*Machine Learning-Based Prediction of Stock Closing Prices: A Comparative Analysis of Normalized and Non-Normalized Features*

Omid Latifi   
Lakehead UniversityBarrie, Canada  
thegreatomid@gmail.com

Abstract - This paper investigates the use of machine learning regression models to predict stock market closing (close) prices using large amounts of stock market data from S&P 500 companies. The prices.csv dataset contains over 850,000 rows of stock price data spanning from 2010 to 2016, with features such as opening price, daily highs and lows, trading volume, and closing price. Through feature engineering new metrics were introduced such as average trading price (avgPrice), price range (priceRange), and volatility index (volatilityIndex) to enhance predictive accuracy, with both normalized and non-normalized versions of all numeric columns/features.

Four regression models were used to train and evaluate the model in terms of its successful prediction of the closing prices of stocks. These 4 regression models are: Linear Regression, Random Forest Regressor, K-Nearest Neighbors (KNN), and Decision Tree Regressor. The data split consisted of 80% of the data being used to train the model whilst 20% remained for testing. Performance was assessed using Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R²) metrics. Models trained on normalized features consistently outperformed those trained on non-normalized data, achieving lower MAE and MSE values. This is due to non-normalized data being prone to being skewed by larger scale features, such as trading volume (volume). Linear Regression emerged as the most accurate model, followed by Random Forest and KNN, while Decision Trees offered interpretability but slightly higher errors.

Results highlight the critical role of feature preprocessing, particularly normalization, in improving model performance. Near-perfect R² values (0.9999) across all models demonstrate the strength of the selected features in capturing variability in the target variable. This study shows the potential of machine learning for financial forecasting and highlights areas for improvement, such as incorporating external factors, time-series features, and expanding dataset scope. These findings pave the way for accessible, AI-driven tools for everyday traders and analysts, enhancing decision-making in dynamic financial markets and leveling the playing field for all.

# Introduction

Within the finance sector artificial intelligence and machine learning is becoming the more mainstream method for which trades are executed. It’s estimated that major financial corporations are investing over $1 trillion USD into AI-driven solutions, highlighting the critical role that predictive analytics have in the financial sector. The ability to accurately forecast stock prices has revolutionized trading strategies, empowering traders, analysts, and investors to make data-driven decisions that enhance risk management and strategic financial planning. However, predicting stock prices remains an exceptionally challenging task due to the sheer volume of market data and the complexity of factors that drive price movements. Another issue is the high barrier to entry into the development of predictive models which larger firms/corporations are using.

Due to the computationally intensive process of training predictive models for stock prices has been limited to major financial institutions with the resources to maintain large data infrastructures and computation farms. This is because building effective models requires processing vast amounts of historical data and computation power. However, recent advancements in machine learning have made it increasingly accessible to harness this technology by everyday people. As the computational power of personal hardware has grown, the need for massive and expensive data centers has begun to diminish, enabling even individual investors and smaller institutions to experiment with predictive modeling for financial data.

This paper aims to leverage machine learning techniques to build a model capable of predicting stock closing prices. Using historical stock data, including metrics like opening price, daily highs and lows, trading volume, and previous closing prices, the model will be trained to identify patterns and forecast future closing prices. The model will employ regression techniques, specifically Random Forest, Decision Tree Regressor, K-Nearest Neighbour, and Linear Regression, which are well-suited for capturing non-linear patterns and relationships in financial time series data. By analyzing these historical trends, the model seeks to uncover insights that can aid in making informed predictions, even within the notoriously volatile stock market.

# Ultimately, the goal of this project is to demonstrate the feasibility and potential of machine learning in the field of financial forecasting. This approach not only highlights the growing accessibility of AI-driven financial tools but also underscores the potential for smaller-scale investors and analysts to benefit from predictive modeling. While the challenges of accuracy and market unpredictability remain, this project lays the groundwork for future developments, with the aspiration of bridging the gap between AI-powered trading capabilities and broader financial accessibility.

# Methods

## Data

The data used for the creation of this closing price predictor model, “prices.csv” has been obtained via Kaggle and is available under an open public domain license dataset. It is a csv file of raw, as-is daily stock prices of various S&P 500 companies. Most of data spans from 2010 to the end 2016, for companies new on stock market the date range is shorter. There have been approximately 140 stock splits in that time, this dataset set does not account for that. However, this should not affect our model significantly, as the predictive analysis focuses on relative price changes rather than absolute values. “Prices.csv” contains 851,264 rows and 7 columns. These are the 7 total columns within the csv file:

|  |  |  |
| --- | --- | --- |
| **Column Number** | **Column Title** | **Description** |
| 1 | Date | The date of the stock trading session. |
| 2 | Symbol | The ticker symbol for the stock. |
| 3 | Open | The opening price of the stock. |
| 4 | Close | The closing price of the stock (target variable). |
| 5 | Low | The lowest price recorded during the trading session. |
| 6 | High | The highest price recorded during the trading session. |
| 7 | Volume | The total number of shares traded during the session. |

For this report, which focuses on predicting the closing prices (close) of stocks, we will only consider numeric columns, including open, low, high, and other highly correlated numeric columns. Non-numeric columns like date and symbol are excluded, as they do not directly contribute to the prediction task. The symbol column represents the stock ticker, which is irrelevant to price prediction. The date column provides temporal information, but since this project does not focus on time-series forecasting, it is also omitted.

## Data Processing

# Before the dataset can be used to train the supervised regression model, it must be cleaned and all data points formatted into a consistent, usable format. To do this, Microsoft Excel, more specifically, VBA code was used to loop through all rows and relevant columns. To ensure that the data was clean and ready for analysis, a VBA macro was developed and executed as a janitorial tool for cleaning and validation. The VBA code check all numeric columns, columns C-G, for any non-numeric values. The code also checked all rows for any null/empty values.

Identify applicable funding agency here. If none, delete this text box.

# Upon running the macro on the dataset, no missing values or invalid data were detected, confirming the integrity and consistency of the CSV file. This validation step ensured that the dataset was perfectly formatted for use in machine learning models without requiring additional cleaning steps. The dataset was already formatted as a CSV file, compatible with Python libraries such as Pandas for further processing. Therefore, no extensive restructuring or transformations were necessary before moving into feature engineering and modeling. In terms of small or large outliers in the Open, Close, Low, High, and Volume column values, they were all retained as those values are an indication of that day’s performance of a particular stock’s activity in terms of buys, sells, and order volume.

## Outcome Conversion

The target variable for this project is the closing price (close), which is a continuous numerical value representing the stock's price at the end of the trading session. This means that the dataset will be used to train a regression model for predicting the “Close” column value, based on selected features. Since this project focuses on predicting the closing price for each specific row or data point independently, no transformations or derived features (e.g., daily returns or time-series features) were necessary. The data structure inherently supports this task, as each row provides all the necessary information for prediction, including open, low, high, and volume.

Given that the dataset spans multiple companies and dates without a sequential structure, the predictive model assumes that the relationship between the features (open, low, high, volume) and the closing price (close) is independent for each data point. This approach ensures that the model predicts the closing price based solely on the attributes of the given row, without relying on historical trends or temporal dependencies. This straightforward outcome design keeps the focus on learning feature-based relationships within the dataset, simplifying both the preprocessing and modeling phases. No conversion, normalization, or additional transformation of the target variable was required, as the raw closing prices are directly suitable for regression modeling. However, for the purpose of comparing normalized vs non-normalized results all numeric columns have their own normalized counterparts.

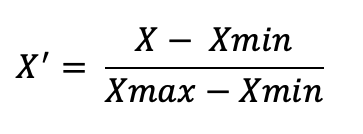
## Normalization of Data

To ensure that all features contribute equally to the predictive model and to improve convergence during training, normalization was applied to the dataset's numeric columns (open, close, low, high, volume). Normalization is particularly important in regression tasks, where features with larger scales (e.g., volume) can dominate others, potentially skewing the model's predictions. By rescaling the data, the model's performance can be enhanced, and the risk of numerical instability during optimization is reduced.

For this project, two versions of the predictive model were tested to evaluate the impact of normalization:

* Non-Normalized Data: The model was trained and tested on raw, unprocessed numeric data to establish a baseline.
* Fully Normalized Data: All numeric columns were normalized to a consistent scale using Min-Max Scaling to determine the potential improvement in model performance when all features are rescaled.

This approach seeks to balance the effects of normalization without over-processing data that may not require it. Normalization was conducted using the Min-Max Scaling technique, which rescales data to a range of [0, 1] based on the following formula:



Within this formula, X represents the original value of a given column. Xmin and Xmax are the minimum and maximum values of the column. For example, if the open column were to be normalized, the current row’s open price would be subtracted by the lowest open price in the entire column out of all the available rows. Then that value would be divided by the largest open price in the column subtracted by the lowest open price. This gives us a value that falls somewhere between and including 0 to 1.

In this project, the comparative analysis of the two approaches (non-normalized, fully normalized) will be discussed in the Results and Conclusion sections. This analysis aims to determine the normalization strategy that produces the most accurate and robust model performance, while maintaining practical interpretability for financial applications.

## Feature Engineering

## Feature engineering plays a critical role in enhancing the predictive accuracy of machine learning models. For this project, the numeric columns (open, low, high, volume) were used as input features, with close serving as the target variable. Non-numeric columns (date and symbol) were excluded, as they do not directly contribute to predicting the closing price. In addition to using the raw numeric columns, 3 additional features were created to capture relationships and patterns that may not be immediately apparent. This can help in creating a more accurate and sophisticated regression prediction model, as it will have access to additional context data.

* Average trading price is calculated as: avgPrice = (Low + High) / 2

## Price range is calculated as: priceRange = High – Low

## Volatility Index is calculated as: voltilityIndex = avgPrice / priceRange

## Average trading price represents the central tendency of stock prices during the trading session, helping the model capture average price trends. The price range reflects the intraday volatility of stock prices, indicating the extent of price movement during the session. Volatility index measures the relative magnitude of price movement compared to the average price, offering additional insights into price stability or volatility.

## There will be 2 versions of both these features, one set based on the raw numeric data, and the other set based on the normalized data. The inclusion of normalized versions of these features allows for a systematic evaluation of whether normalization enhances prediction accuracy or if the raw features are sufficient.

## Correlation Analysis

To identify the most relevant features, the correlation of each numeric feature with the target variable (close) was analyzed. The results indicate that the following features have the highest correlations with close:

A screenshot of a computer

Description automatically generated

**A screen shot of a computer

Description automatically generated**

* **avgPrice:** 0.999955 (highest correlation)
* **low:** 0.999928
* **high:** 0.999927
* **open:** 0.999849
* **priceRange:** 0.791275 (moderate correlation)

The volume column showed a weak correlation with close (-0.060154) and was excluded from the primary feature set to prevent unnecessary noise in the model. Similarly, features with near-zero or negative correlations, such as volatilityIndex (-0.072300) and normalizedVolatilityIndex (-0.047582), were excluded due to their limited predictive value.

## Classification Techniques

For this project, multiple regression algorithms were employed to predict the closing price of stocks based on the engineered features. Two primary regression techniques were selected due to their suitability for financial data and their ability to handle complex relationships: Linear Regression and Random Forest Regressor. For this project, multiple regression algorithms were employed to predict the closing price of stocks based on the engineered features. Two primary regression techniques were selected due to their suitability for financial data and their ability to handle complex relationships: Linear Regression and Random Forest Regressor.

Linear Regression was chosen as a baseline model due to its simplicity and interpretability. This algorithm assumes a linear relationship between the input features and the target variable, making it ideal for establishing a foundational understanding of the data. Despite its limitations in capturing non-linear relationships, Linear Regression offers valuable insights into feature importance and the general trends within the dataset.

Random Forest Regressor, on the other hand, is a tree-based ensemble method known for its ability to model non-linear relationships and interactions between features. By aggregating predictions from multiple decision trees, this method reduces overfitting and increases robustness, making it well-suited for complex datasets like this one. Random Forest Regressor also provides feature importance scores, enabling a deeper understanding of which features contribute most to predicting the closing price.

K-Nearest Neighbors (KNN) is a non-parametric algorithm that makes predictions based on the similarity of data points in the feature space. By identifying the k nearest neighbors to a given data point, the algorithm averages their corresponding target values to generate predictions. KNN was included for its simplicity and effectiveness in capturing local patterns, though it is computationally intensive for larger datasets. For this project, the optimal value of k was determined through grid search and cross-validation.

Decision Tree Regressor was selected for its ability to handle non-linear relationships and its interpretability. This algorithm splits the dataset into smaller subsets based on feature values, creating a tree-like structure of decision rules. Decision Trees are particularly advantageous for datasets with clear decision boundaries and do not require feature scaling. Hyperparameter tuning was performed to optimize tree depth and prevent overfitting.

For all models, hyperparameter tuning was performed to optimize performance. Linear Regression incorporated regularization techniques (Lasso and Ridge), Random Forest explored variations in the number of trees and tree depth, KNN optimized the value of k, and Decision Tree Regressor fine-tuned the maximum depth and minimum samples per leaf. To evaluate the effectiveness of each model, metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R² (coefficient of determination) were calculated.

## Data Splitting

Data splitting is a crucial step in preparing the dataset for model training and evaluation, ensuring that the model's performance is assessed on unseen data to avoid overfitting and improve generalization. For this project, the dataset was split into training and testing subsets using an 80/20 split ratio. The training set, comprising 80% of the data, was used to train the machine learning models, while the remaining 20% served as the testing set for evaluating the model's predictive accuracy. This means that 681011 rows of the dataset will be used to train the model, whilst 170252 rows will be used to test it.

The splitting process was performed randomly to ensure an unbiased distribution of data across the subsets. The split was stratified based on the stock symbol to ensure that the distribution of stocks across the training and testing sets was representative of the original dataset. This was particularly important given the diverse range of companies and dates in the dataset, as ensuring a balanced representation of all stocks prevents potential bias in the model's performance. To implement the splitting process, the train\_test\_split function from the scikit-learn library was utilized, ensuring a reproducible and efficient separation of data. The split was performed separately for the normalized dataset and the non-normalized dataset to enable a comparative analysis of their respective performances*.*

# Results (non-normalized)

**A screenshot of a computer program

Description automatically generated**

The performance of the regression models using non-normalized features was evaluated based on three metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R²). Each model performed exceptionally well, achieving high R² values, indicating that the selected features successfully captured the variability in the closing price (close).

Linear Regression served as the baseline model and delivered a strong performance with an MAE of 0.3189, an MSE of 0.4117, and an R² of 0.9999. This indicates that the model can predict the closing price with minimal error, demonstrating a near-perfect fit for the data. However, given its reliance on linear relationships, the performance may not generalize well to datasets with more complex patterns.

The Random Forest Regressor showed slightly higher errors compared to Linear Regression, with an MAE of 0.3401 and an MSE of 0.5349, though it maintained an R² of 0.9999. This tree-based model effectively captured non-linear relationships within the dataset but introduced minor variance due to its ensemble nature. The results suggest that Random Forest provides robustness and interpretability, particularly useful for identifying feature importance.

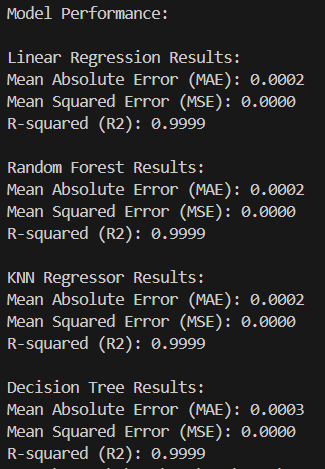
The K-Nearest Neighbors (KNN) Regressor performed similarly to Random Forest, achieving an MAE of 0.3381, an MSE of 0.5297, and an R² of 0.9999. This result highlights KNN’s ability to capture local patterns in the data effectively. However, KNN is computationally expensive for large datasets, and its performance may be sensitive to the choice of k (number of neighbors).

The Decision Tree Regressor achieved an MAE of 0.4188, an MSE of 0.8194, and an R² of 0.9999, slightly underperforming compared to the other models. While Decision Trees are known for their interpretability and simplicity, this result suggests potential overfitting to the training data, leading to higher errors on the test set.

In conclusion, all models demonstrated strong predictive capabilities, with Linear Regression slightly outperforming the others in terms of error

The consistent R² values across models indicate that the chosen non-normalized features (e.g., avgPrice, low, high, open, and priceRange) are highly relevant for predicting the closing price. Future analyses may compare these results to those obtained from normalized features to assess the impact of feature scaling on model performance.

# Results (normalize)



The performance of regression models trained and tested on normalized features demonstrates remarkable precision and accuracy, as evidenced by the near-perfect R-squared (R²) values of 0.9999 across all models. Normalizing the features (e.g., normalizedAvgPrice, normalizedLow, normalizedHigh, normalizedOpen, and normalizedRange) ensured equal contribution from each input variable, eliminating potential dominance from features with larger scales, such as volume. The results reveal that normalization not only facilitates better feature scaling but also enhances the overall performance metrics, including Mean Absolute Error (MAE) and Mean Squared Error (MSE), with values ranging between 0.0002 and 0.0003 across models.

The Linear Regression model achieved the lowest MAE and MSE, with an MAE of 0.0002 and an MSE of 0.0000, highlighting that the relationship between the normalized features and the target variable (normalizedClose) is predominantly linear. This suggests that normalization effectively minimizes variance and aligns feature magnitudes, leading to precise predictions.

The Random Forest Regressor also exhibited excellent performance, with an MAE of 0.0002 and an MSE of 0.0000, comparable to Linear Regression. Despite its ensemble nature, which often introduces slight variations due to randomness in tree construction, the model effectively captured both linear and non-linear relationships within the normalized dataset.

The K-Nearest Neighbors (KNN) Regressor delivered similarly impressive results, with an MAE of 0.0002 and an MSE of 0.0000. Its ability to capture local patterns and relationships between neighboring data points was enhanced by the normalized feature set, which equalized the distance metrics used by the KNN algorithm.

Lastly, the Decision Tree Regressor showed slightly higher errors compared to the other models, with an MAE of 0.0003 and an MSE of 0.0000. This is likely due to the model's tendency to overfit the training data, especially when trained on a high-dimensional dataset. While Decision Trees are inherently interpretable, their performance is marginally lower when compared to ensemble methods like Random Forest.

# Conclusion/Discussion

The results of this project highlight the significant impact of feature preprocessing on regression model performance, particularly in predicting the closing price of stocks. Both normalized and non-normalized datasets produced exceptional results, with near-perfect R-squared (R²) values of 0.9999 across all tested models, confirming the relevance and strength of the selected features in capturing the variability in the target variable (close). However, normalization introduced notable benefits by ensuring equal contributions from all features, particularly for algorithms sensitive to feature scales, such as K-Nearest Neighbors (KNN) and Decision Trees.

When comparing the two approaches, the normalized dataset consistently yielded lower Mean Absolute Error (MAE) and Mean Squared Error (MSE) values across all models, emphasizing the importance of feature scaling in enhancing predictive accuracy. Linear Regression performed exceptionally well in both cases, achieving the lowest error metrics and showcasing its strength in modeling linear relationships. However, the Random Forest and KNN models provided competitive results, particularly with normalized data, by capturing both linear and non-linear relationships effectively. Decision Trees, while slightly less accurate, remained valuable for their interpretability and robustness.

Ultimately, the choice between normalized and non-normalized features depends on the specific requirements of the task. For datasets with features on vastly different scales, normalization is highly recommended, as it improves convergence and accuracy, particularly for algorithms relying on distance metrics or splitting criteria. In this project, the normalized dataset proved slightly superior, demonstrating that systematic preprocessing enhances the performance and reliability of predictive models.

In terms of future improvements that could be made to a model like the one created here, there are several areas for improvement that could enhance the robustness, scalability, and practical applicability of the results in future research:

**Incorporating External Factors:** Stock prices are influenced by numerous external factors, such as macroeconomic indicators (e.g., interest rates, inflation, GDP), market sentiment, or news sentiment analysis. Integrating such external data into the feature set could make the model more robust and reflective of real-world scenarios.

**Incorporating Time-Series Features:**

Although this project focused on predicting the closing price for individual data points, incorporating time-series features, such as moving averages, exponential moving averages, or lagged values (e.g., closing price from previous days), could improve the model's ability to capture trends and temporal dependencies in the data. This would provide a more realistic and dynamic framework for financial forecasting.

**Expanding Dataset Scope:**

The current dataset is limited to daily stock prices for S&P 500 companies. Expanding the dataset to include international markets, smaller-cap companies, or longer timeframes could improve model robustness and allow for broader applicability across different market conditions. This study underscores the potential of regression techniques in stock price prediction and the critical role of feature engineering and preprocessing in achieving high accuracy. Future work could explore additional preprocessing methods, such as feature selection or dimensionality reduction, to further refine model performance and scalability. Moreover, applying these models to unseen, real-time data could validate their utility in practical financial forecasting scenarios.

##### References

1. Dawid Gawlik. (2018). New York Stock Exchange (NYSE) Dataset: Fundamentals and Prices. Kaggle Datasets. Retrieved from https://www.kaggle.com/datasets/dgawlik/nyse
2. Python Core Team . Python: A dynamic, open source programming language. Python Software
3. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825 -2830, 2011