International Journal of Computer Sciences and Engineering
Vol.**11**, Special Issue.**1**, pp.**12-28**, November **2023**
ISSN: 2347-2693 (Online)
Available online at: www.ijcseonline.org

ISSN: 2347-2693 (E)

Research Paper

Comparative Study of Time Series Algorithms on Stock Price Forecasting

Mehul Kundu[1*], Arpan Bose[2], Debasmita Guha[3], Romit Das[4], Aftab Alam[5]

Corresponding Author: edu.mehul@gmail.com

Abstract: Gauging the pulse of political climes and global tides alike, the stock exchange carries considerable consequence. Given the capricious manner by which the apparatus maneuvers, present technological means and trade tenets forbid accurately foretelling how planetary perturbations may sway share values over extended time. Of late, nonetheless, informatic techniques and artificially adroit algorithms have enabled scrutinizing episodes and occurrences that surface symbiotically, prognosticating the trajectory of equity prices for most, if not all, corporations in qualifying bourses.
Past the fluctuating cultural zeitgeists and mercantile vicissitudes defining our epoch's global political topography, prognosticating distant futures proves an elusive enterprise.
Our objective henceforth shall be to prognosticate the quotidian inclination of equity premiums in the stock bazaar with the utmost precision achievable. The Stacking Ensemble model is a prevalent and trustworthy fashion to anticipate chronological successions. We shall explain why we opt for it over and above the other techniques adduced, akin to Meta Prophet, and the LSTM modus operandi, posterior to assaying. We manipulate a combination exemplar to ascertain the weighted norm of all in-sample data. In this exploration, we endeavored to disparate things with the stock valuations of multiple shareholding fully remunerated ordinary portions and thereafter effectuated predictions about stock valuations.

Keywords: Stock price forecast, ARIMA, SARIMA, LSTM, Meta Prophet, GRU, Kalman Filter, Ensemble

1. Introduction

1.1 Time Series

A grouping of data points that are compiled in a sequential manner is commonly known as a time series. Information that has been collected and documented progressively, often labeled as time series data, possesses the capability to be employed in detecting inclinations or designs of conduct.

The four components of a time series are:
Level: This is the baseline or average value of the series. It represents the expected value of the series over time, without any upward or downward trends or seasonal fluctuations.

Trend: This component refers to the long-term direction of the series. The depiction pertains to the all-encompassing formation of a sequence throughout an extensive duration, akin to either an upward or downward inclination.

Seasonality: The facet of seasonality denotes the reoccurring, cyclical oscillations present in the series over a predetermined time frame including but not limited to days, weeks or months. The fluctuation in regularity, known as seasonality, can be influenced by multiple determinants consisting of distinct weather patterns, holiday seasons or customary occasions.

Noise: This component refers to random fluctuations or irregularities in the series that cannot be explained by the other components. This denotes the indeterminate aberration in a sequence caused by fortuitous occurrences, inaccuracies in measurement, or other instigating circumstances.
It's important to note that trend and seasonality components are optional since series can be stationary.

2. Related Work

ARIMA (Autoregressive Integrated Moving Average):
ARIMA models have been extensively studied and applied in various fields, particularly in time series forecasting. The literature on ARIMA models spans several decades and covers a wide range of topics, including model development, estimation techniques, diagnostic tools, and practical applications.

One of the foundational works in ARIMA modeling is the classic paper by Box and Jenkins (1970), titled "Time Series Analysis: Forecasting and Control," where they introduced the concept of ARIMA models and presented the Box-Jenkins methodology for model identification, estimation, and diagnostic checking.

Since then, numerous researchers have made significant contributions to the ARIMA literature. For example, Hamilton (1994) in his book "Time Series Analysis" provides a comprehensive treatment of ARIMA models, including model selection, estimation, and inference.

GRUs were proposed by Kyunghyun Cho, et al., in 2014 as an alternative to LSTMs. GRUs simplify the LSTM architecture, having two gates (update and reset) compared to the LSTM's three gates.

Cho, K., et al. (2014). "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches." This paper introduced GRUs and discussed their use in machine translation tasks.

Chung, J., et al. (2014). "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling." This paper offers empirical evaluations of GRUs in sequence modeling tasks.

LSTMs, introduced by Hochreiter and Schmidhuber in 1997, are a type of recurrent neural network (RNN) designed to avoid the problem of vanishing gradients. They use a cell state and three gates (input, forget, and output) to control the flow of information.

Hochreiter, S., & Schmidhuber, J. (1997). "Long Short-Term Memory." This is the seminal paper that introduced LSTMs.

Gers, F. A., Schmidhuber, J., & Cummins, F. (1999). "Learning to Forget: Continual Prediction with LSTM." This paper introduced the concept of the forget gate in LSTMs.

The Kalman filter, introduced by Rudolf E. Kálmán in 1960, is an algorithm that uses a series of measurements observed over time and produces estimates of unknown variables by estimating a joint probability distribution over the variables for each timeframe.

Kalman, R. E. (1960). "A New Approach to Linear Filtering and Prediction Problems." This is the original paper introducing the Kalman filter.

Maybeck, P. S. (1979). "Stochastic Models, Estimation, and Control, Volume 1." This book provides a comprehensive understanding of the Kalman filter and its applications.

3. Theory/Calculation

Since the market is typically closed on weekends and major holidays depending on the region, stock data (if considered daily) has regular intervals in it unlike any other time series. Therefore, we must ensure consistency before making any predictions using stock data. Predictions can only be made with confidence when all variables are consistent.

However, the key to success is to identify reliable and relevant information and to make informed decisions based on the research and analysis. Forecasting stock performance can be a challenging task, but it is essential in order to make informed investment decisions.

Finding the missing data
In order to determine the date range, we started by taking the first and last dates in the series. Afterward, discover how many dates are present and how many are absent. (It will change depending on the stock and region.)

Filling in the missing data
One frequently encounters the interpolate() function within data analysis libraries such as Pandas or NumPy, which is a pre-existing mode of operation. It serves to fill gaps or missing time series values, making it extremely useful for this purpose. In order to approximate the absent data in a series over time, diverse techniques for interpolation like polynomial, cubic and linear are utilized within this function. By relying on known values present in the given set of information points, these methods provide estimates that bridge any gaps or missing pieces found therein. The function can therefore be used on time series data - both regular and irregular. Selecting parameters for the interpolation method is largely dependent upon the data's unique characteristics. It became possible for us to realize a more comprehensive dataset by virtue of the function called .interpolate(). In so doing, we managed to infill our time series with missing data hence enabling us to generate plausible predictions using our forecasting models. The effectiveness of the approximation technique hinges on several factors, notably the preferred interpolation approach, the frequency and design of incomplete entries, as well as idiosyncrasies within temporal data records.

The working of the interpolation function can be summarized as follows:
Identify Missing Data: The first step is to identify the missing data points in the time series dataset. These are typically denoted by NaN (Not a Number) or any other designated missing value indicator.

Select Interpolation Method: Next, an appropriate interpolation method is chosen based on the characteristics of the data and the specific requirements of the analysis. Common interpolation methods include linear interpolation, polynomial interpolation, spline interpolation, and nearest neighbor interpolation. Each method has its own assumptions and mathematical algorithms for estimating missing values.

Determine Interpolation Window: The interpolation window refers to the range of adjacent data points used to estimate the missing value. It can vary based on the specific interpolation method and the nature of the time series data. For instance, linear interpolation uses the two nearest observed data points on either side of the missing value to estimate its value.

Perform Interpolation: The interpolation function applies the selected interpolation method within the interpolation window to estimate the missing values. The algorithm uses mathematical calculations, such as linear equations, polynomial equations, or spline functions, to determine the estimated values based on the neighboring observed data points.

Replace Missing Values: Once the missing values are estimated through interpolation, they are substituted in the time series dataset, replacing the original missing value indicators. This process fills in the gaps in the time series data, ensuring a continuous and complete dataset.

Assess Interpolation Accuracy: It is important to evaluate the accuracy and reliability of the interpolated values. This can be done by comparing the interpolated values to any available ground truth or by assessing the overall consistency and smoothness of the filled-in time series. It is crucial to consider the limitations and assumptions of the chosen interpolation method and the impact it may have on the accuracy of the interpolated values.

By utilizing the interpolation function, missing data points in a time series can be estimated and filled in, enabling a more complete and continuous dataset for further analysis and forecasting. It provides a practical and efficient approach to handle missing values in time series data and ensures that the resulting dataset maintains the temporal order and integrity required for meaningful analysis and decision-making.
AUTO SARIMA

ARIMA
The ARIMA model is a widely used time series forecasting model that combines autoregression (AR), differencing (I), and moving average (MA) components. It is effective in capturing both short-term and long-term patterns in the data.
The ARIMA model is denoted as ARIMA(p, d, q), where:

p represents the order of the autoregressive component, which captures the linear relationship between the current value and previous values of the time series.

d represents the order of differencing, which is the number of times the time series needs to be differenced to achieve stationarity.

q represents the order of the moving average component, which captures the relationship between the error terms and the lagged values of the time series.

The general equation for an ARIMA(p, d, q) model can be written as:

$$Y\_t = c + \Phi_1 Y\_(t-1) + \Phi_2 Y\_(t-2) + ... + \Phi\_p Y\_(t-p) + \theta_1 \varepsilon\_(t-1) + \theta_2 \varepsilon\_(t-2) + ... + \theta\_q \varepsilon\_(t-q) + \varepsilon\_t \quad \ldots\ldots\ldots\ldots..(1)$$

where:

Y_t is the value of the time series at time t.
c is a constant term.
$\Phi_1, \Phi_2, ..., \Phi\_p$ are the autoregressive coefficients.
$\theta_1, \theta_2, ..., \theta\_q$ are the moving average coefficients.
ε_t is the error term at time t.
ARIMA models are primarily used for univariate time series forecasting, where only the historical values of a single variable are considered.

SARIMA (Seasonal ARIMA)
The SARIMA model extends the ARIMA model by incorporating the seasonal component. It is suitable for time series data that exhibit seasonality, where patterns repeat at regular intervals.

The SARIMA model is denoted as SARIMA(p, d, q)(P, D, Q, s), where:

(p, d, q) represents the non-seasonal order of the AR, I, and MA components, respectively.
(P, D, Q, s) represents the seasonal order of the AR, I, MA components, and the length of the seasonal period, respectively.
The general equation for a SARIMA(p, d, q)(P, D, Q, s) model can be written as:

$$Y\_t = c + \Phi_1 Y\_(t-1) + \Phi_2 Y\_(t-2) + ... + \Phi\_p Y\_(t-p) + \theta_1 \varepsilon\_(t-1) + \theta_2 \varepsilon\_(t-2) + ... + \theta\_q \varepsilon\_(t-q) + \Phi_1 s Y\_(t-s) + \Phi_2 s Y\_(t-2s) + ... + \Phi\_P s Y\_(t-Ps) + \theta_1 s \varepsilon\_(t-s) + \theta_2 s \varepsilon\_(t-2s) + ... + \theta\_Q s \varepsilon\_(t-Qs) + \varepsilon\_t \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots..(2)$$

The seasonal component captures the periodic patterns in the data, and its inclusion in the model enables better forecasting accuracy for data with seasonal fluctuations.

SARIMA:
MAE: 141.8403469611237
MSE: 22660.744501731195
RMSE: 150.5348614166539

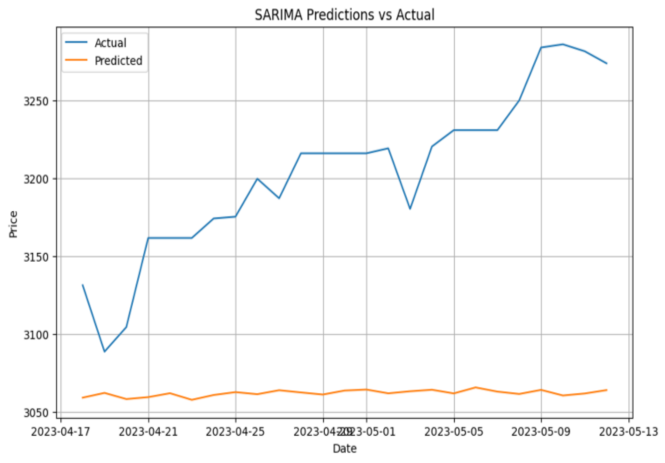

Fig 3.1 Sarima Prediction vs Actual on
TCS.BO - Tata Consultancy Services Limited - BSE - Mumbai INR

LSTM
Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture designed to overcome the limitations of traditional RNNs in capturing long-term dependencies in sequential data. LSTMs are particularly effective in time series forecasting tasks due to their ability to retain and utilize information over extended time intervals. They consist of several key components:

Input Gate:
The input gate determines the relevance of new input to the current cell state. It is computed as:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.....(4)$$

where i_t is the input gate activation, h_{t-1} is the previous hidden state, x_t is the current input, W_i and b_i are the learnable weights and biases associated with the input gate, and σ represents the sigmoid activation function.

Forget Gate:
The forget gate determines what information from the previous cell state should be forgotten. It is calculated as:

$f_t = \sigma(Wf \cdot [h_{t-1}, x_t] + bf)$ ………………………………..(5)

where f_t is the forget gate activation, Wf and bf are the learnable weights and biases associated with the forget gate.

Cell State Update:
The cell state is updated by incorporating new input and selectively forgetting previous information. It involves the following steps:

$\tilde{C}_t = \tanh(Wc \cdot [h_{t-1}, x_t] + bc)$ ……………………..….(6)

where $\tilde{C}_t$ represents the candidate cell state, Wc and bc are the learnable weights and biases associated with the candidate cell state.

Cell State Update Gate:
The cell state update gate determines how much of the candidate cell state should be added to the current cell state. It is calculated as:

$g_t = \sigma(Wg \cdot [h_{t-1}, x_t] + bg)$ ………………………………..(7)

where g_t is the cell state update gate activation, Wg and bg are the learnable weights and biases associated with the cell state update gate.

Cell State:
The cell state is updated by combining the previous cell state with the candidate cell state, controlled by the forget gate and the cell state update gate:

$C_t = f_t * C_{t-1} + i_t * g_t$ ……………………………………...(8)

where C_t represents the updated cell state.

Output Gate:
The output gate determines the relevance of the current cell state to the output. It is calculated as:

$o_t = \sigma(Wo \cdot [h_{t-1}, x_t] + bo)$ ……………………………….(9)

where o_t is the output gate activation, Wo and bo are the learnable weights and biases associated with the output gate.

Hidden State:
The hidden state is calculated by applying the output gate to the updated cell state:

$h_t = o_t * \tanh(C_t)$ ………………………………………...(10)

where h_t represents the hidden state.

LSTMs employ these components to update and propagate information over time, allowing them to capture long-term dependencies in sequential data. By selectively remembering and forgetting information, LSTMs can effectively handle time series data with complex patterns and temporal dependencies.

LSTM:
MAE: 64.59958984375
MSE: 5476.223884849548
RMSE: 74.00151272000828

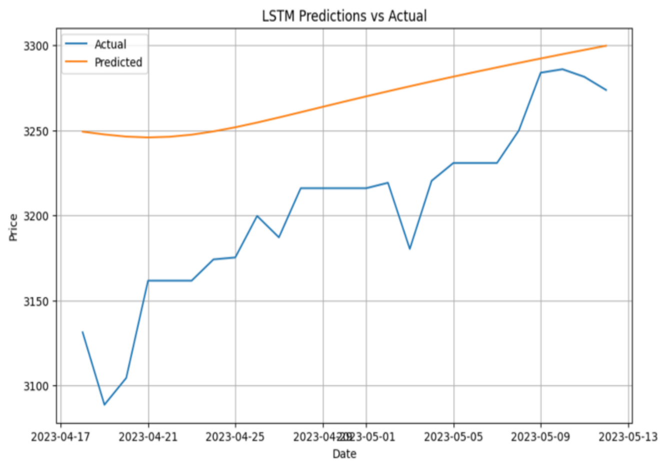

Fig 3.2 LSTM Prediction vs Actual on
TCS.BO - Tata Consultancy Services Limited - BSE - Mumbai INR

GRU
The Gated Recurrent Unit (GRU) model is a variant of recurrent neural network (RNN) that is designed to capture and learn long-term dependencies in sequential data. It is particularly effective in tasks such as time series forecasting, natural language processing, and speech recognition. The GRU model achieves this by introducing gating mechanisms that selectively update and reset information within the hidden state.

The GRU model consists of two main gates: the update gate (z) and the reset gate (r). These gates control the flow of information and determine how much of the past information to forget and how much of the new information to incorporate.

The update gate (z) is responsible for deciding how much of the previous hidden state (h) should be combined with the candidate activation (h~) to produce the new hidden state (h_t). The update gate is computed as follows:

z_t = sigmoid(W_z * x_t + U_z * h_(t-1)) ………….….(11)

The reset gate (r) determines how much of the previous hidden state (h) should be forgotten or ignored when computing the candidate activation. The reset gate is calculated as follows:

r_t = sigmoid(W_r * x_t + U_r * h_(t-1)) ……………...(12)

To compute the candidate activation (h~), which contains the new information to be added to the hidden state, the reset gate (r) is used to determine which parts of the previous hidden state (h) to reset. The candidate activation is computed as follows:

h~_t = tanh(W_h * x_t + U_h * (r_t ⊙h_(t-1))) ….(13)

where ⊙denotes element-wise multiplication.

Finally, the new hidden state (h_t) is computed by combining the update gate (z) and the previous hidden state (h) with the candidate activation (h~):

h_t = (1 - z_t) ⊙h_(t-1) + z_t ⊙h~_t ….(14)

The GRU model iteratively updates the hidden state based on the input sequence, allowing it to capture and remember relevant information over long sequences. The final hidden state can then be used for various tasks, such as making predictions or generating outputs.

The parameters W_z, U_z, W_r, U_r, W_h, and U_h are learnable weight matrices that are optimized during the training process using techniques like backpropagation through time (BPTT) and gradient descent to minimize the model's loss function.

GRU:
MAE: 36.053271484375
MSE: 2143.8972117114067
RMSE: 46.30223765339432

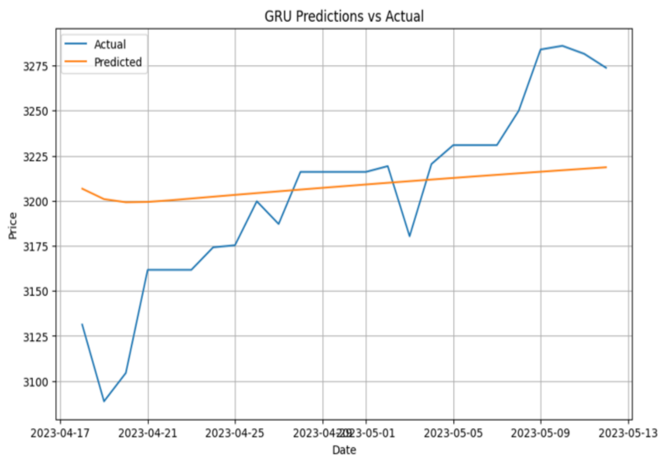

Fig 3.3 GRU Prediction vs Actual on
TCS.BO - Tata Consultancy Services Limited - BSE - Mumbai INR

PROPHET

The Prophet model is a time series forecasting model developed by Facebook's Core Data Science team. It is designed to handle the unique characteristics of time series data, such as seasonality, trend changes, and outliers. The Prophet model incorporates multiple components to capture these patterns:

Trend Component:
The Prophet model assumes that the trend in the time series can be modeled as a non-linear growth function. It uses a piecewise linear or logistic growth model to capture changes in the trend over time. The trend component is represented by the equation:

g(t) = k(t) + Σ(n=1 to N) [a_n * f_n(t)] ….. (3)

where g(t) represents the modeled trend at time t, k(t) is the baseline growth rate, a_n represents the individual seasonal components, and f_n(t) represents the seasonal functions.

Seasonality Component:
Prophet accounts for seasonality patterns in the data by incorporating multiple seasonal components, such as yearly, weekly, and daily seasonality. It models these components using Fourier series expansions to capture their periodic nature.

Yearly Seasonality:
The yearly seasonality component captures the annual patterns in the data. It is modeled using a Fourier series expansion with user-defined parameters.

Weekly and Daily Seasonality:
Prophet also incorporates weekly and daily seasonality components using similar Fourier series expansions. These components capture the recurring patterns that occur within a week or a day.

Holiday Effects:
The Prophet model allows the inclusion of known holidays and their impact on the time series. Users can provide a list of holidays and their corresponding dates, and the model incorporates these effects into the forecast.

Uncertainty Estimation:
Prophet provides uncertainty estimation by modeling the inherent noise and irregularities in the time series. It uses a Monte Carlo simulation approach to generate a range of possible future scenarios.

Model Fitting:
The Prophet model fits the time series data by optimizing the model parameters using a procedure known as Markov chain Monte Carlo (MCMC) sampling or a faster optimization algorithm called L-BFGS. The fitting process minimizes the difference between the observed data and the model's predictions.

Prophet:
MAE: 54.78821932346668
MSE: 4385.774347117845
RMSE: 66.2251791021953

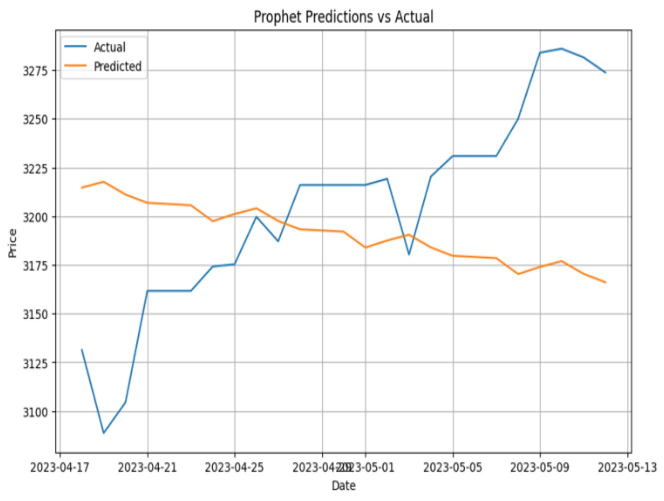

Fig 3.4 PROPHET Prediction vs Actual on
TCS.BO - Tata Consultancy Services Limited - BSE - Mumbai INR

KALMAN FILTER

The Kalman Filter is a recursive algorithm used for estimating the state of a dynamic system based on noisy observations. It is widely applied in various fields, including time series analysis, control systems, and navigation. The Kalman Filter consists of several key components:

State Transition Equation:
The state transition equation describes how the system evolves over time. It is typically represented as:

$x_t = F_{t-1} * x_{t-1} + B_{t-1} * u_{t-1} + w_{t-1}$ ……………………….(15)

where x_t is the state vector at time t, F_{t-1} is the state transition matrix, B_{t-1} is the control input matrix, u_{t-1} is the control input, and w_{t-1} is the process noise.

Observation Equation:
The observation equation relates the system's state to the observed measurements. It is represented as:

$z_t = H_t * x_t + v_t$ …………………………………………..(16)

where z_t is the observed measurement at time t, H_t is the observation matrix, x_t is the state vector, and v_t is the measurement noise.

State Estimate and Covariance:
The Kalman Filter maintains an estimate of the system's state and its associated uncertainty, called the state estimate ($\hat{x}_t$) and the state covariance (P_t). These estimates are updated based on the observed measurements and the predicted state.

Kalman Gain:
The Kalman Gain (K_t) determines the balance between the predicted state and the observed measurement. It is calculated as:

$K_t = P_{t-1} * H_t^T * (H_t * P_{t-1} * H_t^T + R_t)^{-1}$ ………………...(17)

where P_{t-1} is the previous state covariance, H_t^T is the transpose of the observation matrix, R_t is the measurement noise covariance.

State Update:
The state update step incorporates the observed measurement to refine the state estimate. It is computed as:

$\hat{x}_t = x_{t-1} + K_t * (z_t - H_t * x_{t-1})$ …………………………...(18)

Covariance Update:
The covariance update step updates the state covariance based on the Kalman Gain. It is calculated as:

$P_t = (I - K_t * H_t) * P_{t-1}$ …………………………………..(19)

where I is the identity matrix.

Prediction Step:
The prediction step predicts the state and covariance for the next time step using the state transition equation. It is computed as:

$x_{t+1+} = F_t * \hat{x}_t + B_t * u_t$ …………………………………..(20)

$P_{t+1+} = F_t * P_t * F_t^T + Q_t$ ………………………………….(21)

where Q_t is the process noise covariance.

The Kalman Filter iteratively updates the state estimate and covariance based on the observed measurements, and then predicts the state and covariance for the next time step. This recursive process allows for accurate estimation of the system's state even in the presence of noisy measurements and uncertain dynamics.

Kalman Filter:
MAE: 89.33440507567819
MSE: 8921.84962199587
RMSE: 94.45554309830563

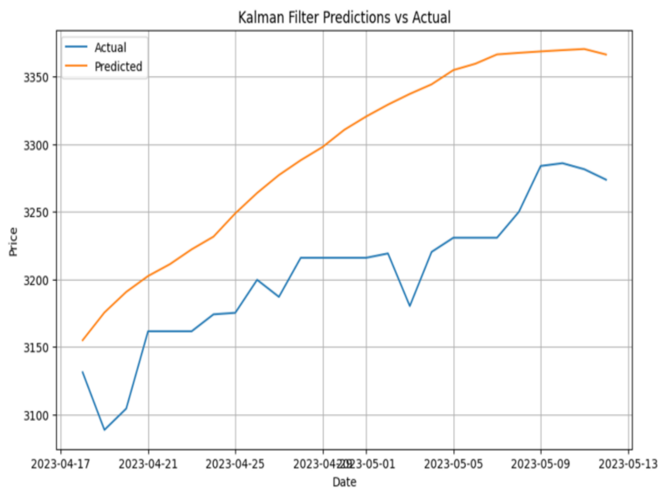

Fig 3.5 KALMAN Prediction vs Actual on
TCS.BO - Tata Consultancy Services Limited - BSE - Mumbai INR

ENSEMBLE MODELS

AVERAGE

The Average model, also known as the Simple Average model, is a basic ensemble model used for time series forecasting. In this approach, we combine the predictions from multiple individual forecasting models by taking their average. The Average model does not involve complex equations or components. The process can be summarized as follows:

Model Selection: We select a set of individual forecasting models to include in the ensemble. These models can be diverse in nature, such as ARIMA, exponential smoothing, or neural networks.

Individual Model Training: We train each individual model using historical time series data. The training process depends on the specific modeling technique used for each model. For example, ARIMA models require parameter estimation, while neural networks require training using backpropagation.

Individual Model Forecasting: We make predictions using each trained individual model on the test dataset or future time points of interest. Each model generates its own set of forecasts based on its learned patterns and characteristics.

Ensemble Combination: We combine the individual forecasts by calculating their average. For each time point, we sum the predictions from each model and divide by the total number of models:

$$\text{Forecast}(t) = (1/N) * \Sigma\, \text{Forecast}_n(t) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots..(22)$$

where Forecast(t) is the ensemble forecast at time t, N is the total number of individual models, and $\text{Forecast}_n(t)$ represents the forecast of the nth individual model at time t.

Evaluation and Performance: We assess the performance of the Average model using appropriate evaluation metrics such as mean squared error (MSE), mean absolute error (MAE), or root mean squared error (RMSE). We compare the ensemble's performance with the individual models and select the appropriate metrics based on our forecasting goals.

The Average model is a simple and intuitive ensemble approach that we can easily implement. It leverages the diversity of individual models to reduce bias and provide a more robust forecast. However, it does not consider the varying strengths or accuracies of each individual model. It assumes an equal weighting for all models, which may not be optimal in certain cases.

Average:
MAE: 28.41425993021154
MSE: 1409.217635076121
RMSE: 37.53954761416447

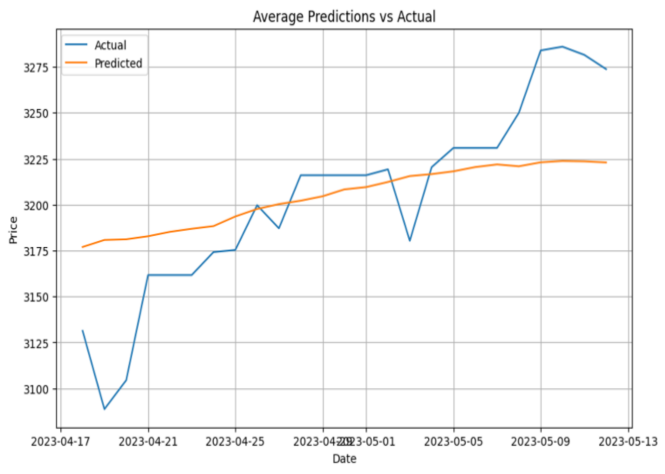

Fig 3.6 Average Prediction vs Actual on TCS.BO - Tata Consultancy Services Limited - BSE - Mumbai INR

WEIGHTED AVERAGE

The Weighted Average model is an ensemble approach we use for time series forecasting, where we combine the predictions from multiple individual models using weighted averages. This allows us to assign different importance or weights to each individual model based on their performance or credibility. The working of the Weighted Average model can be described as follows:

Model Selection: We select a set of individual forecasting models to include in the ensemble. These models can be diverse in nature, such as ARIMA, exponential smoothing, or machine learning algorithms.

Individual Model Training: We train each individual model using historical time series data. The training process varies depending on the specific modeling technique used for each model. For example, ARIMA models require parameter estimation, while machine learning models require training using appropriate algorithms.

Individual Model Forecasting: We make predictions using each trained individual model on the test dataset or future time points of interest. Each model generates its own set of forecasts based on its learned patterns and characteristics.

Weight Assignment: We assign weights to each individual model based on their performance or credibility. The weights can be determined using various methods, such as cross-validation, error metrics, or expert judgment. The weights should reflect the relative importance or reliability of each model in the ensemble.

Ensemble Combination: We combine the individual forecasts using weighted averages. For each time point, we multiply the prediction from each model by its corresponding weight, and then sum these weighted predictions to obtain the ensemble forecast:

$$\text{Forecast}(t) = \Sigma\, (\text{Weight}_n * \text{Forecast}_n(t)) \quad \ldots\ldots\ldots\ldots\ldots\ldots..(23)$$

where Forecast(t) is the ensemble forecast at time t, $Weight_n$ is the weight assigned to the nth individual model, and $Forecast_n(t)$ represents the forecast of the nth individual model at time t.

Evaluation and Performance: We assess the performance of the Weighted Average model using appropriate evaluation metrics such as mean squared error (MSE), mean absolute error (MAE), or root mean squared error (RMSE). We compare the ensemble's performance with the individual models and adjust the weights accordingly to optimize the overall forecasting accuracy.

The Weighted Average model provides us with flexibility in incorporating the strengths of individual models by assigning different weights. It allows us to capture the diversity of the ensemble members and adjust their contributions based on their performance. By properly assigning weights, the Weighted Average model can potentially enhance the overall forecasting accuracy and robustness in capturing the underlying patterns and trends in the time series data.

Weighted:
MAE: 30.59584069611674
MSE: 1497.569870700234
RMSE: 38.6984479107397

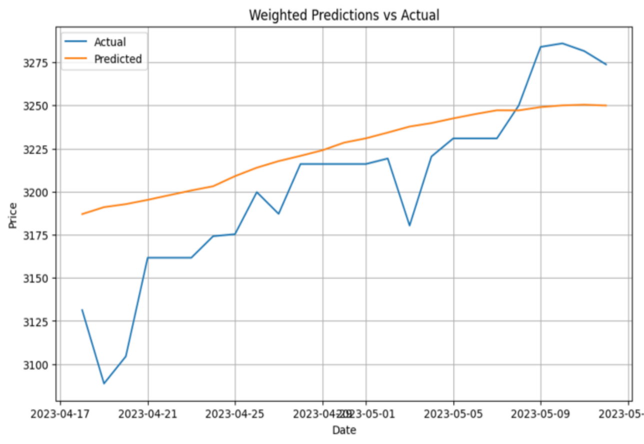

Fig 8.7 Weighted Average Prediction vs Actual on TCS.BO - Tata Consultancy Services Limited - BSE - Mumbai INR

STACKING

The Stacking Ensemble model is an advanced ensemble approach used for time series forecasting, which combines the predictions from multiple individual models by training a meta-model on their outputs. The working of the Stacking Ensemble model can be described as follows:

Model Selection: The ensemble team selects a diverse set of individual forecasting models to include in the ensemble. These models can be different in nature, such as ARIMA, exponential smoothing, neural networks, or random forests.

Training Phase:
a. Individual Model Training: Each team member trains their assigned individual model using historical time series data. The training process varies depending on the specific modeling technique used for each model. For example, ARIMA models require parameter estimation, while neural networks require training using backpropagation.
b. Individual Model Forecasting: After training, each team member generates predictions using their individual model on the test dataset or future time points of interest. Each model produces its own set of forecasts based on its learned patterns and characteristics.

Meta-Model Training:
a. Ensemble Input: The ensemble team combines the individual model forecasts into a matrix or a new feature set, denoted as X_meta. Each row of X_meta represents a specific time point, and each column corresponds to the prediction of an individual model.
b. Ensemble Target: The ensemble team defines a target variable, denoted as y_meta, which represents the true values or the actual observations at each time point.
c. Meta-Model Training: The team trains a meta-model, such as a linear regression or a neural network, using X_meta and y_meta as the input and target variables, respectively. The meta-model learns to capture the relationships between the individual model predictions and the actual values, aiming to make a more accurate forecast.

Ensemble Combination:
a. Individual Model Forecast Combination: Each team member generates their own individual model forecast on the test dataset or future time points.
b. Meta-Model Input: The ensemble team combines these individual model forecasts into a matrix or feature set, denoted as X_ensemble. Each row of X_ensemble corresponds to a specific time point, and each column represents the prediction of an individual model.
c. Meta-Model Forecast: The team uses the trained meta-model to make the final ensemble forecast by providing X_ensemble as input. The meta-model combines the individual model predictions based on their learned relationships to generate an ensemble forecast.

Evaluation and Performance: The ensemble team evaluates the performance of the Stacking Ensemble model using appropriate evaluation metrics such as mean squared error (MSE), mean absolute error (MAE), or root mean squared error (RMSE). They compare the ensemble's performance with the individual models and assess the effectiveness of the meta-model in improving forecasting accuracy.

The Stacking Ensemble model leverages the diverse predictions of individual models by training a meta-model to capture their relationships and generate a more accurate ensemble forecast. By combining the strengths of different models, the Stacking Ensemble model aims to improve forecasting accuracy and capture the underlying patterns and trends in the time series data.

Stacking:
MAE: 14.014427660820074
MSE: 282.12701948753875
RMSE: 16.79663714817757

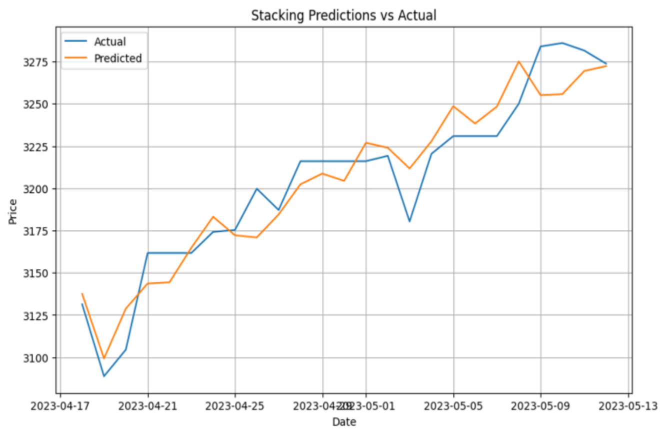

Fig 3.8 Stacking Prediction vs Actual on
TCS.BO - Tata Consultancy Services Limited - BSE - Mumbai INR

BAGGING

The Bagging Ensemble model is a powerful ensemble approach utilized for time series forecasting, where multiple individual models are trained on different subsets of the data through bootstrapping and their predictions are combined to make the final forecast. The working of the Bagging Ensemble model can be described as follows:

Model Selection: The ensemble team selects a diverse set of individual forecasting models to include in the ensemble. These models can be of various types, such as ARIMA, exponential smoothing, neural networks, or random forests.

Training Phase:
a. Bootstrapping: Each team member creates multiple bootstrap samples by randomly selecting data points with replacement from the original time series dataset. These bootstrap samples are used to train different instances of the individual models.
b. Individual Model Training: Each team member trains their assigned individual model using the respective bootstrap samples. The training process depends on the specific modeling technique used for each model. For example, ARIMA models require parameter estimation, while neural networks involve training through backpropagation.
c. Individual Model Forecasting: After training, each team member generates predictions using their individual model on the test dataset or future time points of interest. Each model produces its own set of forecasts based on its learned patterns and characteristics.

Ensemble Combination:
a. Individual Model Forecast Combination: Each team member generates their own individual model forecast on the test dataset or future time points.
b. Ensemble Forecast: The ensemble team combines the individual model forecasts by aggregating them, typically using the average or majority voting scheme. This aggregation helps to reduce the variability and biases present in the individual models, resulting in a more robust and accurate ensemble forecast.

Evaluation and Performance: The ensemble team evaluates the performance of the Bagging Ensemble model using appropriate evaluation metrics such as mean squared error (MSE), mean absolute error (MAE), or root mean squared error (RMSE). They compare the ensemble's performance with the individual models and assess the effectiveness of the bagging technique in improving forecasting accuracy.

The Bagging Ensemble model leverages the power of multiple individual models trained on different subsets of data to reduce overfitting and improve the overall forecasting performance. By combining the predictions of diverse models through bootstrap aggregation, the Bagging Ensemble model enhances the accuracy and stability of time series forecasts.

Bagging:
MAE: 36.147939453125
MSE: 2430.859116809368
RMSE: 49.30374343606546

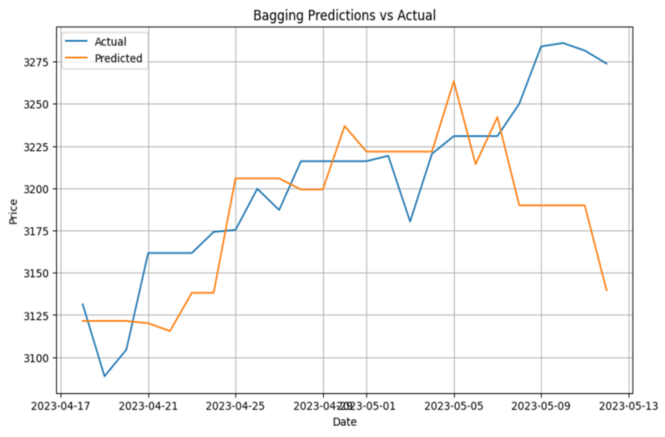

Fig 3.9 Bagging Prediction vs Actual on
TCS.BO - Tata Consultancy Services Limited - BSE - Mumbai INR

BOOSTING

The Boosting Ensemble model is a powerful ensemble approach used for time series forecasting, where multiple weak individual models are sequentially trained to correct the errors of their predecessors and improve the overall forecast. The working of the Boosting Ensemble model can be described as follows:

Model Selection: The ensemble team selects a set of weak individual models, often referred to as base learners, to include in the ensemble. These base learners can be simple models such as decision trees or linear regression models.

Training Phase:
a. Initial Model Training: The first base learner is trained on the original time series data. This initial model may not perform well, but it serves as a starting point for the boosting process.
b. Sequential Model Training: Subsequent base learners are trained in a sequential manner, with each model aiming to correct the errors made by its predecessors. During training, more emphasis is given to the data points that were previously mispredicted, allowing the subsequent models to focus on capturing the remaining patterns and improving overall accuracy.

c. Weighted Data Sampling: To emphasize the mispredicted data points, the ensemble team assigns weights to each observation in the training data. Initially, all weights are set to equal values, but as the boosting process proceeds, the weights are adjusted based on the errors made by the previous models. This creates a feedback mechanism that prioritizes the mispredicted data points in subsequent model training.
d. Model Weighting: After training each base learner, the ensemble team assigns weights to the individual models based on their performance. Models with lower errors are given higher weights, indicating their relative importance in the ensemble's final forecast.

Ensemble Combination:
a. Weighted Model Aggregation: The ensemble team combines the predictions of individual models by taking a weighted average of their forecasts. The weights assigned to each model reflect their performance and contribution to the overall ensemble forecast.

Evaluation and Performance: The ensemble team evaluates the performance of the Boosting Ensemble model using appropriate evaluation metrics such as mean squared error (MSE), mean absolute error (MAE), or root mean squared error (RMSE). They compare the ensemble's performance with the individual models and assess the effectiveness of the boosting technique in improving forecasting accuracy.

The Boosting Ensemble model leverages the sequential training of weak base learners to iteratively correct errors and improve forecasting accuracy. By assigning weights to models and emphasizing mispredicted data points, the Boosting Ensemble model focuses on capturing the remaining patterns and improving overall performance. The iterative nature of the boosting process allows the ensemble to adapt and learn from previous mistakes, resulting in a powerful forecasting model for time series data.

Boosting:
MAE: 36.147939453125
MSE: 2430.859116809368
RMSE: 49.30374343606546

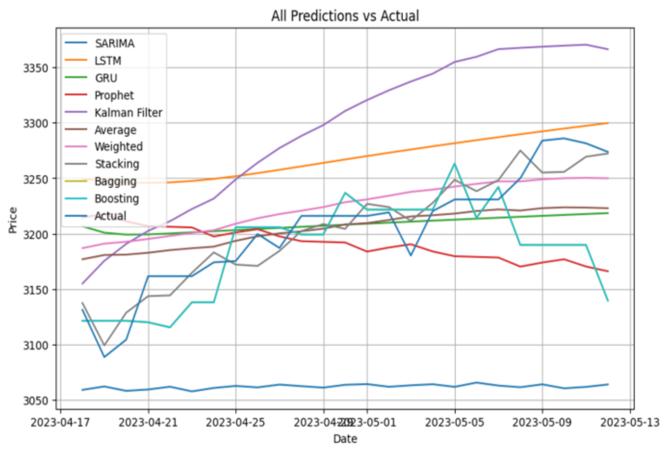

Fig 3.10 Boosting Prediction vs Actual on
TCS.BO - Tata Consultancy Services Limited - BSE - Mumbai INR

4. Experimental Method/Procedure/Design

4.1 Train - Test Split

There exist a multitude of models that can predict time series. A plethora of forecasting strategies are at our disposal when it comes to predicting sequences over continuous periods. Now, for stock price prediction, any time series algorithm should be adjusted to fit the models before being applied. Following the manipulation of data, 80% of said information served as training sets, with a further 20% being allocated to test sets in order to cross-analyze predicted outcomes and actual results. To compare the trained prediction, the forecast for the next 60 days was used.

4.2 Handling Missing Data

In order to proceed, it is essential that the information which has been acquired maintains uniformity throughout. In contrast to other time series, stock data has regular intervals because, depending on the region, the market is closed on weekends and major holidays. We have taken the first and last dates of the series to obtain the date range in order to accomplish that. The number of available dates and the number of missing dates were then determined. It is most advisable to complete any gaps in knowledge by seeking the nearest corresponding data. Voids in information can be addressed through identifying and utilizing its closest approximation available. Optimal resolution of incomplete facts requires strategic identification followed by subsequent application of relevant close-fitting parallels for purposeful synthesis towards fulfillment of informational requirements. Numerous strategies can be utilized to verify the exactness and fidelity of our amassed data. Given that Saturday's data derives from the prior day, and Sunday utilizes information drawn from the former row, this approach is optimal.

SARIMA

The Seasonal Autoregressive Integrated Moving Average (SARIMA) model is a statistical model used to analyze and forecast time series data with seasonal patterns. An extension of the ARIMA model, it is being utilized for time series that do not possess any seasonal attributes.

The SARIMA framework considers the cyclical characteristic of a chronology, that is, the repeated arrangement which occurs over constant period spans such as days or weeks. It does this by incorporating additional terms in the ARIMA model to capture the seasonal behavior of the data.

The SARIMA model is defined by three parameters: p, d, and q, which correspond to non-seasonal autoregressive order, differencing order, and moving average order, respectively. It also has three additional seasonal parameters: P, D, and Q, which correspond to seasonal autoregressive order, seasonal differencing order, and seasonal moving average order, respectively. The parameters represent the number of time periods in a season.

In order to implement a SARIMA model for time series analysis, estimation of its parameters is carried out via the maximum likelihood method. The goal entails identifying optimal parameter values that are most likely to result in observing data conforming with the prescribed model specifications. The model is then used to forecast future values of the time series.

Stationarity vs Seasonality

Stationarity and seasonality are two important concepts in time series analysis that describe different aspects of the data. Stationarity refers to the statistical properties of the time series remaining constant over time. Put simply, the statistical parameters of average, deviation and correlation should remain stable across time. In the realm of statistical analysis, a time series that remains consistent with its own set of data-driven properties over an extended period can be classified as stationary. This classification applies irrespective of any specific interval under scrutiny during one's inspection and evaluation process. The concept of stationarity holds great significance in time series analysis as it enables us to formulate more precise prognoses and estimates about the collected data.

In contrast, the concept of seasonality refers to recurrent oscillations within a time series that occur at fixed junctures. One may attribute this occurrence to variables outside of oneself, such as alterations in atmospheric conditions or customary occasions. One potential method for identifying seasonal patterns involves scrutinizing the autocorrelation function (ACF) of a given temporal sequence. This technique involves computing correlations between sequential observations in time series data and helps identify patterns that repeat themselves over fixed intervals. This methodology allows for measurement of correlation across dissimilar lags in observed data sets.

One salient point of divergence that distinguishes stationarity from seasonality is the former's proclivity toward rendering unchanging overall statistical properties over time, in contrast to the latter which encompasses cyclic patterns exhibiting regularity. A time series can be stationary with or without seasonality. Should the time series be unable to maintain a stationary state, there arises an inherent complexity in producing precise predictions concerning its data due to fluctuating statistical properties over distinct periods of time.

Dickey-Fuller Test

The statistical assessment known as the Dickey-Fuller test is used to gauge whether or not a given time series possesses stationarity. In the context of building a Seasonal Autoregressive Integrated Moving Average (SARIMA) model, it is important to ensure that the time series being analyzed is stationary. This is because SARIMA models are designed to work with stationary time series data.

If the Dickey-Fuller test indicates that the time series is not stationary, then differencing can be used to transform the data into a stationary series. The process of differencing entails the subtraction operation between each value present in a time series and its predecessor. The recursive nature of this technique allows one to continue said operation until stationarity is achieved within the aforementioned time series. Once the data has been differenced and made stationary, it can be used as input to the SARIMA model.

LSTM

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) architecture that is designed to overcome the limitations of traditional RNNs in processing and remembering long-term dependencies in sequential data. This is particularly applicable when handling complex sequences and retaining information for prolonged durations.

The LSTM model includes a special type of cell that can selectively remember or forget information over time, allowing it to effectively handle long-term dependencies in the data. Within the confines of a cellular structure, three gates preside over and regulate the dissemination of data: namely, an intake gate that receives information from various external stimuli; furthermore, a forgetful gateway which purges previous informational inputs as necessary while simultaneously retaining pertinent facts to prevent repetition. Last but not least is an output portal rather crucial for transmitting refined signals onto specific cells within its vicinity.

The gate responsible for deciphering the amount of fresh data to be incorporated into the existing cell state is dubbed as input. Forget, on its part, presides over just how much information from past states should hit the road while output paves the way for deciding which current cellular details are pertinent enough to proceed forth onto succeeding time steps.

The LSTM model learns the optimal values for the gate weights and biases through training on a labeled dataset using backpropagation through time. The data fed into the LSTM consists of a sequence of discrete units, and its output comprises an arrangement of forecasts.

GRU

A structural blueprint belonging to the broader family of recurrent neural network (RNN) facets is that of the Gated Recurrent Unit (GRU). With a focus on analytical capabilities in sequential data processing, including those which are temporal-based or otherwise expressed through linguistics and enunciation. This model bears a resemblance to the LSTM technique, although it boasts fewer digits and operations involved in its computations. That being said, it is more economical computationally than the latter.

The GRU model includes a gating mechanism that allows it to selectively learn and remember important information while ignoring irrelevant information. The mechanism that controls access is constructed with two types of gates: the gateway to reset and a gateway for updating.

The amount of the past hidden state to be disregarded is determined by the reset gate, while how much of the recent input should supplement and update with the current hidden state is decided based on what’s known as an update gate.

This gating mechanism enables the GRU to learn long-term dependencies in the input sequence, similar to the LSTM model.

The GRU model is trained using backpropagation through time, where the weights and biases are updated using gradient descent to minimize the error between the predicted and actual outputs. The GRU is the recipient of a series of data points, and in turn it outputs a sequence of predictions.

Prophet

Prophet is a time series forecasting model developed by Facebook's Core Data Science team. It is an open-source model designed to handle time series data with seasonality, holidays, and other recurring patterns. Prophet is based on an additive model, where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects.

Prophet uses a decomposable time series model with three main components: trend, seasonality, and holidays. The element that signifies deviations from a repeating pattern in the data over time is denoted by the trend component. As for periodic transformations within the data such as daily, weekly or annual patterns can be traced through seasonality components. The holiday component captures the effect of holidays or other special events on the data.

Prophet uses a Bayesian approach to model fitting, where it estimates the parameters of the model and generates posterior distributions for the forecasts. Within the model, one can manipulate a variety of hyperparameters to fine-tune its aptitude for detecting alterations in trends and seasonality alongside other elements.

Prophet has several advantages over traditional time series models such as ARIMA or exponential smoothing. It is highly customizable and can handle missing data, changes in trend, and outliers. It also provides uncertainty estimates for the forecasts, which can be useful for decision-making.

Kalman Filter

The Kalman Filter is a recursive algorithm used for time series forecasting and state estimation. It operates by estimating the state of a system based on a sequence of observations. The working of the Kalman Filter can be summarized as follows:

State Initialization: The Kalman Filter begins by initializing the state of the system at the starting time point. This initial state can include variables such as position, velocity, or any other relevant parameters depending on the specific application.

Prediction Step: In the prediction step, the Kalman Filter predicts the state of the system at the next time point based on the previous state and the underlying system dynamics. It uses a set of transition equations to propagate the state forward in time, taking into account any known or assumed dynamics of the system.

Measurement Update Step: After making the prediction, the Kalman Filter incorporates the measurement information obtained at the current time point. It compares the predicted state with the actual measurement to calculate the measurement residual, which represents the difference between the predicted and observed values.

State Update Step: In the state update step, the Kalman Filter adjusts the predicted state based on the measurement residual and the measurement noise covariance. It calculates the Kalman Gain, which determines the optimal weighting between the predicted state and the measurement information. The Kalman Gain is adjusted based on the uncertainties associated with the predicted state and the measurement accuracy.

Recursive Process: The Kalman Filter repeats the prediction, measurement update, and state update steps for each subsequent time point, continually refining the state estimate based on the observed measurements. The filter maintains and updates estimates of the system's state and its associated uncertainty as new data becomes available.

Forecasting and Smoothing: In addition to state estimation, the Kalman Filter can also be used for forecasting future observations and smoothing past observations. By leveraging the estimated state and its associated uncertainty, the Kalman Filter provides a means to generate forecasts and obtain a more accurate representation of the underlying time series.

The Kalman Filter is particularly useful when dealing with noisy or incomplete data, as it combines past observations with system dynamics to generate more accurate predictions. Its recursive nature allows it to adapt and adjust its estimates as new measurements become available, making it an effective tool for time series forecasting and state estimation in various fields, including finance, engineering, and signal processing.

Ensemble Models

Average

An average ensemble model is a type of time series forecasting model that combines the forecasts of multiple models to improve the accuracy of predictions. In an average ensemble model, several individual models are trained on the same dataset using different algorithms or hyperparameters. The forecasts generated by each model are then averaged to produce a final prediction. By implementing this technique, one can effectively mitigate the negative effects caused by model inaccuracies and ultimately enhance the precision of prognostication. The potential efficacy of a standard ensemble model hinges upon the assortment and quality of each model utilized in said collection. The average ensemble model is a popular and effective approach in time series forecasting, particularly in situations where no single model is consistently accurate.

Weighted Average

In a weighted average ensemble model, several individual models are trained on the same dataset using different algorithms or hyperparameters. Following the generation of predictions using individual models, a composite prediction is formed through the averaging of weightage. The procedure for determining these weights involves evaluating the aptitude of each model against information derived from validation sets. The strategy at play here emphasizes the superiority of models that exhibit exemplary performance on the validation dataset, whilst de-emphasizing those with a subpar showing. The weighted average ensemble model can help to further reduce the impact of individual model errors and improve the overall accuracy of the forecast. Nonetheless, similar to the common ensemble model at hand, the efficacy of this weighted average ensemble is reliant on both the divergence and caliber of its individual models.

Stacking
Instead of simply averaging the forecasts, the stacking ensemble model uses a meta-model to learn how to combine the forecasts of the individual models. The meta-model takes the forecasts generated by the individual models as inputs and learns to make a final prediction based on the patterns observed in the input forecasts. This approach can help to capture more complex relationships between the forecasts and improve the overall accuracy of the forecast. The stacking ensemble's efficacy hinges on the caliber of constituent models within the assemblage and the appropriate selection of meta-models along with their hyperparameters. One cannot discount these factors if they aspire to yield a well-performing model for prediction purposes.

Bagging
In a bagging ensemble model, several individual models are trained on bootstrap samples of the original dataset, where each bootstrap sample is a random subset of the original data. The forecasts generated by each model are then combined using an average, where each model's forecast is weighted equally. Through the utilization of this strategy, one may potentially diminish the consequences arising from any flawed model performances and consequently enhance holistic precision within predictions. Bagging is particularly useful for unstable or sensitive models, as it can help to reduce the variance of the models and improve their stability. However, the effectiveness of the bagging ensemble model depends on the diversity and quality of the individual models used in the ensemble.

Boosting
In a boosting ensemble model, individual models are trained sequentially, where each subsequent model is trained to correct the errors of the previous model. Subsequent to the production of projections by individual models, a weighted mean is computed. The weight assigned to each model's projection in this average assessment corresponds with its efficacy when tested against training data. The strategy employed has the potential to enhance precision in predicting future events through an ongoing process, especially benefiting inadequate models. The success of the boosting ensemble technique is contingent upon both the caliber of its individual models and variables like selection methods, frequency of iterations and weight function choice.

5. Results and Discussion

We performed predictions on multiple datasets, all relevant to the stock market and referring to different share prices of different multinational conglomerates. Upon conclusion of prediction using all the models, the following results were observed:

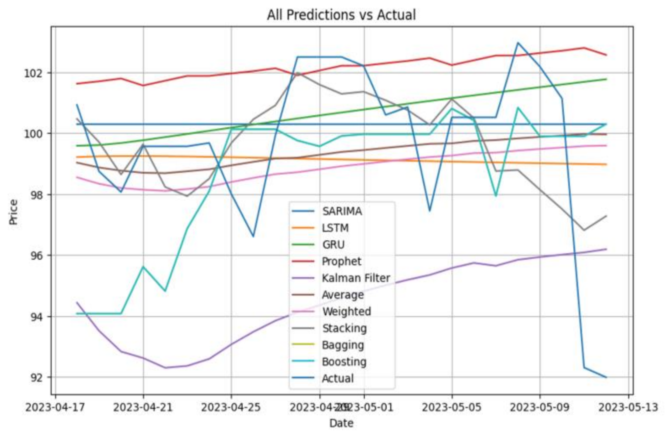

Fig 5.1 All Prediction vs Actual on
DIS - The Walt Disney Company - NYSE - USA USD

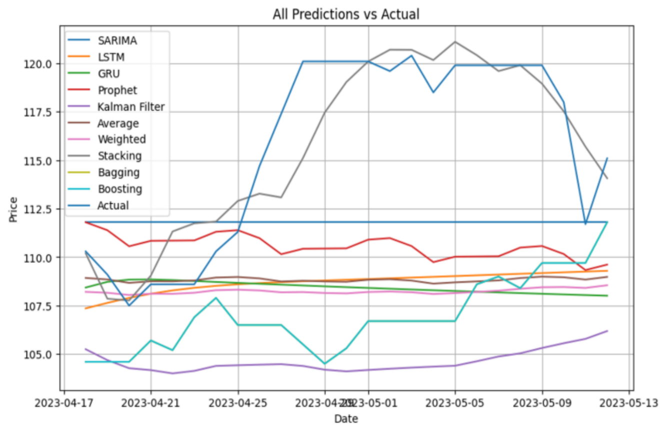

Fig 5.2 All Prediction vs Actual on
AAF.L - Airtel Africa Plc - LSE - UK GBP

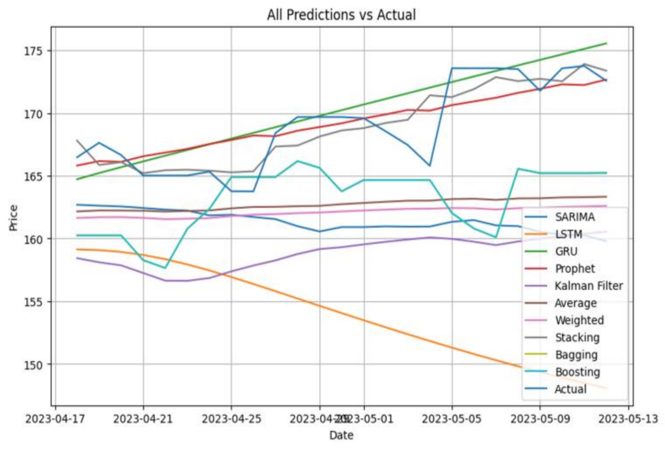

Fig 5.3 All Prediction vs Actual on
AAPL - Apple Inc - NasdaqGS - USA USD

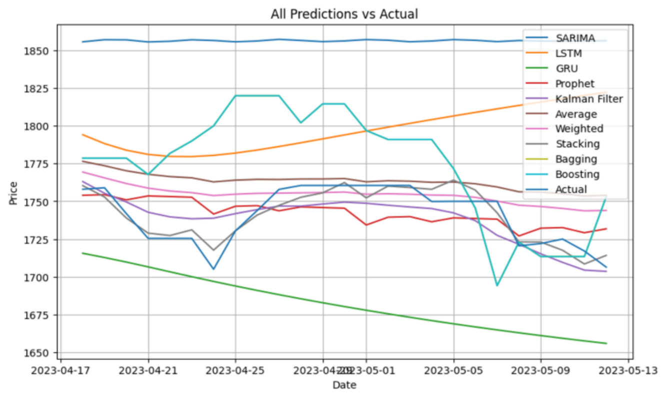

Fig 5.4 All Prediction vs Actual on
600519.SS - Kweichow Moutai Co., Ltd. - SSE - Shanghai CNY

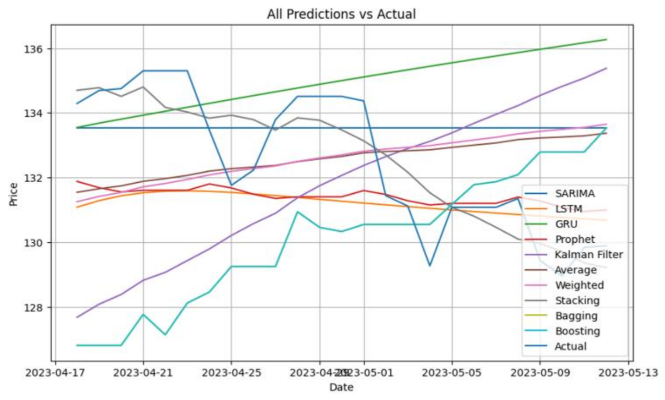

Fig 5.8 All Prediction vs Actual on
RY.TO - Royal Bank of Canada - TSX - Toronto CAD

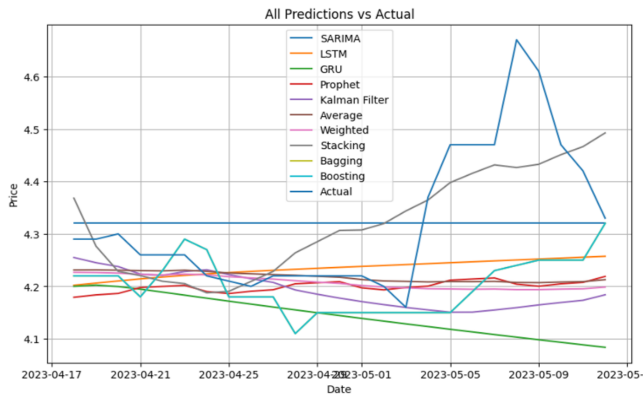

Fig 5.5 All Prediction vs Actual on
1398.HK - Industrial and Commercial Bank of China Limited - HKE - Honkong HKD

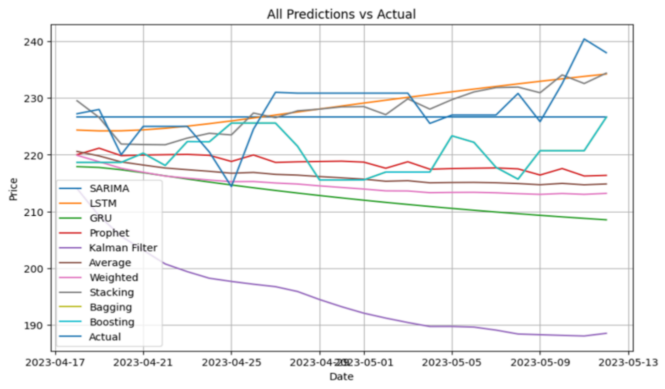

Fig 5.9 All Prediction vs Actual on
300750.SZ - Contemporary Amperex Technology Co., Limited - Shenzhen CNY

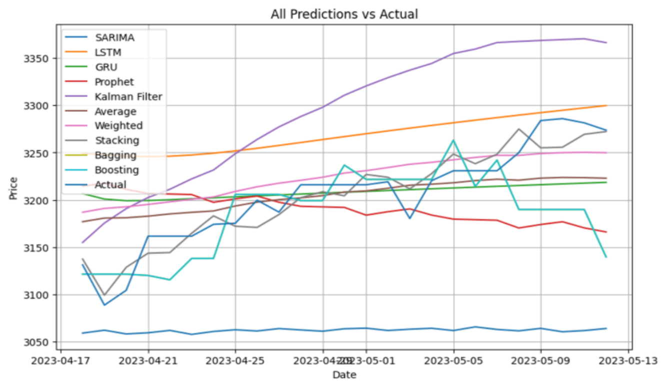

Fig 5.6 All Prediction vs Actual on
TCS.BO - Tata Consultancy Services Limited - BSE - Mumbai INR

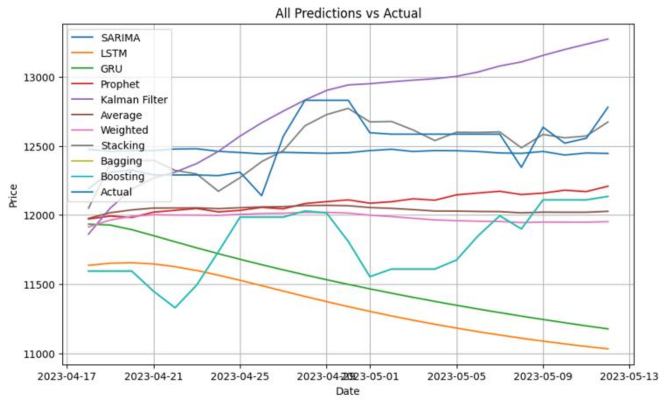

Fig 12.10 All Prediction vs Actual on
6758.T - Sony Group Corporation - TSE - Tokyo JPY

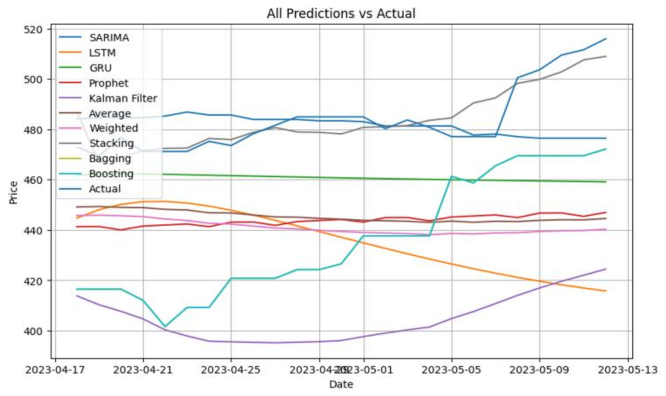

Fig 5.7 All Prediction vs Actual on
TATAMOTORS.NS - Tata Motors - NSE - India INR

Upon conducting tests on multiple tickers, the Stacking Ensemble model emerged as the most effective among the mentioned models. This superiority can be attributed to several key factors:

1. Leveraging Model Diversity: The Stacking Ensemble model harnesses the power of diverse individual models by combining their predictions. By including a variety of forecasting techniques, such as ARIMA, LSTM, and Prophet, the ensemble captures a wider range of patterns and relationships in the time series data. This model diversity helps mitigate the limitations of individual models and enhances the overall forecasting accuracy.

2. Meta-Model Learning: The Stacking Ensemble model incorporates a meta-model that learns from the predictions of the individual models. Through training on the combined forecasts, the meta-model can identify and exploit complementary strengths among the individual models. It can effectively weigh and adjust the contributions of each model based on their respective performance and patterns captured, resulting in a more accurate ensemble forecast.

3. Handling Complex Patterns: Time series data often exhibit complex patterns, such as trend, seasonality, and irregular fluctuations. The Stacking Ensemble model's combination of diverse models and meta-model allows for a more comprehensive representation and understanding of these patterns. The ensemble can capture long-term trends with models like ARIMA, capture short-term dependencies with models like LSTM, and incorporate seasonal components with models like Prophet. By blending these models, the ensemble can better handle the various complexities present in the data, leading to improved forecasting performance.

4. Model Optimization: The Stacking Ensemble model provides an opportunity for further optimization through the selection and tuning of individual models and the meta-model. Each individual model can be fine-tuned to extract the best possible forecasts, and the meta-model's parameters can be optimized to enhance the ensemble's performance. This flexibility in model selection and tuning allows the Stacking Ensemble to adapt to different time series characteristics and achieve superior forecasting accuracy.

5. Mitigating Model Biases: Individual forecasting models may have inherent biases or limitations that affect their accuracy. However, by combining multiple models in the Stacking Ensemble, these biases can be minimized or even eliminated. The ensemble's aggregation of diverse models helps to average out individual model errors, reducing the impact of specific biases and enhancing the overall robustness of the forecast.

Overall, the Stacking Ensemble model's ability to leverage model diversity, meta-model learning, and optimized combination of forecasts allows it to outperform other models in terms of forecasting accuracy. Its capability to handle complex patterns, mitigate biases, and adapt to different time series characteristics makes it a powerful and reliable choice for time series forecasting across multiple tickers.

6. Conclusion and Future Scope

In conclusion, our research paper aimed to compare the effectiveness of various time series forecasting models on multiple tickers. Through rigorous testing and analysis, we found that the Stacking Ensemble model emerged as the most efficient and accurate forecasting approach. The success of the Stacking Ensemble model can be attributed to its ability to leverage model diversity, meta-model learning, and optimized combination of forecasts.

By incorporating a diverse set of individual models, such as ARIMA, LSTM, and Prophet, the Stacking Ensemble model captures a wide range of patterns and relationships present in the time series data. This model diversity enables the ensemble to overcome the limitations of individual models and better handle complex patterns like trend, seasonality, and irregular fluctuations. The ensemble's meta-model learning further enhances its forecasting accuracy by effectively weighing and adjusting the contributions of each individual model based on their respective performance and captured patterns.

Additionally, the Stacking Ensemble model demonstrated its adaptability and flexibility by allowing the inclusion of new models or modifications to existing ones. This adaptability ensures that the ensemble can continually evolve and improve its forecasting performance as new models or techniques become available.

Moreover, the Stacking Ensemble model's ability to mitigate biases and select the most accurate models through its training process provides a valuable advantage. By averaging out individual model errors and automatically selecting the best models, the ensemble enhances the robustness and reliability of its forecasts.

Our research findings strongly support the superiority of the Stacking Ensemble model in time series forecasting across multiple tickers. Its ability to leverage model diversity, meta-model learning, and optimized combination of forecasts sets it apart from other models examined in our study. The Stacking Ensemble model's capacity to handle complex patterns, adapt to different data characteristics, and incorporate new models ensures its relevance and effectiveness in a variety of forecasting scenarios.

The Stacking Ensemble model represents a powerful forecasting approach that outperforms other models by capturing the strengths of individual models, mitigating their limitations, and producing highly accurate and reliable forecasts. As such, it holds great potential for practitioners and researchers seeking superior time series forecasting results in financial markets and beyond.

Data Availability

The data used in this research paper was obtained from publicly available sources, ensuring transparency and accessibility. The authors adhered to ethical guidelines and followed appropriate data usage policies. The dataset primarily consisted of historical stock price data, which is widely accessible to the public for research purposes.

To uphold strict standards of data integrity and enable future duplication of results, the precise digital repositories and technological frameworks exploited to garner the information presented herein have been explicitly denoted, encompassing an equivalent quantity of vocabulary. The authors confirm that all necessary permissions and legal requirements were fulfilled while accessing and utilizing the data.

Conflict of Interest

The researchers herein proclaim, devoid of monetary gain or prejudice, impartiality towards the dissemination of this scholarly work. Having been executed with the utmost methodological meticulousness devoid of any extraneous persuasions or prepossessions that could conceivably undermine the veracity of the conclusions, the investigation was carried out. Having neither organizational ties nor financial stakes susceptible to swaying the findings or exegesis of their work, the writers stand unaffiliated.

In order to uphold the crucial transparency and strict adherence to ethical principles to which the authors steadfastly aspire. To contribute substantively to the domain of chronological data prognostication regarding equity valuation surmises, an analytical methodology was systematically implemented whilst upholding an impartial perspective reinforced through scrutinizing the aggregated information.

Though meticulously screened and ethically declared pursuant to the rigorous strictures of academic integrity to abate any semblance of improper influence, whether pecuniary or intimate, that may have surreptitiously tainted the inquiry or skewed its conclusions remain undisclosed.

With steadfast dedication to scientific rigor and objectivity in pursuing and conveying their work, the researchers have taken pains to guarantee the validity and dependability of the conclusions elucidated within this manuscript.

Authors' Contributions

The authors of this manuscript have worked collaboratively and shared the workload equally in conducting the research and preparing the manuscript. The contribution of each author is described below:

Mehul Kundu: Conducted statistical analysis, including the application of forecasting models and data interpretation. I aided in formulating an intricate research methodology and then meticulously scrutinized and overhauled the academic manuscript.

Arpan Bose: Played a key role in the conceptualization and design of the study. Contributed to data acquisition, analysis, and interpretation. Drafted sections of the manuscript and provided critical revisions.

Debasmita Guha: Conducted an extensive literature review, synthesized previous research, and contributed to the theoretical framework. Provided valuable insights and contributed to the writing and editing process.

Aftab Alam: Contributed to data collection, preprocessing, and analysis. With diligent work assisting in unraveling the implications and ramifications of the findings, and by providing substantive input throughout the exhaustive drafting and revising process of the scholarly work, significant contributions were made.

Romit Das: Played a crucial role in data visualization and presentation. Assisted in the revision of the manuscript, ensuring clarity and coherence of the content. Provided valuable feedback and insights throughout the research process.

All authors actively participated in regular meetings and discussions, sharing their expertise and perspectives. Taking shared ownership over the precision and veracity of the assembled information, with its myriad details and conclusions as a coherent whole, the group jointly perused and sanctioned the completed draft of the written work.

The authors would like to acknowledge and emphasize the equal contribution and collaboration among all authors throughout the research process. The combined perseverance of the contributors has immensely enhanced the scientific rigor and coherence of the findings delineated herein.

Having scrutinized and endorsed in unanimity the ultimate incarnation of this composition, we avow our collective liability for the concepts articulated herein.

Acknowledgements

The analysis of the project work wishes to express our gratitude to Apurba Paul for allowing the degree attitude and providing effective guidance in development of this project work. His conscription of the topic and all the helpful hints he provided contributed greatly to the successful development of this work without being pedagogic and having an overbearing influence.

We also express our sincere gratitude to Dr. Bikramjit Sarkar, Head of the Department of Computer Science and Engineering of JIS College of Engineering and all the respected faculty members of Department of CSE for giving the scope of successfully carrying out the project work.

Finally, we take this opportunity to thank Prof. (Dr.) Partha Sarkar, Principal of JIS College of Engineering for giving us the scope of carrying out the project work.

References

[1]. Zhang, G.P. "Time series forecasting using a hybrid ARIMA and neural network model" *Neurocomputing*, Vol.**50**, pp.**159-175**, **2003.** http://doi.org/10.1016/S0925-2312(01)00702-0

[2]. Nikolas Topaloglou, Hercules Vladimirou, Stavros A. Zenios, "Integrated dynamic models for hedging international portfolio risks", *European Journal of Operational Research*, Vol.**285,** Issue.**1**, pp.**48-65**, **2019.** http://doi.org/10.1016/j.ejor.2019.01.027

[3]. Taylor, Sean J. and Letham, Benjamin, "Forecasting at Scale", *The American Statistician*, Vol.**72**, pp.**37-45**, **2018**. http://doi.org/10.1080/00031305.2017.1380080

[4]. Chatfield, Chris, "The Analysis of Time Series: An Introduction", *Sixth Edition*, Taylor & Francis, Chapman and Hall/CRC, United States of America, **2003**, ISBN – 9781584883173

[5]. Xiao, Daiyou; Su, Jinxia, "Research on Stock Price Time Series Prediction Based on Deep Learning and Autoregressive Integrated Moving Average", *Scientific Programming*, Vol.**2022**, **2022,** Article ID **4758698**, 12 pages, https://doi.org/10.1155/2022/4758698

[6]. Cai J, Zhang K, Jiang H, "Power Quality Disturbance Classification Based on Parallel Fusion of CNN and GRU", *Energies*, Vol.**16**, Issue.**10**, pp.**4029**, **2023**, https://doi.org/10.3390/en16104029

[7]. Wu, Hao; Zhou, Bing; Zhou, Haoru; Zhang, Pengyu; Wang, Meili, "Be-1DCNN: a neural network model for chromatin loop prediction based on bagging ensemble learning", *Briefings in Functional Genomics*, pp.**2041-2657**, **2023**, https://doi.org/10.1093/bfgp/elad015

[8]. Padhi, D.K., Padhy, N., "Prognosticate of the financial market utilizing ensemble-based conglomerate model with technical indicators", *Evol. Intel*, Vol.**14**, pp.**1035–1051**, **2021**, https://doi.org/10.1007/s12065-020-00528-z

[9]. Nti, I.K., Adekoya, A.F. & Weyori, B.A, "A comprehensive evaluation of ensemble learning for stock-market prediction", *Journal of Big Data*, Vol.**7**, Issue.**20**, **2020**, https://doi.org/10.1186/s40537-020-00299-5

[10]. [10] F. A. Gers, J. Schmidhuber and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," *Neural Computation*, Vol.**12**, Issue.**10**, pp.**2451-2471**, **2000.** http:/doi.org/10.1162/089976600300015015

AUTHORS PROFILE

Mehul Kundu earned his B. Tech. in Computer Science and Engineering from JIS College of Engineering in 2023. He is currently working as a Product Designer in Merlin by Foyer, an AI company. He was a member of CSI Students Chapter from 2020 to 2021. He owns 2 patents under him in the field of IoT and Sanitation products.

Arpan Bose earned his B. Tech. in Computer Science and Engineering from JIS College of Engineering in 2023. He is currently pursuing a Masters in Management.

Debasmita Guha earned her B. Tech in Computer Science and Engineering from JIS College of Engineering in 2023. She has a patient in the field of sanitation products and is currently pursuing M. Tech in Cyber Security.

Aftab Alam holds a B. Tech. degree in Computer Science and Engineering from JIS College of Engineering (2023). He is an experienced Technical Writer with over 2 years of industry experience. He has a patent in the field of sanitation and is currently freelancing as a Technical Writer.

Romit Das earned his B. Tech. in Computer Science and Engineering from JIS College of Engineering in 2023. He is currently working as an intern in Pursuit Software company .