Study of statistical, ML and deep learning method for stock price and volatility forecasting

Puneet Mishra

Report of work done for Winter Project (DA5410)

Wadhwani School of Data Science and AI

Indian Institute of Technology Madras

January 11, 2026

I, Puneet Mishra, confirm that the work presented in this report is my own. Wherever information has been derived from other sources, I confirm that this has been indicated in the work.

Contents

List of Tables

List of Figures

Abstract

In recent years, the stock market has become the busiest commercial place. From 2001 onwards, the stock market has witnessed a staggering 42x growth in capital value. With the advent of online apps like Groww and Upstox, the trading has been democratized and more people invest with each minute. With such large stakes involved, both the retail and institutional investors are anxious about which direction the market will take. Here, the forecasting algorithms are of great service. In this study, statistical, physical, machine learning and deep learning algorithms were employed to predict stock price and volatility on a set of 24 stocks from NSE (India), NASDAQ (USA), and LSE (UK) and the relative strengths of the models was determined. Lastly, a robust model was proposed to categorize the stocks into low, moderate, high and extremely high stocks.

Keywords: Stock price forecasting; volatility forecasting; machine learning; deep learning; Long short-term memory (LSTM); GARCH model;

1. Introduction

A stock market is a platform where company shares and other financial products are traded. Two parties are facilitated by the stock market: corporations that require finance and investment and investors who will make the investment. In the recent years, the number of investors in the stock market, particularly retail investors, has dramatically increased. Since the investments have high stakes, the investors want to have insight on whether a certain stock will go up or down. Although the stock market is a highly volatile market where prices are influenced by numerous and almost unpredictable factors, there are several mathematical and statistical models that can predict the prices and their volatility quite accurately. The deep learning models provide even better accuracy. However, these predictions must be taken with a grain of salt as the stocks are influenced by multiple other factors that can't possibly be captured by any model, no matter how good it is.

This study mainly aims on studying how several statistical, physical, machine learning (ML) and deep learning models perform on *stock price and volatility* forecasting. A *stock price* is the value of a stock or a share of a company and *volatility* is the standard deviation of the return obtained on the stock. The rest of the report is organized as

follows – Section 2 provides a review of the literature available on this subject. Section 3 introduces the statistical, physical, ML and deep learning algorithms that were studied. Section 4 provides the experimental setup, results and analysis of prediction. In Section 5, conclusion and future work are presented.

2. Literature Review

Forecasting stock prices and predicting market trends are challenging tasks. Over the years, researchers have proposed several solutions to these challenges. Since the stock market data is basically a time-series, the standard methods for time-series forecasting are applicable. Zhang & He employed *Moving Average (MA), Exponential Moving Average (EMA) and Long Short-Term Memory (LSTM)* algorithms and found that LSTM greatly outperforms MA and EMA. Ariyo et al. implemented *AR and ARIMA* models on NYSE and Nigeria Stock Exchange and concluded that their predictions are close to the actual prices for a period of ~10 days but they diverge for longer periods. Mashadihasanli concluded the same for Borsa Istanbul. Black-Scholes model based on Geometric Brownian Motion (GBM) takes a different approach. It suggests that the stock prices move in a random walk fashion. Abidin & Jaffar tested in on Bursa Malaysia market and found positive results.

The modelling of the time-series can also be seen as a regression task. Soni et al. in their paper provide an extensive review of several machine learning approaches in stock price prediction. Non-linear regression methods such as Support Vector Machines (SVMs) have also been found to be extremely useful for the task. Mailinda et al. experimented with SVM, Back Propagation Neural Network (BPNN) and LSTM models to predict prices for BBRI stock in Indonesia Stock Market (IDX) in COVID-19 period when the market was at its most volatile. They discovered that SVM outperformed even the sophisticated BPNN and LSTM models. Tsai & Wang showed that combining Artificial Neural Network (ANN) and Decision Tree (DT) into a hybrid model increases accuracy. Sonkavde et al. did a systematic analysis of several statistical, ML, deep learning as well as ensemble methods and reported their performance.

Forecasting the volatility is an equally challenging task. A standard statistical method for volatility forecast is Generalized Auto-Regressive Conditional

Heteroskedasticity (GARCH). It assumes that the volatility *clusters* i.e. days of high volatility are followed by days of high volatility and low volatility days are followed by low volatility days. It can be formalized as

$$\sigma_t^2 = \omega + \sum_{\{i=1\}}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{\{j=1\}}^{p} \beta_j \sigma_{\{t-j\}}^2$$

where σ_t^2 is the current conditional volatility; ω is a constant; $\alpha_i \epsilon_{t-i}^2$ are the ARCH terms which record the impact of previous unexpected returns on today's volatility $\beta_j \sigma_{t-j}^2$ these represent how much yesterday's volatility 'lingers' into today. Sharma & Vipul implemented GARCH with several advanced GARCH models on data of 21 major global stocks over a period of 13 years and found that GARCH (1,1) gives the best result. However, for long periods, ML and DL methods are as good, if not better. Christensen et al. did an extensive comparison study of ML models and neural networks against Heterogenous Auto-Regressive (HAR) models and found that even with minimal hyperparameter tuning ML models beat HAR models with same set of predictors. Niu et al. also incorporated how the geopolitical risks affect volatility in their study of ML models.

Ke et al. proposed implementing a BPNN with genetic algorithm in their novel method for volatility prediction. Liu (2019) compared GARCH, LSTM and Support Vector Regressor (SVR) and found that GARCH (1,1) is the best predictor if only the previous day's data is considered. However, if last two days are considered, LSTM outperforms even GARCH (2,2) and SVR. MOON & KIM experimented with different sets of features and targets for LSTM model to predict price and volatility and found that the volatility predictions don't depend upon other variables like Open, Close, High and Low prices. They also found that increasing the number of features does not improve the accuracy for the volatility prediction, while it does for the index itself.

In this study, many of the aforementioned models were tested on certain benchmark experiments and their performance was measured.

3. The Models

In this study, the following models were studied –

3.1 Moving Average: This is one of the simplest forecasting models. It simply takes the average of last n

observations.(Zhang & He, 2023) It can be formally expressed as –

$$y_t = \frac{1}{n}\sum_{i=0}^{n-1} y_{t-i}$$

The lookback window was taken as $n = 7$.

3.2 Exponential Moving Average:

In this model, the recent observations are given more weight than the previous ones while taking moving average. (Zhang & He, 2023) It can be expressed as –

$$y_t = \alpha p_t + (1-\alpha) y_{t-1}, 0 < \alpha < 1$$

Where α is a constant and p_t is the price on the previous day.

3.3 Geometric Brownian Motion (GBM) :

It is a continuous stochastic process which is modelled after the physical phenomenon of Geometric Brownian Motion. A stock is treated as a particle that is moving with a drift and volatility. The change in the position of a stock is directly affected by a stochastic process known as Wiener process. (Abidin & Jaffar, 2014) It is given by –

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

where μ is drift rate, σ is volatility and dW_t is the Wiener process variable.

3.4 Autoregressive (AR):

It models the current value as a linear function of its past values, capturing temporal dependence which is a very effective technique for stock price prediction. (Akaike & Kitagawa, 1999). It can be formalized as –

$$y_t = \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t$$

Where ϕ_i are coefficients for past prices in the regression and ϵ_t is the noise variable. (Akaike & Kitagawa, 1999, p. 25)

3.5 Autoregressive Integrated Moving Averages (ARIMA):

The ARIMA model is based on *AR* and *MA* models. While the *AR* model is used to show that the current observation is dependent on previous observations, the *MA* model is used to show that the current and previous residuals compose a linear function. (Chang et al., 2009). General statement for these models is ARIMA (*p,d,q*) where *p* denotes the degree of *AR* model, *d* denotes the degree of different order and *q* denotes the degree of *MA* model. The ARIMA (p, d, q) model takes the following form (Mashadihasanli, 2022) –

$$\Delta dy_t = c + \sum_{i=1}^{p} \phi_i \Delta dy_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

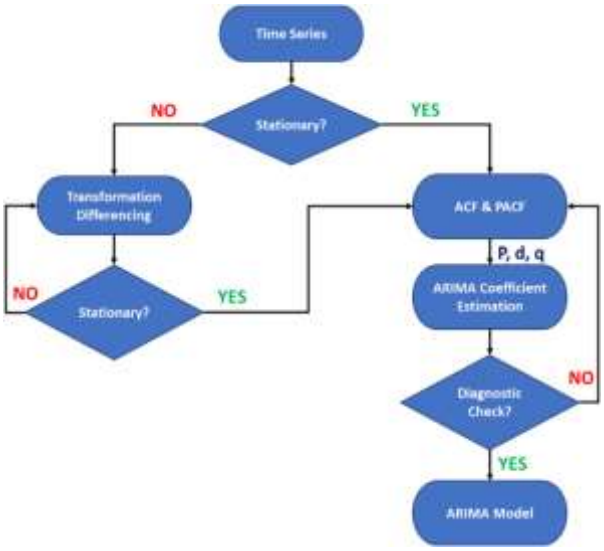

Figure 1: Steps of ARIMA model

3.6 Linear Regression: It is the first ML model employed in this study. It is practically a generalized form of AR model. It gives the target (price or volatility) as –

$$y_t = \beta_0 + \beta_1 X_t + \epsilon_t$$

where βs are a set of weights and biases and ϵ_t denote random noise. X_t is a set of predictors known as features. (Bishop, 2009) The model was tested for different set of features.

Though several variants of LR exist such as Bayesian LR, regularized LR, etc. but only the simplest linear regression model was chosen for this study to have a flexible fit.

3.7 Support Vector Regression (SVR): Though it is primarily designed for non-linear regression tasks, it can as well perform linear regression by mapping the data into a higher-dimensional space and constructing a regression line in that higher space and projecting the prediction onto the lower dimension space. The mapping is done by a *kernel* function such as the RBF function used here. (Liu, 2019) The RBF kernel is given by –

$$k(x_1, x_2) = e^{-\lambda \|x_1 - x_2\|^2}$$

It is particularly robust to avoid overfitting. Contrary to LR which works by minimizing the sum of squared errors and hence is prone to overfitting by trying to close the points to the line as much as possible. SVR, on the other hand, builds a ϵ-*insensitive tube* i.e. if the error for a point is less than the given ϵ, the model ignores that error instead of trying to fit it. (Bishop, 2009)

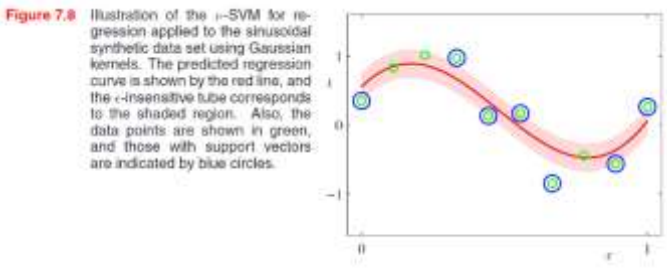

Figure 2: ϵ-insensitive tube description from (Bishop, 2009)

The support vectors are the data points that lie either on the boundary of the ϵ-insensitive region or outside it and they determine the regression line.

3.8 Random Forest Regression:

It is an ensemble method that operates by constructing multiple decision trees model by resampling features and data points using bootstrap sampling. Each tree is trained on the sampled points and makes predictions. Then those predictions are aggregated and their average is taken as final prediction i.e.

$$y_t = \frac{1}{B}\sum_{b=1}^{B} T_b(X)$$

where $T_b(X)$ is the prediction made by a single tree among a total of B trees. It is based on the fact that taking average reduces the variance and hence the overfitting. (James et al., 2023)

3.9 XGB Regression:

It is another ensemble method but unlike RF, it is based on *boosting* instead of simply bagging. In this algorithm, the model constructs decision trees *sequentially* rather than in parallel like RF regression. Each new tree is added to correct the residual errors made by the existing ensemble of trees. It is particularly robust to avoid overfitting due to its unique regularized objective function. Contrary to standard Gradient Boosting Machines (GBM), which minimize a simple loss function and can easily overfit if trees grow too complex, XGB incorporates a regularization term into its objective. This term penalizes the complexity of the model e.g., the number of leaves and the magnitude of leaf weights.

3.10 Back Propagation Neural Network (BPNN):

A BPNN is a multi-layer feedforward neural network. The BPNN used here consists three layers of 128, 64 and 32 neurons with ReLU activation. Before going through the activation, each linear transformation is immediately followed by Batch Normalization to accelerate convergence and stabilize the learning process. (Ioffe & Szegedy, 2015)

The final prediction is generated by a single linear output unit. The network minimizes the loss by propagating error gradients backward through these layers.

It is explicitly regularized to avoid overfitting. While standard BPNNs are prone to memorizing noise, this architecture mitigates that risk by employing *Dropout* with a probability of after each activation. This technique randomly "drops" (zeros out) 30% of the neurons during each training pass, effectively training a large ensemble of thinned networks and forcing the model to learn robust features that do not rely on the presence of any single specific neuron.

3.11 Long Short-Term Memory (LSTM)

The Long Short-Term Memory network is a specialized Recurrent Neural Network (RNN) architecture designed to model temporal sequences. Unlike standard RNNs, LSTMs handles the vanishing gradient problem by maintaining a constant error flow through a dedicated **Cell State** (C_t). Information flow is regulated by three gating mechanisms –Forget, Input, and Output which selectively retain or discard information at each time step. (Hochreiter & Schmidhuber, 1997)

The implemented model utilizes a **Stacked LSTM** architecture to extract hierarchical temporal features. The first layer consists of 64 units and retains the full sequence structure, passing a temporal output to the second layer. The second layer, comprising 32 units, compresses this sequence into a final context vector. This vector is processed by a 64-unit dense layer with ReLU activation before generating the final regression prediction.

It is heavily regularized to prevent overfitting on sequential data. The model employs a multi-faceted dropout strategy: *Input Dropout* ($p = 0.2$) to noise the incoming data, *Recurrent Dropout* ($p = 0.2$) to mask state-to-state transitions within the LSTM cells, and *standard Dropout* ($p = 0.3$) before the final output. This forces the network to learn robust temporal patterns rather than relying on specific past observations.

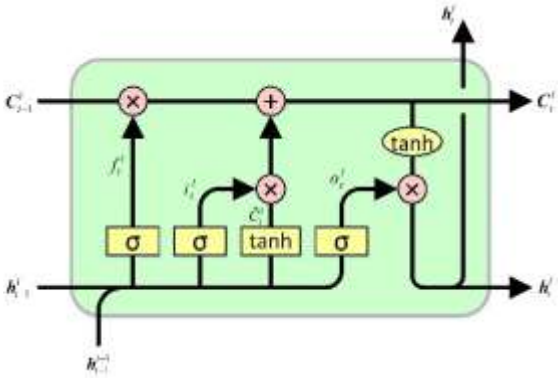

Figure 3: An LSTM cell with input, output and forget gates.

3.12 Generalized Autoregressive Conditional Heteroskedasticity (GARCH) (1,1)

The GARCH(1,1) model is a statistical method used to analyse time-series data where the variance of the error is believed to be serially autocorrelated. While standard regression models assume the volatility of the error term is constant (homoskedasticity), GARCH explicitly models this volatility as changing over time. The (1,1) specification indicates that the current conditional variance depends on one lag of the past squared error (the ARCH term) and one lag of the past conditional variance (the GARCH term).

The current conditional variance is determined by the long-run variance weight , the ARCH coefficient , and the GARCH coefficient :

It is particularly robust in capturing **volatility clustering**. Contrary to Ordinary Least Squares (OLS) regression, which assumes constant variance and fails to account for "bursts" of fluctuation (where large changes tend to be followed by large changes), GARCH(1,1) dynamically adjusts to these periods.

To ensure the model remains stable and does not predict exploding volatility (stationarity), the sum of the persistence parameters is constrained such that . This ensures that shocks to the volatility eventually die out and the variance reverts to a long-run mean, preventing the model from overreacting to temporary noise.

All of these models were implemented using the following methodology.

4. Methodology

4.1 Data loading and preprocessing

The stock price data of period from *2018-03-01 to 2025-04-30* was downloaded from *yfinance* library of python. The models were tested on *'RELIANCE.NS'* ticker and then implemented on a broader set of tickers i.e.

RELIANCE.NS, TCS.NS, HDFCBANK.NS, PIDILITIND.NS, AUBANK.NS, IEX.NS, MAPMYINDIA.NS, LTTS.NS, AAPL, MSFT, GOOGL, AMZN, NVDA, CRWD, PLTR, FSLR, CELH, U, HSBA.L, BP.L, ULVR.L, AUTO.L, JDW.L and ASC.L

These stocks come from NSE (India), NASDAQ (USA) and LSE (UK) and are of diverse market cap.

The library provides the data in form of a multi-index dataframe. The dataframe was transformed to a single-index structure and its index was reset to its date column.

The dataset was then divided into training set consisting data from *2018-03-01* to *2025-03-31* and a test set from *2025-04-01* to *2025-04-30.* The models were trained on the training set and tested on the test set.

4.2 Experiment set up

4.2.1 Comparison of statistical and physical method

The first experiment in the study was to compare the statistical models of MA, EMA, AR and ARMA with the physical GBM model. This experiment is recorded in the notebook `price_stat_phys.ipynb` and the results are displayed in the section 4.3.1. The MA and EMA models were constructed with a lookback window of 7 days. The GBM model was constructed from the formula –

$$S_{t+1} = S_t e^{\left(\mu - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t}Z}$$

where S_t is the price at the current time step; μ is the drift parameter, calculated as the mean of the logarithmic returns; σ is the volatility parameter, calculated as the standard deviation of the logarithmic returns; Δt is the time increment; and Z is the random walk factor drawn from the Z-distribution and $\sqrt{\Delta t}Z$ terms denote the Wiener process.

The stats models were taken from python's `statsmodels.tsa` module, trained and then forecasts for the test period was obtained using the `.forecast` method.

4.2.2 Comparison of ML models

Three experiments were designed to study the ML models.

a) Models were trained on the raw OHLCV data with 'Close' price as target and the rest as predictors.
b) They were then trained with 'Momentum' as the target variable. Momentum is defined as –
$$M(S_t, m) = S_t - S_{t-m}$$
where m is the lookback considered to calculate momentum, here 20 days.
c) Five set of features were extracted to understand which group of features contain the most of the information. The set of features were extracted in block as follows –

Block	Features included
B1	Close_{t-1}, Close_{t-2}

Block	Features included
B2	1-day return, 5-day return
B3	B2 + MA(5), MA(20), EMA(5), EMA(20)
B4	B3 + distance from trend
B5	B4 + Volume dynamics

These experiments were conducted in `price_ml.ipynb` for single stock and in `price_ml_full_expt.ipynb` for the entire set of 24 stocks. The results are in section 4.3.2.

4.2.3 BPNN and LSTM for Price forecasting

The BPNN and LSTM models were first tested with 'Close' as target then with 'log return of Close' as target. The log return is given by –

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right)$$

where P_t and P_{t-1} are closing prices.

The features were lagged for one day in both ML and DL models so that they learn to predict the closing price for today from the data of the previous day.

The BPNN and LSTM experiments were conducted in `price_nn.ipnyb` and `price_lstm.ipynb` respectively and the results are in section 4.3.3.

4.2.4 Volatility forecasting

The general approach remained the same in studying models for volatility forecasting. The models were trained on data from *March 2018* to *March 2025* and the predictions were tested against the actual data of *April 2025.* Three experiments were conducted for volatility forecasting in the notebook `volatility.ipynb` and the results are shown in section 4.3.4.–

a) GARCH (1,1), ML, BPNN, and LSTM algorithms were implemented with volatility derived from log returns using a rolling window of 5 days given by –

$$\sigma_t = \frac{1}{w-1}\sum_{i=0}^{w-1}(r_{t-i} - \bar{r_t})^2$$

where r_t is the log return; $w = 5$ is the window size and $\bar{r_t}$ is the mean of return within the window. The algorithms were trained with this volatility as target. GARCH parameters were determined from the partial

autocorrelation plot of the square of log returns (a proxy for volatility).

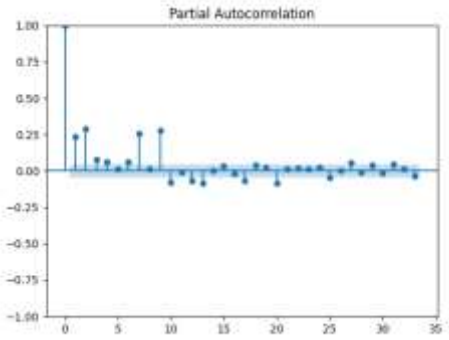

Figure 4: PACF plot of RELIANCE.NS data

This plot shows significant autocorrelation at the initial lags, suggesting volatility clustering. Further, instead of a sharp cutoff like in an ARCH process, there is a slow decay pattern which re-emerges at lags 7 and 9. This means that volatility has a "long memory" therefore a generalized heteroskedasticity model such as GARCH (1,1). (Engle, 2001).
An extensive feature extraction was necessary to train robust ML models. The following features were extracted –

*** Volatility clustering features:** It's observed in stock market that volatility clusters i.e. periods of high volatility are followed by periods of high volatility. (Akaike & Kitagawa, 1999).The lagged realizations of the realized volatility were extracted for 1, 2, 5 and 10 days, capturing both short and long memory.

*** Intraday volatility estimators:** Intraday shocks also contribute to volatility. Two proxy features were extracted –

(i) Parkinson's proxy, given by $P = mean\left(\left[\ln\left(\frac{H_{t-i}}{L_{t-i}}\right)\right]^2\right)$ where H and L are High and Low prices respectively.

(ii) Garman-Klass proxy, as

$$\sigma_{GK}^2 = 0.5\left(\ln\left(\frac{H_t}{L_t}\right)\right)^2 - (2\ln 2 - 1)(\ln\frac{C_t}{O_t})^2$$

where C and O are closing and opening prices respectively.

Both proxies were implemented with a rolling window of 5 and 10 days to smooth out daily noise.

*** Volume** was log-transformed to handle the heavy-tailed distribution of volume data.

*** Squared log returns** of previous day was taken as

direct proxy for most recent variance shock.

All features were lagged to prevent a *look-ahead bias.* The same set of features were also used to train BPNN and LSTM models. However, unlike the static BPNN, LSTM takes a sliding-window approach to preserve the temporal order of the data.

b) The best performing algorithm in the previous experiment was taken and it was tested that on which return, daily, weekly or monthly, do they give the best results.

c) The best model and the best interval for returns was selected and the volatility of 24 stocks were predicted for April 2025 and they were categorized into low, moderate, high and extremely high volatile stocks.

4.3 Results

4.3.1 Comparison of statistical and physical models

It was found that EMA model produced the best results among all the statistical and physical models.

The results are presented in *Table 1.*

	RMSE					SMAPE				
Model	**AR**	**ARIMA**	**EMA**	**GBM**	**MA**	**AR**	**ARIMA**	**EMA**	**GBM**	**MA**
Ticker										
AAPL	24.96	24.07	13.33	25.14	15.39	10.66	10.19	4.85	10.65	5.68
AMZN	11.45	10.98	9.36	12.61	10.55	5.06	4.81	4.25	5.65	4.83
ASC.L	34.60	30.16	17.83	22.51	21.80	10.43	8.81	5.60	6.35	6.64
AUBANK.NS	87.25	88.54	38.33	85.67	42.35	11.19	11.41	4.89	10.68	5.42
AUTO.L	48.34	48.65	24.15	44.95	27.15	5.28	5.31	2.83	4.87	3.09
BP.L	77.65	77.84	29.93	78.88	34.91	18.39	18.44	5.75	18.64	7.06
CELH	1.30	1.46	1.10	1.14	1.02	3.05	3.51	2.49	2.64	2.36
CRWD	40.17	42.15	25.02	38.62	27.93	9.11	9.59	5.91	8.54	6.75
FSLR	7.23	7.50	5.55	7.24	5.96	4.36	4.50	3.40	4.33	3.61
GOOGL	5.33	5.35	5.71	5.35	6.47	2.88	2.95	3.21	2.98	3.75
HDFCBANK.NS	39.09	39.54	23.95	35.43	28.10	3.81	3.84	1.99	3.55	2.31
HSBA.L	93.02	92.21	53.01	94.67	63.43	9.71	9.60	5.43	9.84	6.86
IEX.NS	11.13	11.92	4.80	10.53	5.04	5.06	5.44	2.29	4.65	2.28
JDW.L	46.79	42.43	22.01	54.02	24.66	6.29	5.72	3.22	7.22	3.44
LTTS.NS	254.29	239.37	172.69	263.85	196.13	5.08	4.69	3.49	5.29	4.03
MAPMYINDIA.NS	94.43	98.16	45.68	83.87	51.01	4.82	5.03	2.12	4.03	2.36
MSFT	12.32	12.84	13.34	12.78	15.33	2.96	3.04	3.23	3.03	3.79
NVDA	7.39	6.80	7.00	6.90	7.93	5.43	5.09	5.68	5.14	6.83
PIDILITIND.NS	70.13	73.13	29.06	68.03	30.58	4.09	4.28	1.75	3.93	1.80
PLTR	13.99	15.88	8.49	14.89	9.48	11.64	13.50	7.59	12.21	8.86
RELIANCE.NS	61.48	61.91	49.08	63.45	56.63	3.95	3.99	3.38	4.07	4.04
TCS.NS	261.10	256.09	115.37	269.61	140.94	7.11	6.95	2.86	7.30	3.61
U	1.66	1.78	1.61	1.84	1.79	7.30	7.78	6.56	7.90	7.46
ULVR.L	167.47	171.98	110.46	134.80	128.01	2.84	2.91	1.90	2.29	2.29

Table 1: Comparison of statistical and Physical mode

The reason could be that EMA models the *memoryless* nature of stock prices well by giving more weight to the recent observations whereas other models are just overfitting the random changes in daily prices which is futile.

4.3.2 Comparison of ML models

a) Linear Regression model gave the best result for simple modelling on OHLCV data with *"Close"* as target. The reason could be that the LR model simply sets a high coefficient for the previous day's price which matches the *memorylessness* of price trend. These results are presented in *Table 2*.

Figure 5: performance of LR for RELIANCE.NS

b) The models performed very poorly when *'momentum'* was taken as the target. Because here first the momentum was predicted then price was derived from the predicted momentum. This two-step approach causes an *error accumulation* where a tiny error in momentum prediction causes a large error in price prediction. Another reason is a theory known as ***Weak Form Efficient Market Hypothesis***, which argues that the future movement of prices is an absolute random walk and past trends are not good predictors of the future. (Malkiel, 2015). This does not, however, forbids taking insights from past data.

c) The feature block B1 gave very poor results implying that a simple short-memory model is not enough. The results started to stabilise once the returns and moving averages were added to the feature set suggesting keeping both long and short memory is beneficial. In fact, this is the basis behind using LSTM RNNs for this task.

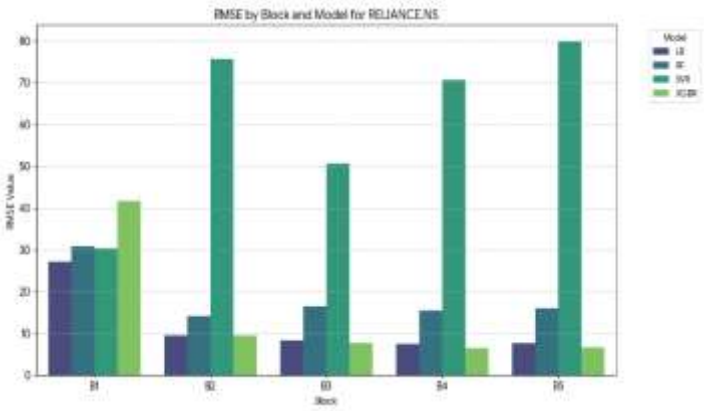

Figure 6: performance of models for different feature blocks.

	MAE				RMSE				R2			
Model	LR	RF	SVR	XGBR	LR	RF	SVR	XGBR	LR	RF	SVR	XGBR
Ticker												
AAPL	7.68	9.16	7.49	8.51	10.41	11.70	9.74	10.31	0.14	-0.08	0.25	0.16
AMZN	5.74	5.11	5.30	4.72	7.85	6.60	6.78	6.25	-0.03	0.27	0.23	0.34
ASC.L	11.15	16.65	115.05	13.43	14.10	19.05	120.91	15.94	0.57	0.21	-30.80	0.45
AUBANK.NS	17.34	16.79	24.97	17.62	21.64	20.91	36.12	22.10	0.85	0.86	0.59	0.85
AUTO.L	11.78	13.37	14.40	14.65	16.32	19.32	20.10	20.06	0.77	0.68	0.65	0.65
BP.L	13.81	9.63	12.64	11.01	20.67	13.76	18.55	15.74	0.21	0.65	0.36	0.54
CELH	0.79	1.28	0.70	1.09	1.10	1.64	1.03	1.43	-0.40	-2.10	-0.24	-1.36
CRWD	14.61	16.83	18.50	14.54	18.29	21.53	22.60	18.58	0.66	0.53	0.48	0.65
FSLR	6.02	5.92	5.46	5.88	7.19	7.10	6.48	7.17	-0.42	-0.39	-0.16	-0.41
GOOGL	3.72	3.88	4.01	4.41	4.68	5.00	4.72	5.32	0.20	0.09	0.19	-0.03
HDFCBANK.NS	12.26	23.30	57.64	24.00	15.65	26.06	68.71	27.09	0.77	0.36	-3.46	0.31
HSBA.L	21.79	21.76	22.86	23.68	29.96	33.49	28.96	33.06	0.60	0.50	0.63	0.51
IEX.NS	2.25	2.99	3.57	3.11	2.73	3.57	4.32	3.84	0.82	0.69	0.55	0.64
JDW.L	10.19	11.55	20.72	13.43	12.82	13.49	31.60	16.01	0.86	0.85	0.16	0.78
LTTS.NS	99.86	98.10	124.29	102.06	124.15	128.32	149.35	126.05	0.33	0.29	0.04	0.31
MAPMYINDIA.NS	24.81	32.81	34.13	30.94	33.68	44.14	44.88	39.99	0.35	-0.11	-0.15	0.09
MSFT	8.83	8.86	9.66	7.84	11.37	10.72	12.46	9.29	0.20	0.29	0.04	0.46
NVDA	4.66	3.89	4.71	5.10	6.05	4.96	5.62	6.15	-0.02	0.32	0.12	-0.05
PIDILITIND.NS	15.16	19.49	15.98	18.48	19.62	24.18	19.07	23.02	0.71	0.56	0.73	0.60
PLTR	4.61	4.64	5.56	6.51	5.69	5.47	6.84	7.37	0.77	0.79	0.67	0.62
RELIANCE.NS	22.39	27.86	30.90	28.16	29.41	32.38	40.66	33.36	0.77	0.72	0.56	0.71
TCS.NS	48.53	42.22	63.64	49.82	63.05	57.93	91.04	67.00	0.49	0.57	-0.06	0.43
U	0.76	1.00	0.85	0.94	1.12	1.32	1.08	1.20	0.60	0.45	0.63	0.55
ULVR.L	80.03	84.31	89.85	90.54	112.17	113.15	114.78	121.40	0.00	-0.02	-0.05	-0.17

Table 2: Comparison of ML Models (Price target)

	MAE				RMSE				R2			
Model	LR	RF	SVR	XGBR	LR	RF	SVR	XGBR	LR	RF	SVR	XGBR
Ticker												
AAPL	16.99	25.51	25.32	23.96	18.68	29.83	29.01	29.33	-1.38	-5.07	-4.74	-4.87
AMZN	14.41	21.59	23.87	21.98	16.54	23.51	25.12	23.71	-3.34	-7.77	-9.01	-7.92
ASC.L	39.06	55.00	22.67	55.38	46.03	60.62	29.32	62.48	-3.86	-7.42	-0.97	-7.95
AUBANK.NS	65.62	61.20	67.73	60.02	78.12	70.96	78.40	69.19	-0.84	-0.52	-0.85	-0.44
AUTO.L	33.05	51.94	41.21	52.32	37.14	57.96	47.93	60.67	-0.25	-2.04	-1.08	-2.33
BP.L	62.98	73.36	75.70	73.46	68.63	78.77	80.45	81.45	-5.03	-6.94	-7.28	-7.49
CELH	5.48	3.39	6.37	4.21	6.27	3.78	7.19	4.95	-45.25	-15.77	-59.81	-27.81
CRWD	22.95	16.32	17.03	16.74	28.59	23.54	20.83	22.08	0.14	0.42	0.54	0.49
FSLR	7.75	11.04	8.60	13.51	9.05	12.41	11.09	15.14	-1.34	-3.41	-2.52	-5.56
GOOGL	10.96	17.09	12.22	17.03	13.38	18.87	14.42	19.08	-5.78	-12.49	-6.88	-12.79
HDFCBANK.NS	46.21	18.92	22.66	20.35	52.37	22.93	27.46	24.53	-1.47	0.53	0.32	0.46
HSBA.L	98.44	122.72	123.05	124.20	104.15	127.90	125.43	127.85	-3.35	-5.56	-5.31	-5.55
IEX.NS	20.92	20.00	22.62	16.89	21.89	21.26	23.45	18.88	-10.30	-9.65	-11.96	-7.41
JDW.L	40.38	52.20	42.16	58.41	50.96	72.83	54.95	74.75	-1.19	-3.46	-1.54	-3.70
LTTS.NS	344.31	292.21	376.85	334.40	408.64	356.65	439.65	406.12	-5.51	-3.96	-6.53	-5.43
MAPMYINDIA.NS	110.81	144.16	144.32	136.16	125.67	169.20	159.71	171.11	-7.46	-14.33	-12.66	-14.68
MSFT	14.24	30.47	15.83	29.45	18.03	34.20	19.94	33.16	-1.11	-6.59	-1.58	-6.13
NVDA	11.65	10.65	10.45	10.76	13.79	13.34	12.76	13.18	-4.35	-4.01	-3.59	-3.89
PIDILITIND.NS	89.85	103.04	92.35	95.04	94.41	111.68	96.62	109.66	-4.78	-7.09	-5.06	-6.80
PLTR	9.97	12.72	8.21	13.08	11.57	14.81	10.69	15.54	0.03	-0.58	0.17	-0.74
RELIANCE.NS	44.81	47.14	42.66	48.66	51.92	60.84	49.55	67.59	0.25	-0.03	0.31	-0.27
TCS.NS	229.39	228.47	206.29	229.63	253.00	238.56	224.58	245.53	-5.75	-5.00	-4.32	-5.36
U	2.55	2.17	2.13	2.33	3.19	2.86	2.63	2.99	-2.39	-1.72	-1.31	-1.97
ULVR.L	131.85	128.81	146.29	150.80	164.56	162.32	173.20	179.49	-1.22	-1.16	-1.46	-1.65

Table 3: Comparison of ML Models (Momentum target)

4.3.3 BPNN and LSTM for price forecasting

BPNN when trained on the raw data with *"Close"* as target gave disappointing and inconsistent results. However, the results dramatically improved when the network was trained with log-returns as target and price was predicted by taking the exponential of the predicted returns.

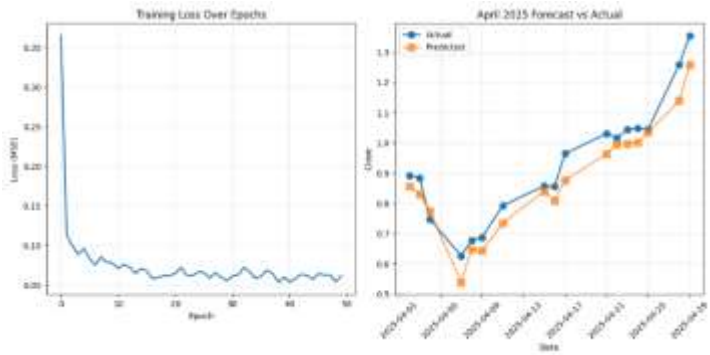

Figure 7: BPNN results with 'Close' as target for RELIANCE.NS

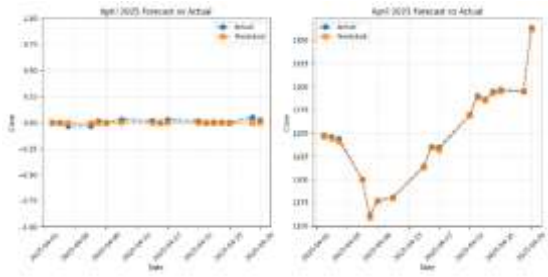

Figure 8: BPNN results with 'log-returns' as target. The left plot shows the actual and the predicted values of the returns while the right plot shows that of the 'Close' price for RELIANCE.NS

That is so because 'log-return' is a stationary variable whereas 'Close' is not. Neural networks struggle in learning non-stationary targets as they violate the assumption of *independent and identically distributed (i.i.d.)* data. (Goodfellow et al., 2017)

LSTM model followed the same pattern but it didn't give as good results, probably because LSTM requires huge amount of data compared to a simple MLP architecture and the dataset of simple ~1500 rows was not enough for it to function properly.

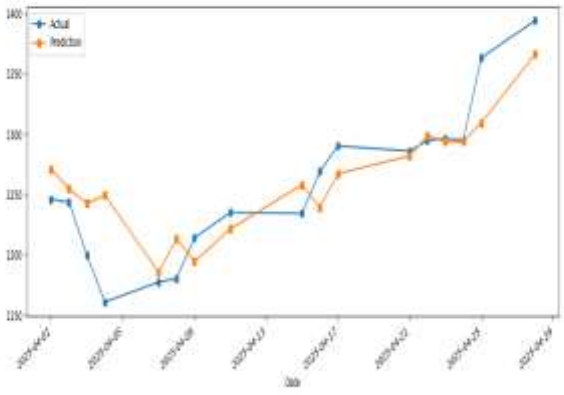

Figure 9: Prediction of RELIANCE.NS 'Close' prices with LSTM model.

Ticker	MAE	RMSE	R2
RELIANCE.NS	40.29	46.61	0.39
TCS.NS	64.89	74.32	0.42
HDFCBANK.NS	36.87	39.98	-0.44
PIDILITIND.NS	71.30	73.34	-2.49
AUBANK.NS	22.99	31.65	0.70
IEX.NS	7.65	8.21	-0.59
MAPMYINDIA.NS	24.68	30.74	0.49
LTTS.NS	101.85	120.05	0.44
AAPL	7.81	9.05	0.44
MSFT	12.73	15.31	-0.52
GOOGL	4.48	5.79	-0.27
AMZN	7.04	8.54	-0.16
NVDA	5.66	6.57	-0.22
CRWD	22.36	25.86	0.30
PLTR	7.44	8.35	0.50
FSLR	6.24	7.77	-0.73
CELH	1.07	1.32	-1.05
U	4.67	4.78	-6.60
HSBA.L	28.67	32.67	0.57
BP.L	13.18	19.32	0.52
ULVR.L	97.32	119.90	-0.18
AUTO.L	26.23	27.75	0.30
JDW.L	18.09	22.03	0.59
CER.L	61.97	74.18	0.56

Table 4: BPNN results with 'Close' as target

Ticker	MAE	RMSE	R2
RELIANCE.NS	0.533	0.674	1.000
TCS.NS	4.295	5.619	0.997
HDFCBANK.NS	1.227	1.438	0.998
PIDILITIND.NS	1.805	1.962	0.998
AUBANK.NS	1.462	1.736	0.999
IEX.NS	0.476	0.521	0.993
MAPMYINDIA.NS	1.843	1.927	0.998
LTTS.NS	12.190	13.117	0.993
AAPL	0.149	0.169	1.000
MSFT	0.672	0.722	0.996
GOOGL	0.167	0.189	0.999
AMZN	0.329	0.458	0.997
NVDA	0.214	0.349	0.997
CRWD	0.757	0.851	0.999
PLTR	0.203	0.255	0.999
FSLR	0.408	0.432	0.995
CELH	0.069	0.090	0.990
U	0.063	0.093	0.997
HSBA.L	0.812	0.883	1.000
BP.L	0.381	0.529	1.000
ULVR.L	6.489	7.428	0.996
AUTO.L	0.607	0.996	0.999
JDW.L	0.812	1.376	0.998
CER.L	0.698	0.799	1.000

Table 5: BPNN results with 'log return' as target and 'Close' extrapolated from log return predictions.

Ticker	MAE	RMSE	R2
AAPL	10.51	12.10	-0.16
AMZN	6.43	7.40	0.08
ASC.L	205.35	205.90	-91.22
AUBANK.NS	18.28	23.59	0.83
AUTO.L	11.67	17.87	0.72
BP.L	12.85	19.07	0.33
CELH	0.62	0.81	0.24
CRWD	21.00	24.89	0.37
FSLR	11.16	13.06	-3.69
GOOGL	4.20	5.05	0.07
HDFCBANK.NS	66.66	69.86	-3.61
HSBA.L	45.06	52.74	-0.24
IEX.NS	5.14	5.57	0.25
JDW.L	9.40	12.25	0.87
LTTS.NS	127.14	155.22	-0.04
MAPMYINDIA.NS	30.32	42.91	-0.05
MSFT	9.37	12.24	0.07
NVDA	3.70	5.13	0.27
PIDILITIND.NS	68.81	70.90	-2.78
PLTR	42.41	43.75	-12.53
RELIANCE.NS	26.85	34.86	0.68
TCS.NS	58.93	75.77	0.27
U	4.03	4.48	-5.37
ULVR.L	73.89	105.73	0.11

Table 6: LSTM results with 'Close' as target variable.

Ticker	MAE	RMSE	R2
RELIANCE.NS	19.73	27.15	0.79
TCS.NS	38.15	52.84	0.71
HDFCBANK.NS	12.53	15.69	0.78
PIDILITIND.NS	14.54	17.54	0.80
AUBANK.NS	14.08	18.61	0.90
IEX.NS	2.24	2.74	0.82
MAPMYINDIA.NS	22.94	27.60	0.59
LTTS.NS	89.95	116.47	0.47
AAPL	6.62	9.43	0.39
MSFT	7.57	10.79	0.24
GOOGL	3.54	4.56	0.21
AMZN	5.71	7.60	0.08
NVDA	4.25	5.82	0.04
CRWD	12.78	17.52	0.68
PLTR	4.25	5.46	0.78
FSLR	5.12	5.94	-0.01
CELH	0.77	1.07	-0.35
U	0.82	1.16	0.56
HSBA.L	17.89	26.51	0.72
BP.L	9.47	13.43	0.77
ULVR.L	72.65	91.71	0.31
AUTO.L	12.31	15.85	0.77
JDW.L	8.75	10.58	0.91
ASC.L	9.67	11.78	0.68

Table 7: LSTM results with 'log return' as target and 'Close' extrapolated from the predicted log returns.

4.3.4 Volatility Forecasting

a) SVR gave the best result for volatility prediction among GARCH (1,1), LR, RF, SVR, XGBR, BPNN and LSTM methods.

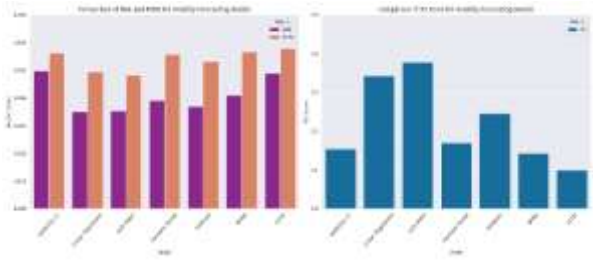

Figure 10: The performance of different models in volatility prediction. The left plot shows metrics MAE (brown) and RMSE (purple) and the right plot shows their R^2 score. SVR (third bar from left) has lowest MAE/RMSE and highest R^2 score.

Realized volatility is extremely noisy. SVR considers errors less than a given ϵ as zero and ignores the noise. Thus, SVR learns the underlying structure rather than overfitting the daily noise.

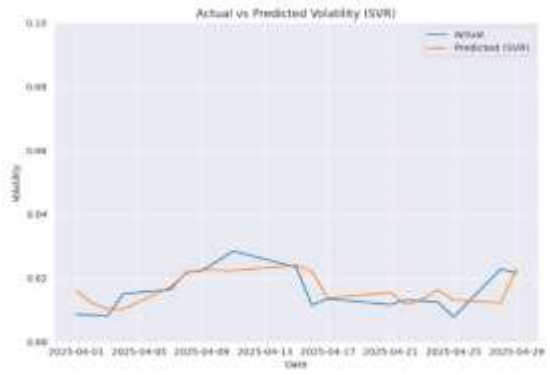

Figure 11: Predicted volatility by SVR vs actual volatility

b) Daily returns trump weekly and monthly returns in volatility prediction.

Frequency	MAE	RMSE	R2
Daily	0.00359	0.00456	0.21612
Weekly	0.00848	0.00894	-1.40882
Monthly	0.01507	0.01507	NaN

Table 8: Error of SVR model for different frequencies

This fact has been established by (Merton, 1980) that while you need a *long history* (many years) to estimate the **Mean** (Return), you need *high frequency* (many observations per year) to estimate the **Variance** (Risk).

c) Finally, the volatility of the given set of 24 stocks was predicted using SVR (RBF) with daily returns and the stocks were classified into low, moderate, high and extremely high daily volatile.

However, this classification is in no way an estimation of *market risk* which depends on several factors that are not part of this study. These results simply show which stocks have what degree of daily volatility. An investor can make certain decision on this information though he is advised to research more instead of simply relying on these results.

Low	Moderate	High	Extremely High
RELIANCE.NS	MAPMYINDIA.NS	AUBANK.NS	AAPL
TCS.NS	LTTS.NS	MSFT	NVDA
HDFCBANK.NS	HSBA.L	GOOGL	CRWD
PIDILITIND.NS	BP.L	AMZN	PLTR
IEX.NS	AUTO.L	CELH	FSLR
ULVR.L	JDW.L	ASC.L	U

Table 9: Classification of stocks on the basis of their volatility

5. Conclusion

Thus conclude the study of physical (GBM), statistical (AR, ARIMA, GARCH), ML (LR, SVR, RF, XGBR) and deep learning frameworks such as BPNN and LSTM. It was found that

i. Exponential Moving Average (EMA) is the best statistical model for forecasting as it gives high weight to the recent observations.
ii. Among the ML models, Linear Regression provides the best prediction of 'Close' price from the past data by assigning a high coefficient to the most recent observation.
iii. BPNN provide excellent prediction when modelled on returns. LSTM has great potential in price prediction if it is provided with big data.
iv. SVR with RBF kernel gives highly accurate volatility prediction when trained on a high frequency data. The results were not so promising when the interval of data collection was increased.

This is just the tip of the iceberg. A great deal of research has been conducted in this subject. Tsai & Wang have proved that a hybrid model built by joining seemingly different ML techniques gives better results. Soni et al. and Sonkavde et al. have corroborated this by experimenting with different models. WU et al. further expand this idea by building a hybrid model of ANNs. This is an avenue that I'd like to explore more. The transformers which are more powerful deep learning tools than neural networks are also used for this purpose. Temporal Fusion Transformers (TFTs) have been employed by Hu stock price prediction and he found positive results. I intend to study TFTs further.

Another exciting field of research is *Sentiment Analysis for volatility prediction*. The volatility in the market is shaped by the sentiment in market. Liu et al. present a method of volatility prediction with RNNs. Deveikyte et al. analyse

social media buzz to understand market sentiment. I'd like to study the social media sentiment on markets in detail using NLP and neural network in future.

Acknowledgement

I'm truly grateful to you sir for your guidance and mentorship. I would have been lost and could have never directed my focus without you. Thanks again.

PUNEET MISHRA

DA25C016

References

Abidin, S. N. Z., & Jaffar, M. M. (2014). Forecasting Share Prices of Small Size Companies in Bursa Malaysia Using Geometric Brownian Motion. *Applied Mathematics & Information Sciences*, *8*(1), 107–112. https://doi.org/10.12785/amis/080112

Akaike, H., & Kitagawa, G. (1999). *The Practice of Time Series Analysis* (H. Akaike & G. Kitagawa, Eds.). Springer New York. https://doi.org/10.1007/978-1-4612-2162-3

Ariyo, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Stock Price Prediction Using the ARIMA Model. *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, 106–112. https://doi.org/10.1109/UKSim.2014.67

Bishop, C. M. . (2009). *Pattern recognition and machine learning*. Springer Science + Business Media.

Chang, C.-L., Sriboonchitta, S., & Wiboonpongse, A. (2009). Modelling and forecasting tourism from East Asia to Thailand under temporal and spatial aggregation. *Mathematics and Computers in Simulation*, *79*(5), 1730–1744. https://doi.org/10.1016/j.matcom.2008.09.006

Christensen, K., Siggaard, M., & Veliyev, B. (2023). A Machine Learning Approach to Volatility Forecasting. *Journal of Financial Econometrics*, *21*(5), 1680–1727. https://doi.org/10.1093/jjfinec/nbac020

Engle, R. (2001). GARCH 101: The Use of ARCH/GARCH Models in Applied Econometrics. *Journal of Economic Perspectives*, *15*(4), 157–168. https://doi.org/10.1257/jep.15.4.157

Goodfellow, Ian., Bengio, Yoshua., & Courville, Aaron. (2017). *Deep learning*. The MIT Press.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hu, X. (2021). Stock Price Prediction Based on Temporal Fusion Transformer. *2021 3rd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, 60–66. https://doi.org/10.1109/MLBDBI54094.2021.00019

Ioffe, S., & Szegedy, C. (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*.

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *An Introduction to Statistical Learning*. Springer International Publishing. https://doi.org/10.1007/978-3-031-38747-0

Ke, Z., Xu, J., Zhang, Z., Cheng, Y., & Wu, W. (2024). A Consolidated Volatility Prediction with Back Propagation Neural Network and Genetic Algorithm. *2024 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML)*, 1671–1675. https://doi.org/10.1109/ICICML63543.2024.10958142

Liu, Y. (2019). Novel volatility forecasting using deep learning–Long Short Term Memory Recurrent Neural Networks. *Expert Systems with Applications*, *132*, 99–109. https://doi.org/10.1016/j.eswa.2019.04.038

Liu, Y., Qin, Z., Li, P., & Wan, T. (2017). *Stock Volatility Prediction Using Recurrent Neural Networks with Sentiment Analysis* (pp. 192–201). https://doi.org/10.1007/978-3-319-60042-0_22

Mailinda, I., Ruldeviyani, Y., Tanjung, F., Mikoriza T, R., Putra, R., & Fauziah A, T. (2021). Stock Price Prediction During the Pandemic Period with the SVM, BPNN, and LSTM Algorithm. *2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, 189–194. https://doi.org/10.1109/ISRITI54043.2021.9702865

Malkiel, B. Gordon. (2015). *A random walk down Wall Street : the time-tested strategy for successful investing*. W.W. Norton & Company.

Mashadihasanli, T. (2022). Stock Market Price Forecasting Using the Arima Model: an Application to Istanbul, Turkiye. *Journal of Economic Policy Researches / İktisat Politikası Araştırmaları Dergisi*, *9*(2), 439–454. https://doi.org/10.26650/JEPR1056771

Merton, R. C. (1980). On estimating the expected return on the market. *Journal of Financial Economics*, *8*(4), 323–361. https://doi.org/10.1016/0304-405X(80)90007-0

MOON, K.-S., & KIM, H. (2019). Performance of Deep Learning in Prediction of Stock Market Volatility. *ECONOMIC COMPUTATION AND ECONOMIC CYBERNETICS STUDIES AND RESEARCH*, *53*(2/2019), 77–92. https://doi.org/10.24818/18423264/53.2.19.05

Niu, Z., Wang, C., & Zhang, H. (2023). Forecasting stock market volatility with various geopolitical risks categories: New evidence from machine learning models. *International Review of Financial Analysis*, *89*, 102738. https://doi.org/10.1016/j.irfa.2023.102738

Sharma, P., & Vipul. (2015). Forecasting stock index volatility with GARCH models: international evidence. *Studies in Economics and Finance*, *32*(4), 445–463. https://doi.org/10.1108/SEF-11-2014-0212

Soni, P., Tewari, Y., & Krishnan, D. (2022). Machine Learning Approaches in Stock Price Prediction: A Systematic Review. *Journal of Physics: Conference Series*, *2161*(1), 012065. https://doi.org/10.1088/1742-6596/2161/1/012065

Sonkavde, G., Dharrao, D. S., Bongale, A. M., Deokate, S. T., Doreswamy, D., & Bhat, S. K. (2023). 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*, *11*(3), 94. https://doi.org/10.3390/ijfs11030094

Tsai, C. F., & Wang, S. P. (2009). Stock Price Forecasting by Hybrid Machine Learning Techniques. *Proceedings of the International Multiconference of Engineers and Computer Scientists*, 60.

WU, C., LUO, P., LI, Y., & Chen, K. (2015). Stock Price Forecasting: Hybrid Model of Artificial Intelligent Methods. *Engineering Economics*, *26*(1), 40–48. https://doi.org/10.5755/j01.ee.26.1.3836

Zhang, Z., & He, K. (2023). Prediction of Stock Price Using LSTM, MA and EMA Models. *2023 IEEE International Conference on Control, Electronics and Computer Technology (ICCECT)*, 191–194. https://doi.org/10.1109/ICCECT57938.2023.10140454