# Machine Learning on Stock Exchange Prediction

Aash Makwana

April 29, 2025

## Contents

# Abstract

In classical mathematical finance, stock price dynamics are modeled using stochastic differential equations (SDEs) driven by Brownian motion in which we assume Geometric Brownian Motion (GBM) as the price evolution framework. While working with empirical data it reveals non-linear, noisy, and path-dependent behaviors that makes it hard for such simplified models. In this report, I used historical data from 2015 to 2024 and reinterpret statistical learning models (Random Forests, LSTMs, and hybrid ensembles) as tools for learning drift and diffusion characteristics from data, approximating Brownian-driven price processes. I assess model accuracy and discuss their ability to generalize beyond parametric assumptions which further offers insights into financial modeling through a stochastic way.


# 1 Introduction

Brownian motion forms the theoretical backbone of modern finance which appears in models like the Black-Scholes equation and geometric Brownian motion (GBM) for stock price dynamics:

$$dS_t = \mu S_t dt + \sigma S_t dW_t$$

where $\mu$ and $\sigma$ represent drift and volatility, and $W_t$ is a standard Brownian motion. Yet real markets show behavior beyond such assumptions. Volatility clustering, regime shifts, and long memory effects suggest the need for flexible, data-driven approaches to learn stochastic dynamics.

In this study, I aim to:

- Explore how machine learning and deep learning models can approximate non-linear drift and stochastic volatility patterns inherent in stock price trajectories.
- Evaluate these models against NVIDIA stock price data and assess their alignment with Brownian-inspired structures.

# 2 Data and Feature Construction in the Context of Stochastic Processes

## 2.1 Overview

I am using data from Kaggle's NVIDIA Stock Prediction dataset spanning 2015 to 2024. The features derived mimic statistics observable in Brownian paths: price changes (returns), volatility proxies, and temporal dependencies.

## 2.2 Feature Engineering

In this section I'll define all variables and engineered features which we derived and then used in the NVIDIA stock prediction model.

### Core Variables (Raw Data)

- Open, High, Low, Close, Adjusted Close, Volume.

**Technical Indicators (Engineered Features)**

$\Longrightarrow$ **Moving Averages**

- **SMA7 (Simple 7-Day Moving Average)**:

$$\text{SMA}_7 = \frac{1}{7}\sum_{i=1}^{7}\text{Close}_i$$

  This feature helps us to smooth price trends over a 7-day window.

- **SMA21**:

$$\text{SMA}_{21} = \frac{1}{21}\sum_{i=1}^{21}\text{Close}_i$$

  This feature helps us to identify mid-term trends (e.g., monthly).

- **EMA12 (Exponential 12-Day Moving Average)**:

$$\text{EMA}_{12} = \text{Close}_t \cdot \alpha + \text{EMA}_{12,t-1} \cdot (1-\alpha), \quad \alpha = \frac{2}{12+1}$$

  This feature emphasizes recent prices with a smoothing factor $\alpha$.

- **EMA26**:

$$\text{EMA}_{26} = \text{Close}_t \cdot \beta + \text{EMA}_{26,t-1} \cdot (1-\beta), \quad \beta = \frac{2}{26+1}$$

  This feature is further used for calculating the MACD.

$\Longrightarrow$ **Momentum & Volatility**

- **MACD (Moving Average Convergence Divergence)**:

$$\text{MACD} = \text{EMA}_{12} - \text{EMA}_{26}$$

  This feature gives us signals for bullish/bearish trends when crossing its signal line.

- **MACD Signal**:

$$\text{Signal} = \text{EMA}_9(\text{MACD})$$

  This feature is a 9-day EMA of the MACD line.

- **RSI (Relative Strength Index)**:

$$\text{RSI} = 100 - \frac{100}{1 + \frac{\text{Avg Gain (14 days)}}{\text{Avg Loss (14 days)}}}$$

  This feature identifies overbought (¿70) or oversold (¡30) conditions.

- **TR (True Range)**:

$$\text{TR} = \text{High} - \text{Low}$$

  This feature gives us daily price range.

- **ATR (Average True Range)**:

$$\text{ATR}_{14} = \frac{1}{14}\sum_{i=1}^{14}\text{TR}_i$$

This feature gives us 14-day average volatility.

$\Longrightarrow$ **Volume-Based Features**

- **OBV (On-Balance Volume)**:

$$\text{OBV}_t = \begin{cases} \text{OBV}_{t-1} + \text{Volume}_t & \text{if Close}_t > \text{Close}_{t-1}, \\ \text{OBV}_{t-1} - \text{Volume}_t & \text{if Close}_t < \text{Close}_{t-1}, \\ \text{OBV}_{t-1} & \text{otherwise.} \end{cases}$$

This feature tracks cumulative buying/selling pressure.

- **VWAP (Volume-Weighted Average Price)**:

$$\text{VWAP} = \frac{\sum(\text{Price} \times \text{Volume})}{\sum \text{Volume}}$$

This feature shows and captures institutional trading activity.

$\Longrightarrow$ **Returns & Temporal Features**

- **Daily Return**:

$$\text{Return} = \frac{\text{Close} - \text{Open}}{\text{Open}} \times 100$$

This feature calculates daily percentage price movement.

- **RoC (Rate of Change)**:

$$\text{RoC} = \frac{\text{Close} - \text{Open}}{\text{Open}} \times 100$$

This feature gives us a duplicate of Daily Return in this dataset.

- **Year/Month/Weekday/Day**: This features are extracted from **Date** to analyze seasonality and temporal patterns.

# 3 Methodology: Learning the Stochastic Dynamics

## 3.1 Random Forest Regressor

As a non-parametric regression tool, Random Forests approximate the functional form of drift $\mu(S_t, t)$ using lagged values and engineered indicators. Though lacking temporal memory, they are interpretable and provide feature importance.

Random Forest achieved good training accuracy but struggled to extrapolate beyond its training distribution—analogous to poor generalization of a drift estimator under changing stochastic regimes. The reason for overfitting was I used train-test split which

already gave some future values to model. This model was particularly trained by me to know feature importance which I can use further in Hybrid model.

Table 1: Random Forest Metrics (Training vs. Test)

| Metric | Training | Test |
|---|---|---|
| MSE | 0.0142 | 0.0747 |
| MAE | 0.1190 | 0.2733 |
| MAPE | 0.17% | 0.37% |
| $R^2$ | 1.0000 | 0.0977 |

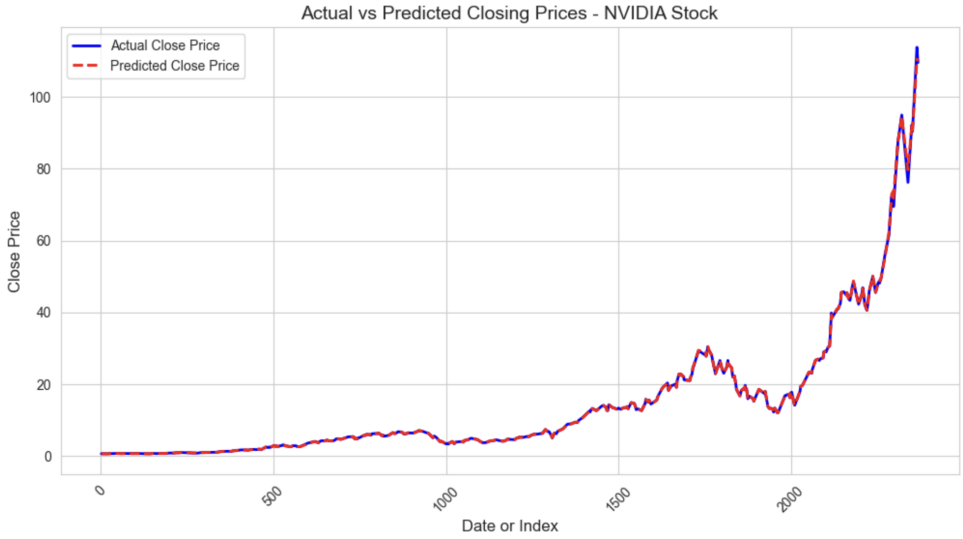


Figure 1: Actual vs Predicted Closing price using Random Forest

## 3.2 LSTM Networks

LSTMs model stock prices as sequences, allowing implicit modeling of temporal dependencies, which mimic integrated Brownian paths. Their architecture captures non-linearities in both drift and volatility processes without explicit model specification.

LSTM demonstrated excellent performance in capturing sequential dependencies, aligning well with the Brownian path view of price evolution. Its ability to model memory effects parallels concepts like fractional Brownian motion.

Table 2: LSTM Metrics (Training, Validation, Test)

| Metric | Training | Validation | Test |
|---|---|---|---|
| RMSE | 0.6086 | 1.2455 | 5.0986 |
| $R^2$ | 0.9934 | 0.9739 | 0.9360 |

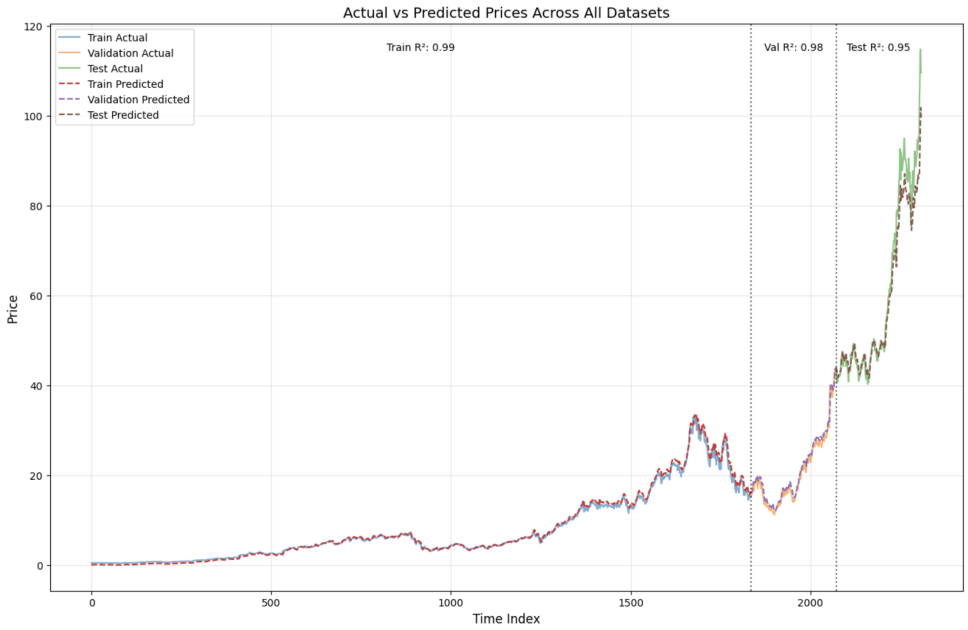


Figure 2: Actual vs Predicted (for training, validation and test set) Closing price using LSTM model

## 3.3 Hybrid Model: LSTM + AdaBoost

To refine the stochastic approximation, we use LSTM to forecast base trajectories (path approximation) and feed its output into an AdaBoost regressor. The ensemble enhances non-linear correction and volatility clustering capture.

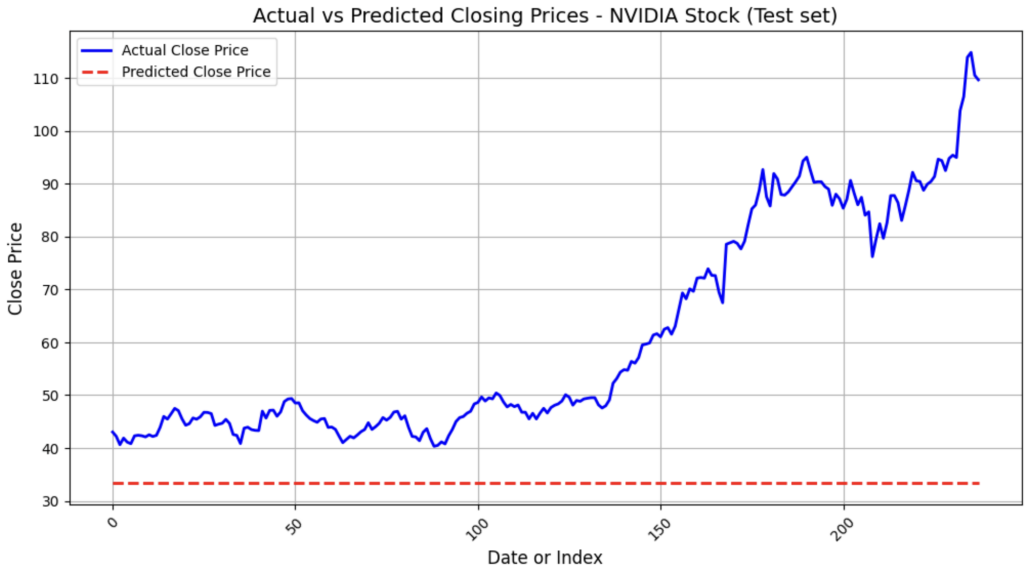


Figure 3: Actual vs Predicted (for test set) Closing price using Hybrid model, it badly overfits the data

Table 3: LSTM Metrics (Training, Validation, Test)

| Metric | Training | Validation | Test |
|---|---|---|---|
| MSE | 0.3105 | 3.7819 | 1155.3794 |
| $R^2$ | 0.9999 | 0.9362 | -1.8442 |

# 4 Discussion: Learning vs Modeling Stochasticity

- **Why ML models help**: Conventional stochastic models, such as Heston, Ornstein-Uhlenbeck, and Geometric Brownian Motion, rely on strict assumptions regarding volatility ($\sigma$) and drift ($\mu$), frequently handling them as simple functions of time or constants. These dynamics are implicitly learnt by ML models from data:
  - **Drift ($\mu$) as state-dependent**: Instead of assuming constant returns, machine learning models condition on features like volatility or macroeconomic indicators to capture regime-switching behaviour (such as bull/bear markets).
  - **Volatility ($\sigma$) as latent structure**: Without explicit parametric specifications, algorithms such as Random Forests infer volatility clustering, which is comparable to GARCH effects.
  - **Nonlinear dependencies**: ML models identify intricate relationships that are impossible to model with SDEs, such as those between order flow and price fluctuations.
- **Limitations**:
  - **Theoretical grounding**: There are no built-in no-arbitrage constraints or martingale guarantees in ML, in contrast to SDEs derived from Itô calculus.
  - **Tail behavior**: Since machine learning depends on sparse extreme-event data, models frequently miss jumps and black swans (for example, Mer-ton's jump-diffusion explicitly models these).
  - **Temporal consistency**: LSTMs may produce temporally incoherent predictions without regularisation, to overcome this SDEs maintain characteristics like stationary increments.
- **Overfitting stochastic noise**:
  - **False trends**: Machine learning most of the time confuse persistent signals with microstructure noise, such as bid-ask bounce.
  - **Spurious volatility**: Outliers can be interpreted as regime shifts by overparameterized models.
  - **Diagnostics**: Checking for identically and independently distributed residuals and compare ML volatility to realised volatility.
  - **Remedies**: Regularization (e.g., dropout), ensembling, or adversarial validation.

# 5 Future Directions

- Calibrate learned LSTM paths with simulated GBM paths.
- Incorporate stochastic volatility models (e.g., Heston) with learned parameter functions.
- Bridge with functional analysis via the Karhunen–Loève expansion to represent LSTM states.

# 6 Conclusion

We can better understand how data-driven models approximate stochastic dynamics by looking at stock price prediction through the lens of Brownian motion. In particular, LSTMs are adaptable estimators of price evolution that can capture important Brownian trajectory features that conventional models might miss. LSTMs outperformed Hybrid model of LSTM + AdaBoost in predicting NVIDIA stock prices, achieving a test $R^2$ of 0.9360. Future work I will explore more hybrid models to further refine predictions. This work highlights the potential of deep learning in financial forecasting.

# References


1. Karatzas, I., and Shreve, S. E. (1991). *Brownian Motion and Stochastic Calculus*. Springer.
2. Bobrowski, A. (2005). *Functional Analysis for Probability and Stochastic Processes*.
3. Breiman, L. (2001). Random Forests. *Machine Learning*.
4. Hochreiter, S., and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*.