%), Simple Linear Regression (4.49%), Prophet (Meta) (5.42%), XGBRegressor (6.12%), and Random Forest Regressor (8.57%).

Confusion Matrix Data

The confusion matrix, as shown in the last columns of tables 1 and 2, depicts that the overall prediction of true positives proves to be significantly better than the prediction of true negatives, which essentially means that the prediction of bullish trends is of substantially higher accuracy throughout. Although with features, the accuracy of bearish trends also increases significantly (remaining significantly lower than the bullish prediction throughout). This can be explained by 1-day dips that are often overcome within the same week, giving false bearish signals. As a result, the model predicts true-positives and false-negatives at a greater magnitude.

Regression Performance

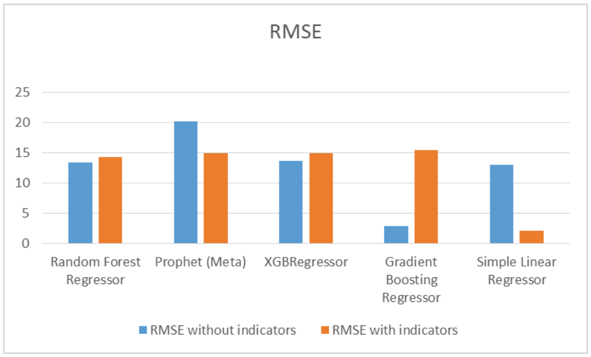
The R^2 Score proves to be ineffective in indicating the model's performance due to the noisy dataset and short-term predictions, which lack a clear pattern, rendering the R^2 score useless.

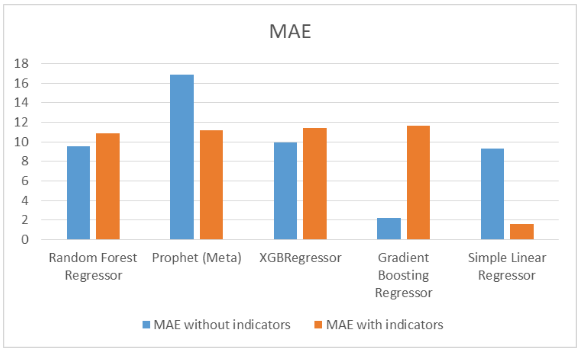
Previous research demonstrates that models with negative R^2 values can still provide value in price prediction. For instance, in one MDPI study, a KNN regressor achieved $R^2 = –2.42$, yet its

SMAPE of 14.32% was considered a valid indicator of forecasting ability. While it is helpful to see the variance in the dependent variable explained by the score, it is more important to know the context in which the regression is taking place (11).

Since the study is taking place in a short-term financial time series in our forecasts and contains too much volatility, which became the cause of the low or even negative R^2 score, with and without feature engineering, this occurs because the R^2 score is sensitive to minor deviations from the actual values. The presence of the R^2 score in our data can be misleading because of the poor scores, although our models prove to be improved, as evidenced by improved directional accuracy with added features.

RMSE & MAE





Figures 2 & 3: RMSE and MAE across all the models with and without indicators

The RMSE is slightly higher than the MAE throughout, as the RMSE is much higher with higher absolute error. This means that there were a fair number of instances where the Absolute Error was slightly higher than the RMSE.

With technical indicators (features), RMSE remained around $14. In contrast, the MAE remained at around $11 (except in simple linear regression, wherein the RMSE and MAE were at $2.0688 and $1.5688, respectively, due to its tendency to find a linear relationship with not only the closing price but also technical indicators, capturing underlying trends). With technical indicators, XGBRegressor and Random Forest Regressor, RMSE and MAE remain similar, Prophet is overshot significantly, which shows its dependence on features for making accurate predictions ($20.1956 and $16.8988, respectively).

Simple linear regression is also overshot as it aims to capture a linear relationship between the features and time. With only the closing price, this becomes inaccurate due to the lack of context of other features, which change with time. Gradient Boosting Regressor captures non-linear trends, which ultimately result in a significantly lower RMSE and MAE of 2.8269 and 2.2326, as evidenced by its consistent directional accuracy.

6. Conclusion

The overall results of the study showcase the impact of feature engineering and reflect the volatility of the stock market. The result of directional accuracy showed that there is a measurable impact of the addition of technical indicators overall, making a difference in the probability of a coin-flip chance. In the real world, this improvement in accuracy plays a substantial role in predicting stock market performance, especially in the short term, such as in “day trading.” The overall accuracy was at a lower band than expected, around 50%, which is essentially due to the future-like setup in the testing process (testing for a completely different year without any context). Moreover, the dataset was limited to ensure a controlled setup while evaluating the performance of models. This dataset allowed for the comparative study of the models throughout without any changes in the training/testing dataset, without any biases. The study also demonstrates that regressor models are worse off in predicting directional trends in the short run, despite predicting the data trends (not directional trends) well in the long run (as seen by the RMSE and MAE). This is also due to the limitations of predicting the stock market in the short term solely based on technical indicators, implying that news and sentiment often play a significant role in the movement of the stock market. Therefore, while technical indicators give an edge during predictions, they can not fully predict short-term trends without economic, financial, and sentimental context. This essentially portrays that a hybrid approach i.e. technical indicators + sentiment analysis approach may be necessary for meaningful short-term predictions.

7. References

[1] Rodriguez, F. S., P. Norouzzadeh, Z. Anwar, E. Snir, and B. Rahmani. “A Machine Learning Approach to Predict the S&P 500 Absolute Percent Change.” Discover Artificial Intelligence, vol. 4, no. 1, 2024, https://doi.org/10.1007/s44163-024-00104-9. Accessed 10 July 2025
https://doi.org/10.1007/s44163-024-00104-9

[2] Montgomery, Douglas. “Introduction to Linear Regression Analysis.” *Journal of the American Statistical Association*, vol. 88, no. 421, 1993, p. 383. https://doi.org/10.2307/2290746. Accessed 26 July 2025
https://doi.org/10.2307/2290746

[3] Poole, Michael A., and Patrick N. O’Farrell. “The Assumptions of the Linear Regression Model.” Transactions of the Institute of British Geographers, no. 52, 1971, p. 145. https://doi.org/10.2307/621706. Accessed 27 July 2025.
https://doi.org/10.2307/621706

[4] Biau G., Cadre B., and Rouvière L. “Accelerated Gradient Boosting.” Machine Learning, vol. 108, no. 6, 2019, pp. 971–92. https://doi.org/10.1007/s10994-019-05787-1. Accessed 2 August 2025
https://doi.org/10.1007/s10994-019-05787-1

[5] Natekin, Alexey, and Alois Knoll. “Gradient Boosting Machines, a Tutorial.” Frontiers in Neurorobotics, vol. 7, 2013, https://doi.org/10.3389/fnbot.2013.00021. Accessed 26 July 2025
https://doi.org/10.3389/fnbot.2013.00021

[6] Rigatti, Steven J. “Random Forest.” *Journal of Insurance Medicine*, vol. 47, no. 1, 2017, pp. 31–39. https://doi.org/10.17849/insm-47-01-31-39.1. Accessed 10 July 2025
https://doi.org/10.17849/insm-47-01-31-39.1

[7] Speiser L. Jaime, Michael E. Miller, Janet Tooze, and Edward Ip. “A Comparison of Random Forest Variable Selection Methods for Classification Prediction Modeling.” *Expert Systems With Applications*, vol. 134, 2019, pp. 93–101. https://doi.org/10.1016/j.eswa.2019.05.028. Accessed 16 July 2025
https://doi.org/10.1016/j.eswa.2019.05.028

[8] Svetnik, Vladimir, Andy Liaw, Christopher Tong, J. Christopher Culberson, Robert P. Sheridan, and Bradley P. Feuston. “Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling.” *Journal of Chemical Information and Computer Sciences*, vol. 43, no. 6, 2003, pp. 1947–58. https://doi.org/10.1021/ci034160g. Accessed 26 July 2025
https://doi.org/10.1021/ci034160g

[9] Taylor, Sean J., and Benjamin Letham. “Forecasting at Scale.” *The American Statistician*, vol. 72, no. 1, 2017, pp. 37–45. https://doi.org/10.1080/00031305.2017.1380080. Accessed 20 July 2025
https://doi.org/10.1080/00031305.2017.1380080

[10] Shen, Justin, Davesh Valagolam, and Serena McCalla. “Prophet Forecasting Model: A Machine Learning Approach to Predict the Concentration of Air Pollutants (PM2.5, PM10, O3, NO2, SO2, CO) in Seoul, South Korea.” *PeerJ*, vol. 8, 2020, p. e9961. https://doi.org/10.7717/peerj.9961. Accessed 20 July 2025
https://doi.org/10.7717/peerj.9961

[11] Sonkavde, Gaurang, Deepak Sudhakar Dharrao, Anupkumar M. Bongale, Sarika T. Deokate, Deepak Doreswamy, and Subraya Krishna Bhat. “Forecasting Stock Market Prices Using Machine Learning and Deep Learning Models: A Systematic Review, Performance Analysis and Discussion of Implications.” *International Journal of Financial Studies*, vol. 11, no. 3, 2023, p. 94. https://doi.org/10.3390/ijfs11030094. Accessed 4 August 2025
https://doi.org/10.3390/ijfs11030094

[12] Santhanam, Ramraj, Nishant Uzir, Sunil Raman, and Shatadeep Banerjee. “Experimenting with XGBoost Algorithm for Prediction and Classification of Different Datasets.” ResearchGate, 2017, www.researchgate.net/publication/318132203_Experimenting_XGBoost_Algorithm_for_Prediction_and_Classification_of_Different_Datasets. Accessed 25 July 2025
www.researchgate.net/publication/318132203_Experimenting_XGBoost_Algorithm_for_Prediction_and_Classification_of_Different_Datasets