Predicting Stock Trends with Binary Classification

1st Tony Gonzalez
Cal Poly Pomona
CS4440 - Data Mining
Pomona, California
tonygonzalez@cpp.edu

2nd Kaitlin Yen
Cal Poly Pomona
CS4440 - Data Mining
Pomona, California
kaitlinyen@cpp.edu

3rd Joseline Ly
Cal Poly Pomona
CS4440 - Data Mining
Pomona, California
joselinely@cpp.edu

4th Milosz Kryzia
Cal Poly Pomona
CS4440 - Data Mining
Pomona, California
mkryzia@cpp.edu

5th Armin Erika Polanco
Cal Poly Pomona
CS4440 - Data Mining
Pomona, California
aopolanco@cpp.edu

***Abstract*—This study addresses the problem of stock price movement prediction by developing a binary classification model. The original objective was to accurately predict whether a given stock's closing price would rise or fall on the following trading day. However, based on initial model analysis, the team pivoted to predicting high versus low stock volatility for the next trading day of the S&P 500. The inherently high stakes of the financial market make this a compelling challenge, particularly for students interested in applying data science techniques to time-series analysis.**

To solve this problem, the yfinance Python library was used to collect and process several years of historical data from Yahoo Finance. This extensive dataset was then used to train a Long Short-Term Memory (LSTM) neural network. The model employs a sigmoid activation function on the output layer, which yields a probability that directly represents the model's confidence in a stock's high or low volatility. The results of this work offer insight into the predictive capacity of deep learning models in the volatile financial domain.

***Index Terms*—binary classification, machine learning, stock prediction, S&P index, yfinance, Python, LSTM.**

I. Introduction

Stock market prediction is a complex problem due to the erratic and nonlinear nature of financial time-series data. This project addresses the task of predicting the volatility of the S&P 500 index as a binary classification problem. Our model will determine whether a stock's volatility is expected to be high or low on the basis of specific features. The motivation behind this study is to explore how data mining methods can effectively capture hidden patterns and dependencies in stock market data to improve forecast accuracy.

The main method used in this study is Long Short-Term Memory (LSTM) networks, which are particularly important for modeling sequential data and reducing information loss over time. An LSTM neural network is a subset of machine learning models called Recurrent Neural Networks (RNNs) [1]. Like other neural networks, they have multiple layers consisting of weights and biases to predict outputs. The main difference between RNNs and other neural networks is that RNNs reuse the same weights and biases of previous layers throughout the entire network for multiple inputs, allowing them to handle input sequences of varying lengths. General neural networks are restricted to the same number of inputs. An example illustrating the strength of an RNN is looking at the history of different stock prices, where one stock may have noticeably more history than another. A general neural network would be unable to process these data unless the histories were truncated to be of the same length.

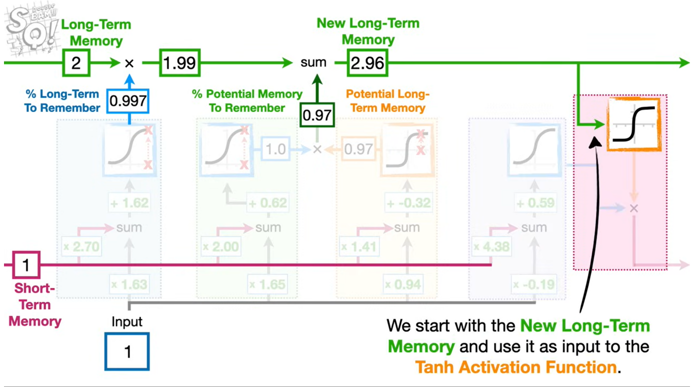

Fig. 1. Example of a Long Short-Term Memory Model

An LSTM model [2] takes the RNN adaptability concept further by solving the Exploding/Vanishing Gradient Problem. This issue occurs when an RNN applies the same weights to multiple unfolded layers, causing the weights to become too large or too small. Consequently, the model will either overestimate or underestimate a prediction by the end of its calculation. LSTM addresses this by using two paths: a long-term memory (cell state) and a short-term memory (hidden state). This allows the model to keep track of recent patterns over a time-series dataset while simultaneously observing the overall longer-term trends. An LSTM unit contains multiple internal gates (the forget gate, input gate, and output gate) which determine how much long-term and short-term memory the model should retain, with these decisions being affected by the layer's weights and biases.

II. Dataset

The dataset is obtained using `yfinance` [3], a Python data retrieval tool that scrapes the Yahoo Finance website [4]. The `yfinance` library can request live price data for a multitude of stock options and stores this information in a pandas DataFrame for use. Daily data from 2015 to 2025 was collected, including the Open, High, Low, Close, and Volume (OHLCV) features. This extensive historical data is preprocessed and used to train and evaluate predictive models for short-term stock market trend analysis. The dataset's size depends on the number of stock tickers to classify, time range (1 month vs 10 years), and how often `yfinance` is queried (every minute, hour, or day).

`yfinance` can provide the following information [5]:

- Historical Market Data: open, high, low, close, and adjusted close prices, and number of shares traded over a specified time period.
- Corporate Actions: stock splits and other events that can impact stock prices.
- Financial Statements: balance sheets, income statements and cash flow statements.
- Meta Data: earnings dates, revenue and earnings per share (EPS) which are key factors of a company's performance. Multiple Tickers: multiple stocks can be simultaneously tracked.

An example of `yfinance`'s dataset for Apple (APPL) is shown below:

	Price				
Date	**Close**	**High**	**Low**	**Open**	**Volume**
2020-01-02	72.5380	72.5990	71.2920	71.5460	135480400
2020-01-03	71.8330	72.5940	71.6090	71.7660	146322800
2020-01-06	72.4060	72.4440	70.7030	70.9540	118387200
2020-01-07	72.0650	72.6710	71.8450	72.4150	108872000
2020-01-08	73.2240	73.5260	71.7680	71.7680	132079200

The dataset spans from January 2015 to December 2024 and includes the OHLCV columns for each trading day, totaling to 2518 rows and 217 KB.

III. Methodology

A. Data Preprocessing

Before modeling, any missing values are removed and all features are scaled to the range [0, 1] using a Min–Max scaler to improve the convergence of the neural network.

To transform the problem into a binary classification task, a target variable is created to indicate whether the next day's stock price exhibits high or low volatility. Volatility is calculated using the difference between the next day's high and low prices (the trading range) and compared against a defined historical threshold. The target is assigned a value of 1 for high volatility and 0 for low volatility. A 30-day sliding window of historical data is used as the input to predict this binary volatility outcome for the next day.

B. Scope Adjustment

As previously mentioned, the original objective was to accurately predict whether a given stock's closing price would rise or fall on the following trading day. The initial binary classification model would employ a sigmoid activation function on the output layer, yielding a probability that directly represents the model's confidence in an increase in the next day's closing price. However, the resulting initial model did not accurately and consistently predict the next trading day's direction – at least with the current dataset. According to the dataset's feature analysis, the correlation signals were too weak to be able to predict the next day's direction, which is reasonable given the market's "random walk" nature. The original model usually defaulted to predicting only one class and could not find any distinction.

To adapt for a more consistent model, the scope is changed to predicting high/low stock market volatility instead. This method yields more accurate results as:

- Volatility prediction shows 7.2x stronger correlation signals compared to direction prediction.
- Maximum feature correlation: 0.29 (compared to $\sim$0.04 for direction prediction).
- It demonstrates strong volatility clustering behavior (high volatility tends to follow high volatility).

C. Feature Engineering

The goal of this step is to find which features affect next-day market volatility the most. Given that volatility is defined as the absolute return of the next day:

$$\text{volatility}_{t+1} = \left| \frac{\text{Close}_{t+1} - \text{Close}_t}{\text{Close}_t} \right| \tag{1}$$

Where a value of 1 will be volatility above the median and a value of 0 is volatility below the median. A median volatility threshold is chosen for a balanced classification problem ($\sim$50, $\sim$50 split).

Given the Open-Close, High-Low features, features are extrapolated according to direction, price, volatility, strength, and other indexes to feed into the LSTM model. Features include RSI (Relative Strength Index), MACD (Trend Momentum Indicator), Bollinger Bands (Volatility Indicators), etc.

Using correlation-based feature selection, the top 18 features are selected for LSTM feature inputs. The features are:

Feature Name	**Description**
Target	1 = next-day volatility above median, 0 = below
High_Low_Range	Intraday volatility: (High Low) relative to price
Open_Close_Range	Price change from open to close
SMA_5, SMA_10, SMA_20, SMA_50	Simple moving averages over different windows
EMA_12, EMA_26	Exponential moving averages (react faster to price changes)
Price_SMA5_Ratio	% difference between price and SMA(5)
Price_SMA20_Ratio	% difference between price and SMA(20)
SMA5_SMA20_Ratio	Short MA relative to long MA (trend signal)
Momentum_5, Momentum_10, Momentum_20	Price momentum over 5/10/20 days
ROC_5, ROC_10	Rate of change of price (momentum strength)
RSI_14	Overbought/oversold momentum indicator
MACD	Difference between short- and long-term EMAs
MACD_Signal	Smoothed version of MACD
MACD_Histogram	MACD Signal (trend momentum)
BB_Width	Width of bands = volatility level
BB_Position	Price location inside the bands
Volatility_5, Volatility_10, Volatility_20	Rolling standard deviation (5/10/20 days)
ATR	True range of price moves (includes gaps)
ADX	Strength of trend (not direction)
Plus_DI	Upward movement strength
Minus_DI	Downward movement strength
Price_Position_5	Where price sits in last 5-day high/low range
Price_Position_20	Same but over 20-day range
Volume	Trading volume
Volume_Change	% change in volume
Volume_SMA_20	20-day average volume
Volume_Ratio	Volume / Volume_SMA_20
OBV	On-balance volume cumulative indicator
OBV_Change	% change in OBV
VWAP	20-day volume-weighted average price
Price_VWAP_Ratio	Price relative to VWAP
Skewness_10	Skew of 10-day return distribution
Kurtosis_10	Tail risk (fat tails/excess kurtosis)
Trend_Strength_20	Absolute distance from SMA(20)
Trend_Strength_50	Absolute distance from SMA(50)

Top 10 features are:

1) High_Low_Range (+0.44)

2) Volatility_5 (+0.35)
3) Volatility_10 (+0.32)
4) ATR (+0.31)
5) Volatility_20 (+0.29)
6) Volume (+0.28)
7) RSI_14 (-0.27)
8) Minus_DI (+0.26)
9) Price_Position_20 (-0.26)
10) BB_Position (-0.23)

D. Model Architecture

As previously mentioned, the team utilized a Long Short-Term Memory (LSTM) neural network as it was able to effectively capture temporal dependencies in stock market movements. The model consists of two LSTM layers with 50 and 100 hidden units respectively, each followed by a dropout layer with a rate of 0.2 to prevent overfitting. A fully connected dense layer with a sigmoid activation function outputs a probability representing the likelihood that the next day's market will be high or low volatility.

The network is trained using the binary cross-entropy loss function, which is ideal for binary classification tasks, and optimized with the Adam optimizer. Training is performed for a fixed number of epochs with early stopping based on validation accuracy to further mitigate overfitting.

E. Training and Evaluation

The dataset is divided chronologically to preserve the temporal structure inherent to financial time-series data. We used data from 2015-2022 for training, 2023 for validation, and 2024-2025 for testing. This ensures that the model is evaluated on unseen future data, simulating real-world forecasting conditions.

Model performance evaluation uses the following classification metrics: accuracy, precision, and recall. Accuracy provides an overall measure of correct predictions, while precision and recall help assess the model's ability to correctly identify positive (high volatility) and negative (low volatility) cases. The combination of these metrics offers a balanced and comprehensive view of the model's performance in the context of financial time-series prediction, particularly in handling the inherent class imbalance or skewed priorities often found in market forecasting.

IV. Results

With the predicted classifications, the model's performance on the test set is summarized by the following classification metrics:

TABLE I
Model Performance Metrics on Test Data

Metric	**Score** (%)
Accuracy	59.46
Precision	62.12
Recall	38.68
F1-Score	47.67
ROC-AUC	62.86

Overall, the model achieved an accuracy of 59.46% and an F1-score of 47.67%.

A closer look at the class-specific performance reveals that the model was significantly better at identifying **low volatility** days. This is demonstrated by the recall score of 78% for low volatility classifications in comparison to the 39% recall achieved for high volatility days.

Additionally, the ROC-AUC (Receiver Operating Characteristic – Area Under the Curve) score of approximately 63% indicates that the model has a higher predictive power than random chance (50%). The ROC-AUC represents the trade-off between the True Positive Rate and the False Positive Rate, suggesting that the model successfully distinguishes between the two classes (high vs. low volatility) most of the time.

V. Related Work

Deep learning methods have increasingly been applied to financial time-series forecasting, with LSTM networks showing particular promise due to their ability to capture temporal dependencies in sequential data. Fischer and Krauss applied LSTM to predict directional movements of S&P 500 constituent stocks and found that LSTM outperformed traditional machine learning methods including random forests and logistic regression [6]. However, a key challenge remains: stock price movements often follow a random walk pattern that makes directional prediction inherently difficult.

In contrast, volatility exhibits distinctly different properties that make it more amenable to forecasting. Engle's foundational work established that volatility tends to cluster—large price changes are followed by large changes, and small changes by small changes [7]. Cont further documented this as one of several "stylized facts" of financial returns, noting that while returns themselves show little autocorrelation, absolute returns exhibit persistent patterns [8]. This distinction is reflected in the results of this study, where the initial directional prediction model showed weak correlation signals of approximately 0.04, but after pivoting to volatility prediction, correlation signals improved to 0.29. This 7.2x increase aligns with the theoretical literature on volatility's predictable nature.

VI. Conclusion

This study examined the use of Long Short-Term Memory (LSTM) neural networks to predict the next-day volatility in the S&P 500 index using a variety of engineered financial features. The original aim of this project was directional forecasting, but weak correlation led to a shift in focus, directing it toward volatility prediction. This enabled the model to leverage stronger statistical signals present in the dataset, resulting in improved performance. Incorporating technical indicators, volatility measures, and trend-based features enabled the LSTM network to capture both short-term fluctuations and longer-term market behavior.

Although the model's overall accuracy of 59.46% and ROC-AUC of 62.86% indicate performance meaningfully above random chance, the imbalance between high- and low-volatility predictions suggests that further improvement is needed to increase the model's sensitivity to high-volatility events. Future work may include experimenting with alternative sequence lengths, incorporating macroeconomic variables and other external data such as news sentiment, adding attention-based models, or using ensemble models to further improve robustness. Overall, the results show deep learning methods, particularly LSTMs, to be promising tools for modeling complex financial time-series patterns and can contribute valuable insights to quantitative market forecasting.

Supplementary Material

Please find the link to our source code below:

- **Source Code Repository:** https://github.com/tonygonzalez14/CS4440-Final-Project
- **Overleaf Project:** https://www.overleaf.com/read/wftqbpdmxwzd#463449

References

[1] Josh Starmer [StatQuest with Josh Starmer], "Long Short-Term Memory (LSTM), clearly explained," 11 2022.

[2] R. T. J. J., "LSTMs Explained: A Complete, Technically Accurate, Conceptual Guide with Keras," 9 2020.

[3] "yfinance," 9 2025.

[4] "Yahoo Finance."

[5] GeeksforGeeks, "What is YFinance library?," 5 2025.

[6] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.

[7] R. F. Engle, "Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of United Kingdom Inflation," *Econometrica*, vol. 50, no. 4, pp. 987–1008, 1982.

[8] R. Cont, "Empirical properties of asset returns: stylized facts and statistical issues," *Quantitative Finance*, vol. 1, no. 2, pp. 223–236, 2001.