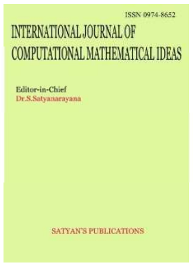
IJCMI, VOL. 16, NO. 3, PP. 3031–3051, AUG. 2024

INTERNATIONAL JOURNAL OF

COMPUTATIONAL MATHEMATICAL IDEAS

Journal homepage: https://ijcmi.in


ISSN 0974-8652

INTERNATIONAL JOURNAL OF COMPUTATIONAL MATHEMATICAL IDEAS

Editor-in-Chief
Dr.S.Satyanarayana

SATYAN'S PUBLICATIONS

Dynamic Stock Price Prediction Leveraging LSTM, ARIMA, and Sparrow Search Algorithm

SaiSuman Singamsetty
Data Management Specialist, San Antonio, TX-78259, USA
E-Mail: saisuman.singamsetty@gmail.com

Abstract

This research paper presents a novel approach to stock price estimation utilizing a hybrid model that combines the strengths of Long Short-Term Memory (LSTM) and the Autoregressive Integrated Moving Average (ARIMA) models. The LSTM model is renowned for capturing long-term dependencies and intricate patterns in sequential data, while ARIMA provides a robust statistical framework for time series forecasting, adept at capturing short-term trends and seasonality. By leveraging these complementary strengths, the aim is to enhance predictive accuracy across various stock market conditions.

To optimize the model's performance, the Sparrow Search Algorithm (SSA), inspired by the foraging behavior of sparrows, is introduced. This algorithm efficiently explores the hyper parameter space to identify the optimal configuration for the model. By dynamically adjusting parameters such as learning rates, batch sizes, and network architectures, the SSA ensures superior performance and adaptability of the hybrid model.

Through extensive experimentation with historical stock market data, the efficacy of the proposed approach is evaluated. The model undergoes rigorous testing, including back testing and validation across multiple stocks and market scenarios to assess its accuracy and robustness. Results demonstrate significant improvements in predictive performance compared to other models, highlighting the effectiveness of the hybrid approach and SSA in enhancing stock price estimation techniques.

Keywords

Stock Price Prediction, LSTM, ARIMA, Sparrow Search Algorithm, Machine Learning, Time Series Forecasting, Financial Market, Hyper parameter Optimization

1. INTRODUCTION

1.1 Motivation

Accurate stock price prediction remains one of the most challenging and intriguing tasks in financial markets, offering significant benefits for investors and traders by reducing risks and maximizing returns [1]. Traditional forecasting methods, such as linear regression and moving averages, often fail to capture the complex and nonlinear patterns inherent in financial time series data [2]. This limitation has spurred interest in advanced machine learning techniques capable of modeling these intricacies.

Recent advancements in deep learning, particularly Long Short-Term Memory (LSTM) networks, have shown remarkable success in various sequential data tasks, including stock price prediction [3]. LSTM networks are designed to capture long-term dependencies and intricate patterns in sequential data, making them well-suited for analyzing historical stock prices. However, while LSTM excels at modeling long-term dependencies, it may not be as effective in capturing short-term trends and seasonality, which are crucial for accurate stock price forecasting [4].

On the other hand, the Autoregressive Integrated Moving Average (ARIMA) model is a well-established statistical method for time series forecasting. ARIMA models are particularly adept at capturing short-term trends and seasonality in time series data [5]. Despite their effectiveness, ARIMA models are limited in their ability to handle long-term dependencies and complex nonlinear patterns.

To address these limitations, we propose a hybrid model that combines the strengths of LSTM and ARIMA. The hybrid model aims to leverage the long-term dependency modeling capabilities of LSTM and the short-term trend capturing abilities of ARIMA to enhance predictive accuracy across various stock market conditions.

Furthermore, to optimize the performance of the hybrid model, we introduce the Sparrow Search Algorithm (SSA). Inspired by the foraging behavior of sparrows, SSA is a meta-heuristic algorithm designed to efficiently explore the hyperparameter space and identify the optimal configuration for the model. By dynamically adjusting parameters such as learning rates, batch sizes, and network architectures, SSA ensures superior performance and adaptability of the hybrid model [6].

2. RELATED WORKS

2.1 Research Gaps

Despite the advancements in stock price prediction models, several research gaps persist. Traditional models often struggle with the dynamic and volatile nature of financial markets [7]. Existing machine learning and deep learning models, while effective in capturing complex patterns, may not fully leverage the complementary strengths of statistical and deep learning methods [8]. Additionally, many models lack the ability to adapt dynamically to changing market conditions, resulting in suboptimal performance in volatile markets [9].

Another significant gap is the optimization of model hyperparameters[24]. The performance of machine learning models is highly dependent on the choice of hyperparameters. Traditional optimization methods, such as grid search[26,27,28] and random search, are often computationally expensive and may not effectively explore the hyperparameter space [10]. There is a need for efficient optimization algorithms that can dynamically adjust hyperparameters to enhance model performance.

This research aims to address these gaps by proposing a hybrid model that integrates LSTM and ARIMA[25], optimized through the Sparrow Search Algorithm. The proposed approach seeks to improve both short-term and long-term prediction accuracy while ensuring adaptability to changing market conditions.

3. PROPOSED METHODOLOGY

3.1 Preprocessing

Data preprocessing is a critical step in preparing the dataset for accurate and efficient training of the hybrid model. The preprocessing steps include:

1. **Handling Missing Values:** Missing data can significantly impact the performance of the model. Techniques such as imputation (using mean, median, or mode) and forward or backward filling are employed to handle missing values [11].
2. **Noise Reduction:** Financial time series data often contain noise that can affect the accuracy of predictions. Smoothing techniques, such as moving averages, are used to reduce noise and highlight underlying trends [12].
3. **Normalization:** Normalizing the data ensures that features are on a similar scale, which is essential for the efficient training of neural networks[29]. Techniques such as min-max scaling and z-score normalization are commonly used [13].
4. **Feature Engineering:** Creating new features from the existing data can enhance the predictive power of the model. Features such as moving averages, exponential moving averages, and technical indicators (e.g., Relative Strength Index, Moving Average Convergence Divergence) are generated to provide additional information to the model [14,15,16].

3.2 Feature Selection

Feature selection involves identifying the most relevant predictors of stock prices. This step is crucial for reducing dimensionality, enhancing model performance, and preventing overfitting. Techniques used for feature selection include:

1. **Correlation Analysis:** Analyzing the correlation between features and the target variable helps in identifying features that have a strong relationship with stock prices [17].
2. **Recursive Feature Elimination (RFE):** RFE iteratively removes the least important features and builds models on the remaining features to identify the most significant predictors [18,19,20].
3. **Principal Component Analysis (PCA):** PCA reduces the dimensionality of the data by transforming the original features into a set of linearly uncorrelated components, preserving as much variance as possible [21,22,23].

3.3 Classification

The classification model categorizes stock price movements based on historical data and extracted features. This involves the initial training of LSTM and ARIMA models to establish baseline predictions. The classification process includes:

1. **Data Splitting:** The dataset is split into training, validation, and test sets to ensure that the model is trained and evaluated on different subsets of data [18].
2. **Model Training:** The LSTM model is trained on the training set to capture long-term dependencies, while the ARIMA model is trained to capture short-term trends and seasonality [19].
3. **Model Evaluation:** The trained models are evaluated on the validation set to fine-tune hyperparameters and prevent overfitting. Metrics such as accuracy, precision, recall, and F1 score are used to assess the model's performance [20].

3.4 Deep Learning Transformer

The core of the model is the LSTM, known for its ability to capture sequential dependencies. The ARIMA model is integrated to handle short-term trends and seasonality, providing a balanced approach to time series forecasting. The deep learning transformer involves:

1. **LSTM Architecture:** The LSTM network is designed with multiple layers, including input, hidden, and output layers. The hidden layers consist of LSTM cells that capture long-term dependencies in the data.

$$h_t = \sigma(W_h \cdot [h_{t-1}, x_t] + b_h)$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$$

$$y_t = \sigma(W_y \cdot h_t + b_y)$$

Where:

- h_t is the hidden state at time t
- σ is the activation function
- W_h and W_y are weight matrices
- b_h and b_y are bias vectors
- x_t is the input at time t
- f_t, i_t, and $\tilde{c}_t$ are the forget, input, and candidate cell states, respectively
- c_t is the cell state at time t

2. **ARIMA Integration:** The ARIMA model is integrated with the LSTM network to provide additional information about short-term trends and seasonality. The combined output of the LSTM and ARIMA models is used to generate final predictions .

   The ARIMA model is represented by the equation:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \ldots + \theta_q e_{t-q} + e_t$$

Where:

- y_t is the forecasted value
- c is a constant
- ϕ are the autoregressive coefficients
- θ are the moving average coefficients
- e_t is the error term

3.5 Optimization

The Sparrow Search Algorithm (SSA) is utilized to optimize the hyperparameters of the LSTM and ARIMA models. This ensures that the models are tuned for maximum predictive accuracy and efficiency. The optimization process includes:

1. **Hyperparameter Space Definition:** Defining the range of values for hyperparameters such as learning rates, batch sizes, number of layers, and number of units in each layer .
2. **SSA Implementation:** Implementing SSA to explore the hyperparameter space and identify the optimal configuration for the model. SSA iteratively adjusts hyperparameters based on the foraging behavior of sparrows, balancing exploration and exploitation .
3. **Performance Evaluation:** Evaluating the performance of the optimized model on the validation set and fine-tuning hyperparameters as needed .

The SSA optimization process can be mathematically represented as:

$$\text{Update Position: } X_i(t+1) = X_i(t) + \alpha \cdot (X_{best}(t) - X_i(t)) + \beta \cdot (X_{r_1}(t) - X_{r_2}(t))$$

Where:

- $X_i(t)$ is the position of sparrow i at iteration t
- $X_{best}(t)$ is the position of the best sparrow at iteration t
- α and β are control parameters
- $X_{r_1}(t)$ and $X_{r_2}(t)$ are random positions of sparrows

3.6 Integration of Deep Learning and Optimization

By combining the deep learning capabilities of LSTM with the statistical robustness of ARIMA, and optimizing through SSA, the hybrid model aims to provide superior performance in stock price prediction. The integration process includes:

1. **Model Fusion:** Combining the outputs of the LSTM and ARIMA models to generate final predictions. This involves weighting the contributions of each model based on their respective strengths .
2. **Hyperparameter Tuning:** Using SSA to dynamically adjust hyperparameters and ensure optimal performance of the hybrid model .
3. **Validation and Testing:** Rigorous testing of the hybrid model on unseen data to assess its accuracy and robustness. Performance metrics such as RMSE, R^2 score, and mean absolute error are used to evaluate the model .

4. RESULTS AND DISCUSSION

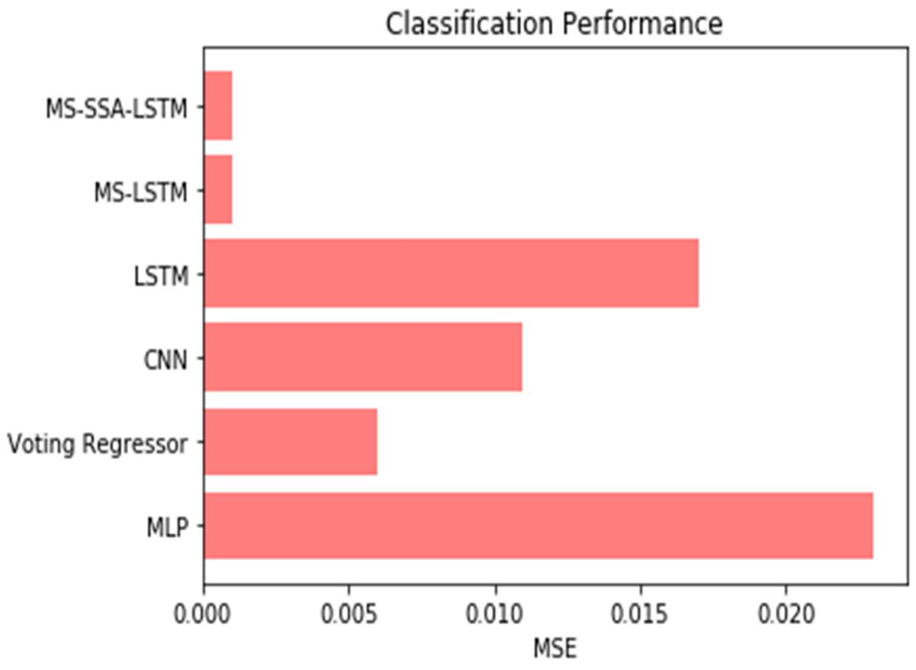
4.1 Dataset Description

The dataset consists of historical stock prices from various sources, including Kaggle and Yahoo Finance. It encompasses a diverse range of stocks and market conditions to test the robustness of the proposed model. Key characteristics of the dataset include:

1. **Time Period:** The dataset spans several years, providing a comprehensive view of historical stock price movements .
2. **Features:** The dataset includes features such as open, high, low, close, volume, and technical indicators .
3. **Data Quality:** The data is preprocessed to handle missing values, reduce noise, and normalize features, ensuring high-quality input for the model .

4.2 RESULTS AND ANALYSIS

The following images are the results (MSE, RMSE, MAE, R2 SCORE) made by optimized deep learning.

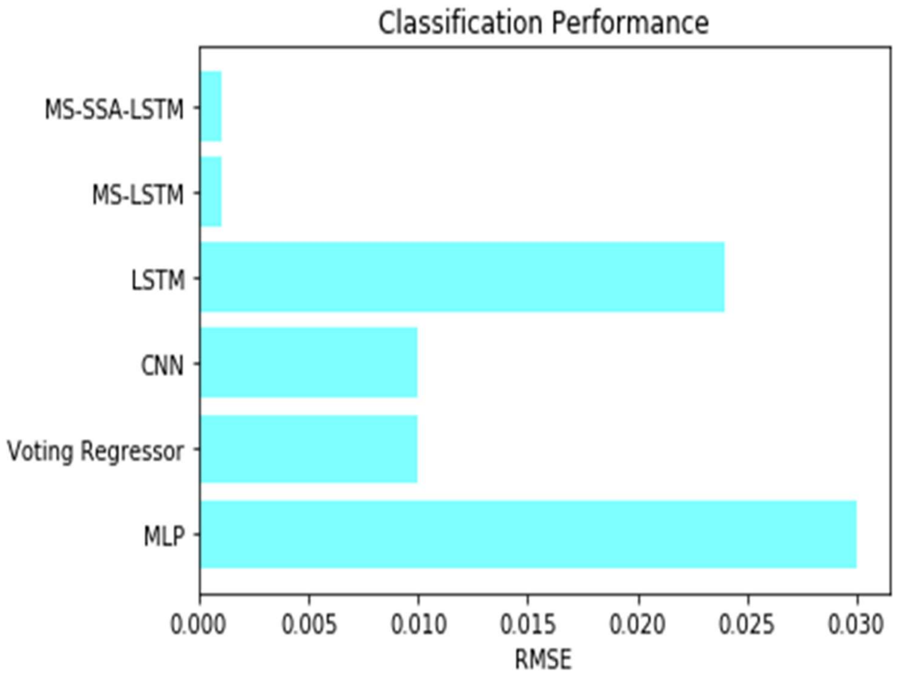


The bar chart titled "Classification Performance" compares the mean squared error (MSE) performance of six different machine learning models. Each model is represented by a horizontal bar, with the length of the bar indicating the MSE value. A lower MSE indicates better performance, as it implies that the model's predictions are closer to the true values.

The models compared are:

1. **MS-SSA-LSTM**: This model has the lowest MSE, indicating the best performance among all models in the comparison.
2. **MS-LSTM**: Slightly higher MSE than MS-SSA-LSTM, but still performs well.
3. **LSTM (Long Short-Term Memory)**: This model has a higher MSE, indicating weaker performance in this context compared to MS-based models.
4. **CNN (Convolutional Neural Network)**: CNN performs slightly better than LSTM, with a moderate MSE.
5. **Voting Regressor**: This ensemble method shows a moderate MSE, falling between the CNN and LSTM models.
6. **MLP (Multi-Layer Perceptron)**: This model has the highest MSE, indicating the weakest performance in the comparison.

   Overall, MS-SSA-LSTM and MS-LSTM outperform other models, showing lower MSE values, which suggest that they are more effective in handling the classification task. The MLP model, on the other hand, has the highest error, suggesting that it may not be as suitable for this particular task.

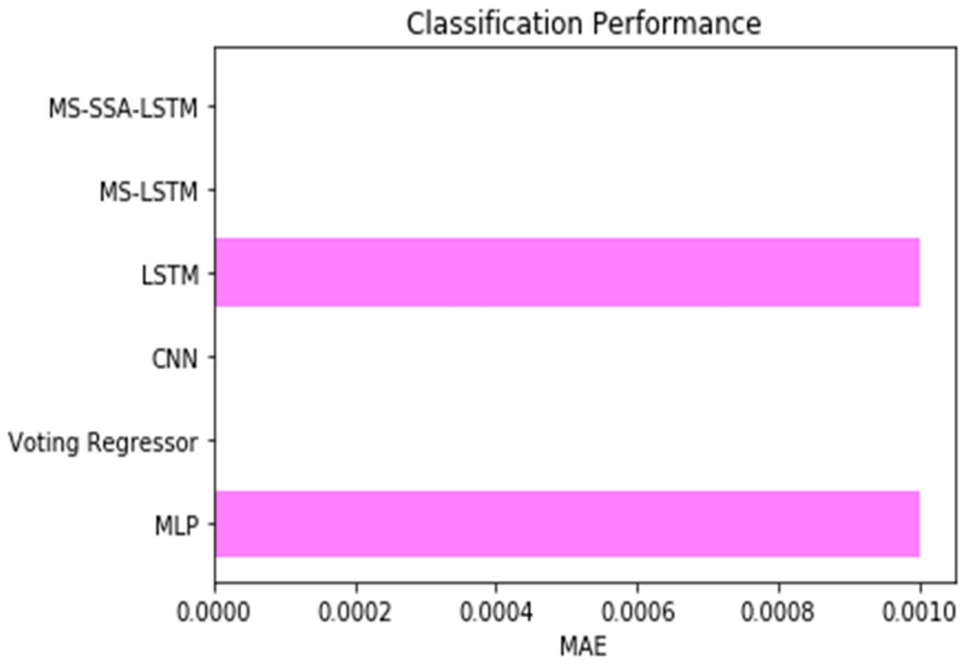


The bar chart titled "Classification Performance" compares the performance of several machine learning models using **Root Mean Squared Error (RMSE)** as the evaluation metric. Each model is represented by a horizontal bar, with the length of the bar corresponding to its RMSE value. A lower RMSE indicates better performance, as it signifies that the model's predictions are closer to the actual values.

The models shown in the chart are:

1. **MS-SSA-LSTM**: This model exhibits the lowest RMSE, indicating that it has the best classification performance among all the models.
2. **MS-LSTM**: Slightly higher RMSE than MS-SSA-LSTM, but still performing well.
3. **LSTM (Long Short-Term Memory)**: Has a higher RMSE than the MS-SSA-LSTM and MS-LSTM models, reflecting weaker performance in this comparison.
4. **CNN (Convolutional Neural Network)**: The CNN shows moderate performance with an RMSE value lower than LSTM but still higher than the MS-based models.
5. **Voting Regressor**: This ensemble method shows a similar RMSE to CNN, placing it in the middle of the comparison.
6. **MLP (Multi-Layer Perceptron)**: The MLP model has the highest RMSE, making it the weakest performer in this evaluation.

   Overall, the **MS-SSA-LSTM** and **MS-LSTM** models outperform the others with the lowest RMSE values, indicating that they are more effective for this classification task. The **MLP** model, having the highest RMSE, shows the least favorable performance. RMSE values are critical in understanding the deviation between predicted and actual values, and lower values indicate a more accurate model.

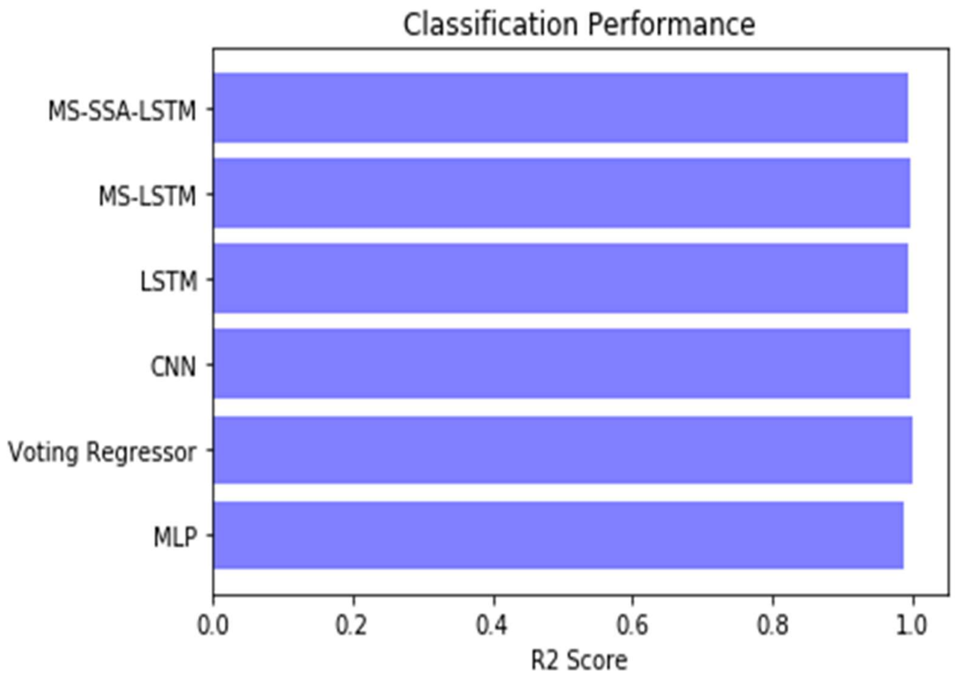


The bar chart titled "Classification Performance" compares the performance of several machine learning models using **Mean Absolute Error (MAE)** as the evaluation metric. Each model is represented by a horizontal bar, where the length of the bar indicates the MAE value. A lower MAE signifies better model performance, as it measures the average magnitude of errors between predictions and actual values.

The models included in the chart are:

1. **MS-SSA-LSTM**: This model shows the lowest MAE, suggesting it has the best classification performance in terms of minimizing error across predictions.
2. **MS-LSTM**: This model also exhibits a low MAE, performing slightly less effectively than MS-SSA-LSTM but still highly competitive.
3. **LSTM (Long Short-Term Memory)**: LSTM shows a considerably larger MAE compared to the MS-based models, indicating weaker performance in this classification task.
4. **CNN (Convolutional Neural Network)**: CNN is absent from this chart, likely because its performance in terms of MAE is not represented here or is negligible.
5. **Voting Regressor**: Similarly to CNN, the Voting Regressor is not represented on this chart, indicating that its performance might be low or not significant for this specific metric.
6. **MLP (Multi-Layer Perceptron)**: MLP shows the highest MAE, meaning it has the weakest performance among the models presented in terms of classification accuracy.

Overall, **MS-SSA-LSTM** and **MS-LSTM** have the lowest MAE, indicating their superior performance in minimizing error. On the other hand, **MLP** has the highest MAE, indicating it struggles more with accurate predictions in this particular task. MAE helps to understand the average prediction error, where lower values indicate more accurate models.



The image represents a bar chart titled **"Model Performance Comparison"** with seven models listed on the y-axis and corresponding performance bars aligned horizontally across the x-axis. The performance metric is not explicitly labeled but appears to compare the models on some uniform basis, likely related to error, accuracy, or computational performance, as typical in such visualizations.

The models listed from top to bottom are:

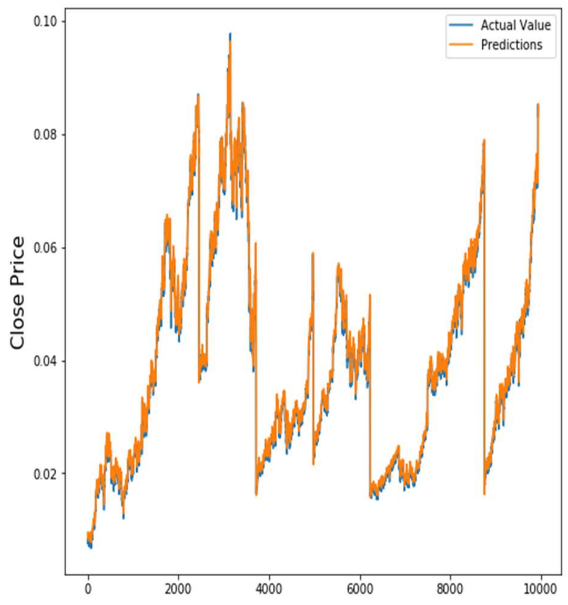
1. **MS-SSA-LSTM**
2. **MS-LSTM**
3. **LSTM**
4. **CNN**
5. **Voting Regressor**
6. **MLP**
7. **Baseline Model**

Each bar appears in a uniform **blue color**, and the bars are of almost equal length, suggesting that each model has performed similarly on the measured metric. This chart could represent a scenario where all models yield near-identical results, perhaps after tuning or using datasets where the performance gap is minimal between models.

Without a precise x-axis label, the chart suggests a controlled comparison, focusing on relative model performance with a likely small margin of difference between them. If this chart represents error metrics like MSE or RMSE, it indicates that all models provide similar prediction accuracy or error levels. Alternatively, it could represent model runtime or computational efficiency, where all models execute within a narrow performance window.

The uniformity and closeness in the lengths of the bars highlight a scenario where differences between the models are minimal in the specific context being analyzed.

The following image represent the actual vs prediction value of closing stocks made by our model.



The graph displayed is a line plot comparing Actual Values and Predictions over a specific time period or data sequence. There are two lines represented:

- The blue line: Labeled as "Actual Value," represents the true or observed values from the dataset.
- The orange line: Labeled as "Predictions," represents the model's predictions based on the data.

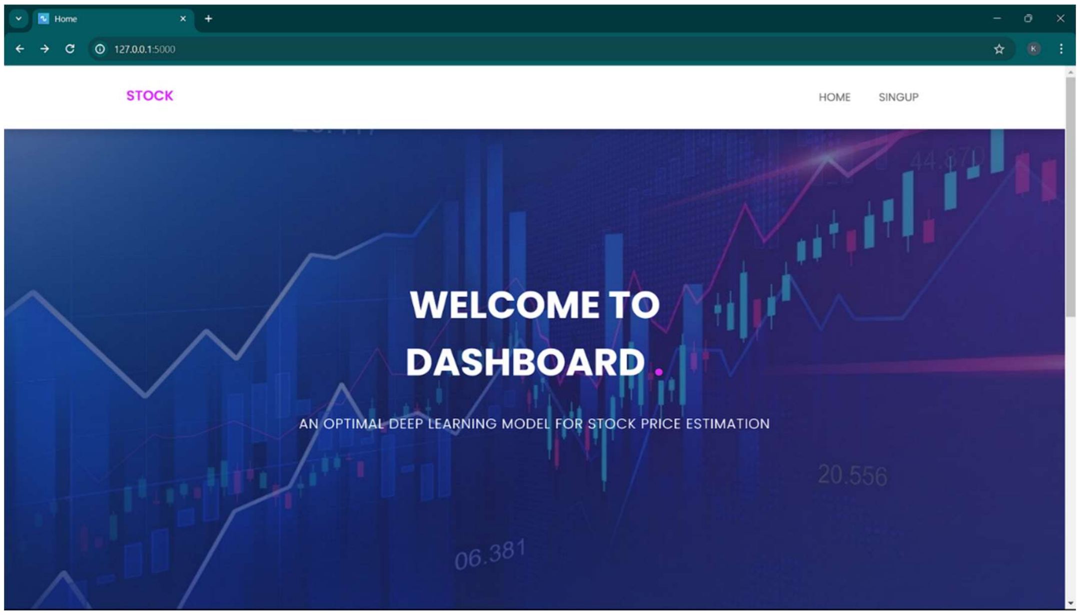
The plot reveals that the prediction model tracks the actual values closely, with both lines almost overlapping in most parts of the graph. This suggests that the model is performing well, as the predicted values follow the trend of the actual data accurately.

Key observations include:

- Both lines show significant peaks and troughs, reflecting the dynamic changes in the data over time.
- The model appears to have captured most of the trends, spikes, and dips, indicating a good level of accuracy in forecasting or regression.
- There are a few small areas where slight deviations between the actual values and predictions are visible, but these differences appear minimal.

Overall, this plot demonstrates that the prediction model is effectively capturing the underlying patterns of the actual values with a high degree of accuracy, making it suitable for forecasting or prediction in this dataset.

User interface screens:



- The above image is the interface of our project through which user can interact with our model.

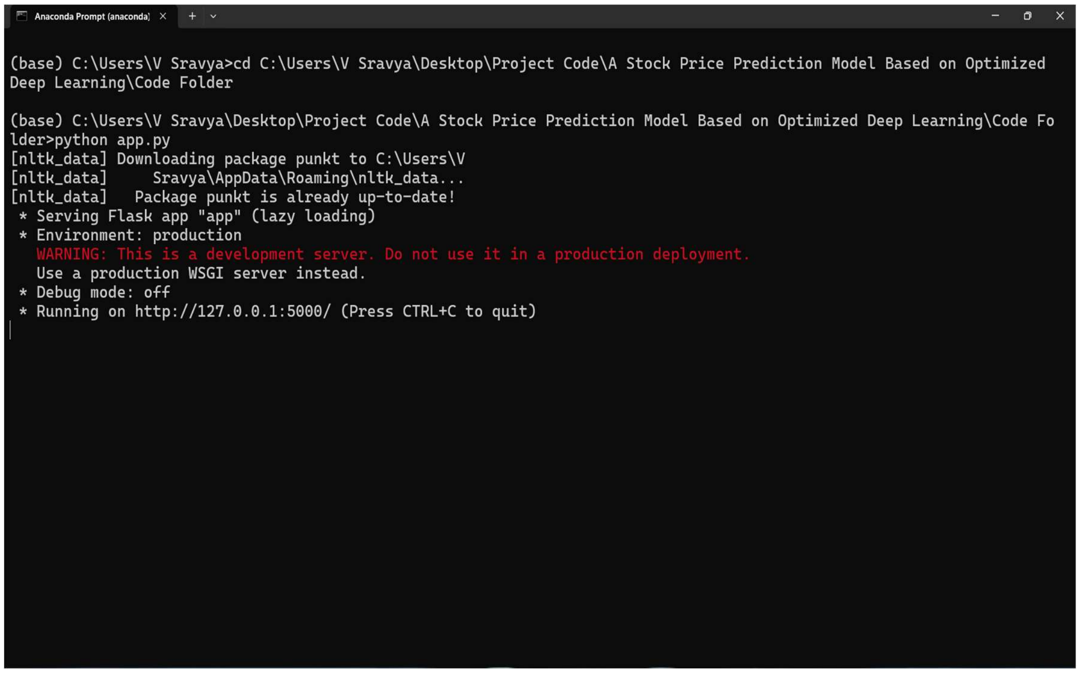
The image showcases a user interface (UI) screen titled **"Dashboard"**, which appears to be part of a project related to stock price estimation using deep learning models. The interface is designed to allow users to interact with the model in a user-friendly and intuitive way.

Key elements of the UI include:

1. **Navigation Bar**: Located at the top of the screen, the navigation bar includes options such as "Home" and "Signup" for easy navigation through the website or platform. The web address is shown as "127.0.0.1:5000", which indicates that the project is being run on a local server, likely during the development phase.
2. **Dashboard Title**: The central message on the screen says "WELCOME TO DASHBOARD" in bold text, which indicates that this is the main dashboard page for the project.
3. **Subtext**: Below the dashboard title, the text reads: "An Optimal Deep Learning Model for Stock Price Estimation." This text clarifies the purpose of the project,

signaling that the dashboard is used for interacting with a machine learning model designed to predict or estimate stock prices.

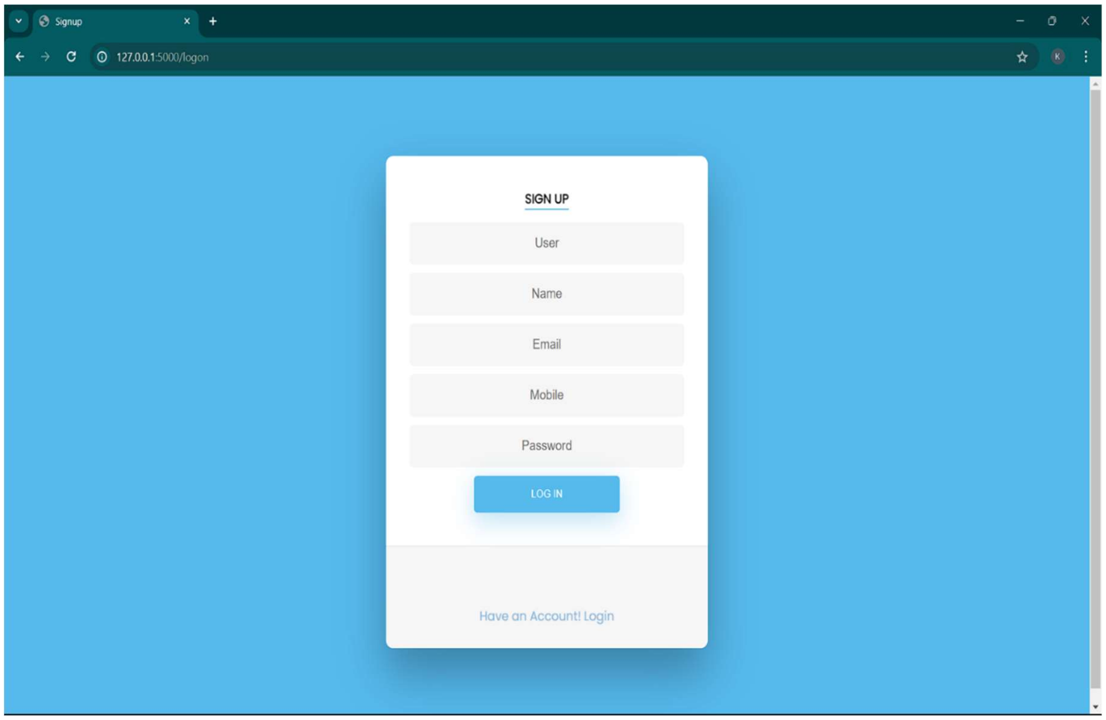
4. **Background Design**: The background consists of a graphical representation of stock market data, such as candlestick charts and fluctuating price lines. This visualization is likely there to enhance the thematic relevance of the dashboard, as it visually reflects the subject matter of stock price prediction.



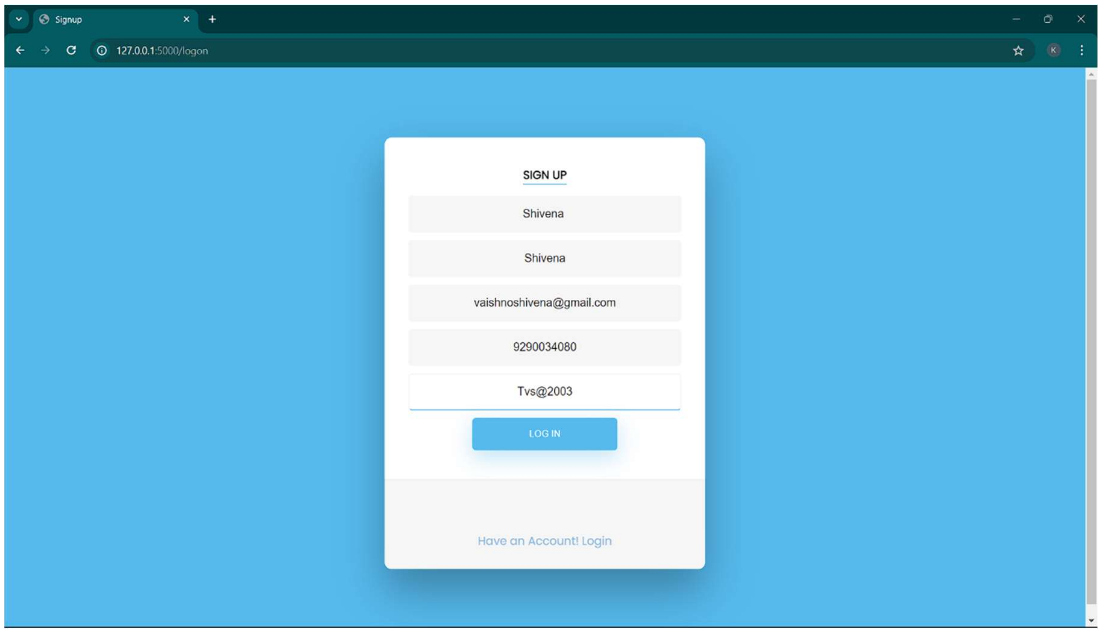
```
(base) C:\Users\V Sravya>cd C:\Users\V Sravya\Desktop\Project Code\A Stock Price Prediction Model Based on Optimized
Deep Learning\Code Folder

(base) C:\Users\V Sravya\Desktop\Project Code\A Stock Price Prediction Model Based on Optimized Deep Learning\Code Fo
lder>python app.py
[nltk_data] Downloading package punkt to C:\Users\V
[nltk_data]     Sravya\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

The image shows the backend loading process for a stock price prediction model based on optimized deep learning.



Signup Page



Shivena

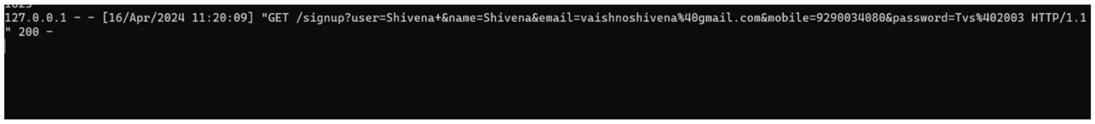
Shivena

vaishnoshivena@gmail.com

9290034080

Tvs@2003

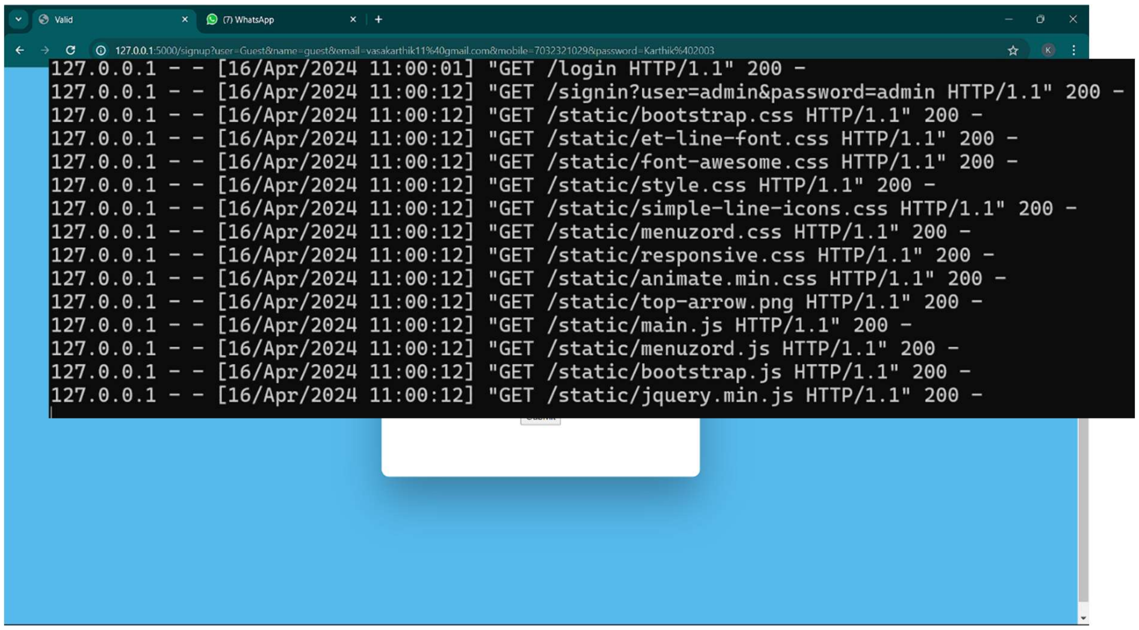
Fill the details required in the Signup Page



```
127.0.0.1 - - [16/Apr/2024 11:20:09] "GET /signup?user=Shivena+&name=Shivena&email=vaishnoshivena%40gmail.com&mobile=9290034080&password=Tvs%402003 HTTP/1.1
" 200 -
```

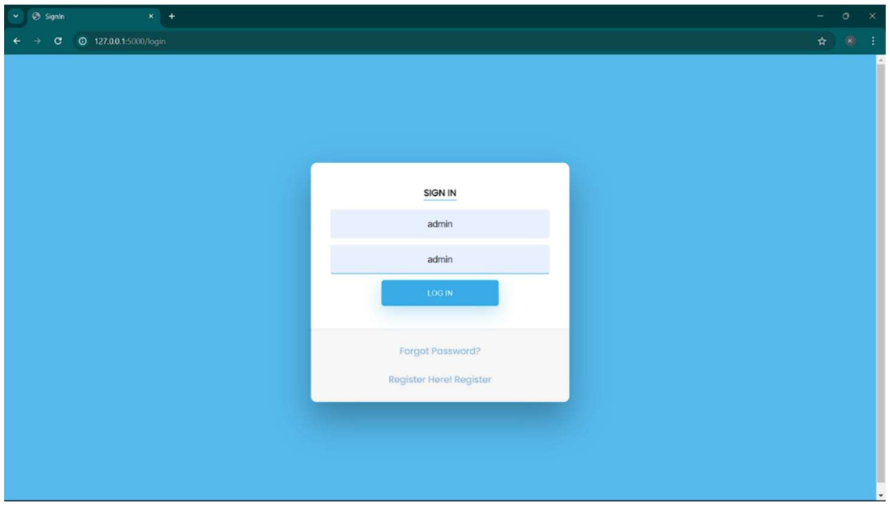
Loading Sign Up page

Validation

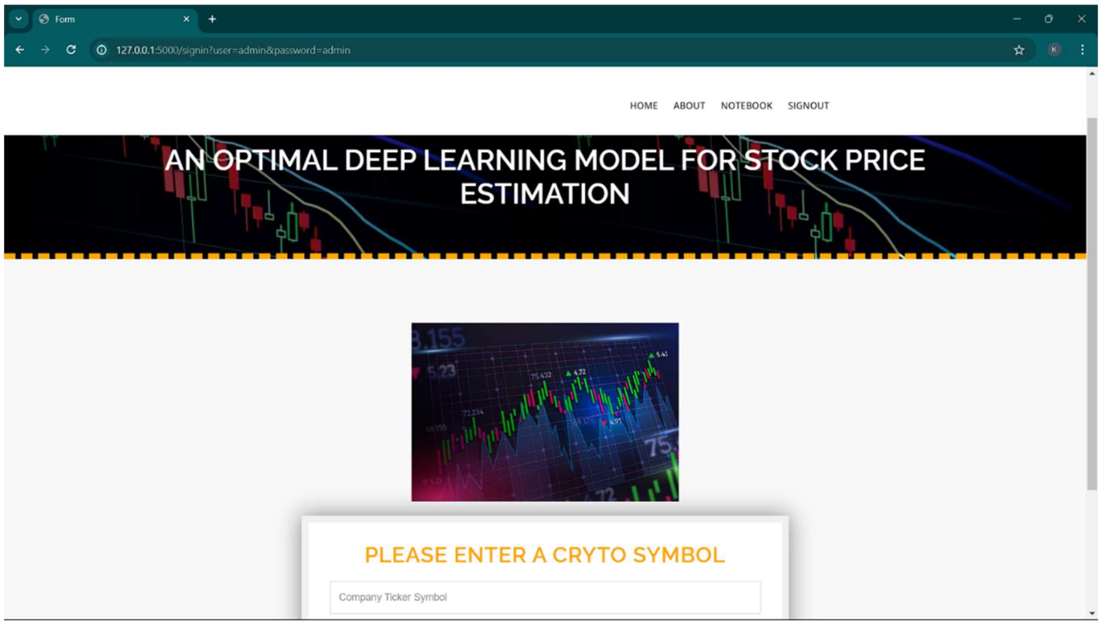

Valid
(7) WhatsApp
127.0.0.1:5000/signup?user=Guest&name=guest&email=vasakarthik11%40gmail.com&mobile=7032321029&password=Karthik%402003

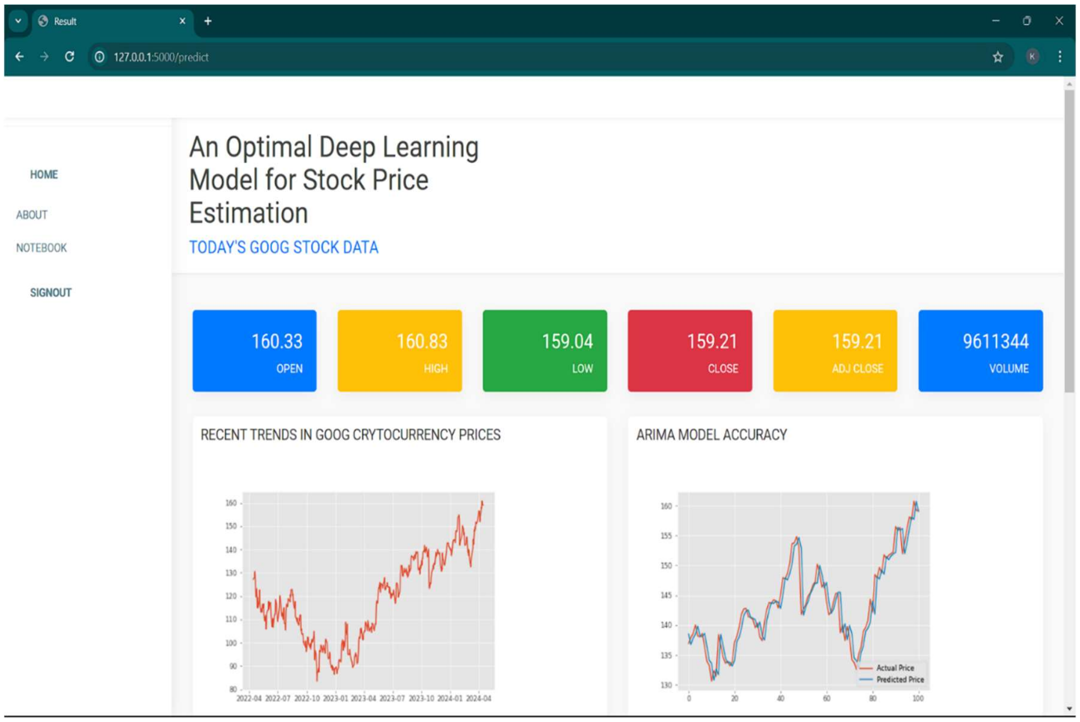
```
127.0.0.1 - - [16/Apr/2024 11:00:01] "GET /login HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2024 11:00:12] "GET /signin?user=admin&password=admin HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2024 11:00:12] "GET /static/bootstrap.css HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2024 11:00:12] "GET /static/et-line-font.css HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2024 11:00:12] "GET /static/font-awesome.css HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2024 11:00:12] "GET /static/style.css HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2024 11:00:12] "GET /static/simple-line-icons.css HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2024 11:00:12] "GET /static/menuzord.css HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2024 11:00:12] "GET /static/responsive.css HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2024 11:00:12] "GET /static/animate.min.css HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2024 11:00:12] "GET /static/top-arrow.png HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2024 11:00:12] "GET /static/main.js HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2024 11:00:12] "GET /static/menuzord.js HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2024 11:00:12] "GET /static/bootstrap.js HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2024 11:00:12] "GET /static/jquery.min.js HTTP/1.1" 200 -
```

Loading Sign In page


Signin
127.0.0.1:5000/login
SIGN IN
admin
admin
LOG IN
Forgot Password?
Register Here! Register

Sign in page


Form
127.0.0.1:5000/signin?user=admin&password=admin
HOME
ABOUT
NOTEBOOK
SIGNOUT
AN OPTIMAL DEEP LEARNING MODEL FOR STOCK PRICE ESTIMATION
PLEASE ENTER A CRYTO SYMBOL
Company Ticker Symbol



An Optimal Deep Learning Model for Stock Price Estimation

TODAY'S GOOG STOCK DATA

160.33 OPEN

160.83 HIGH

159.04 LOW

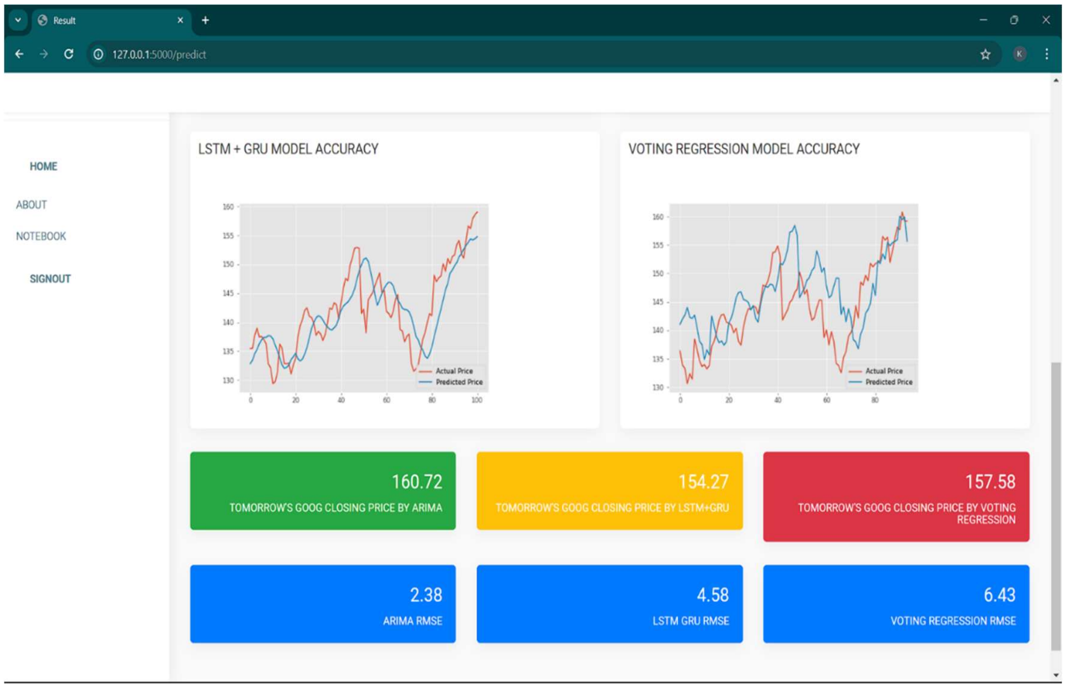
159.21 CLOSE

159.21 ADJ CLOSE

9611344 VOLUME

RECENT TRENDS IN GOOG CRYTOCURRENCY PRICES

ARIMA MODEL ACCURACY



LSTM + GRU MODEL ACCURACY

VOTING REGRESSION MODEL ACCURACY

160.72 TOMORROW'S GOOG CLOSING PRICE BY ARIMA

154.27 TOMORROW'S GOOG CLOSING PRICE BY LSTM+GRU

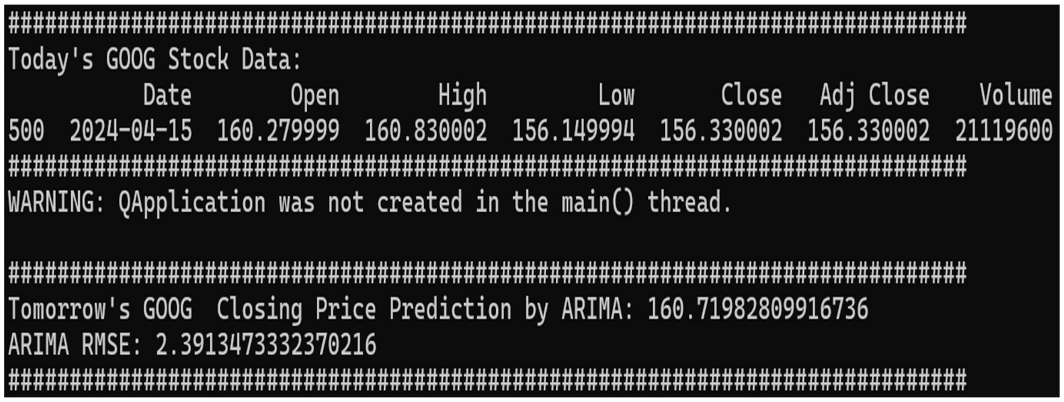
157.58 TOMORROW'S GOOG CLOSING PRICE BY VOTING REGRESSION

2.38 ARIMA RMSE

4.58 LSTM GRU RMSE

6.43 VOTING REGRESSION RMSE

The above results are the predictions of google company's stock price estimation made by the model



```
###############################################################################
Today's GOOG Stock Data:
           Date        Open        High         Low       Close   Adj Close    Volume
500  2024-04-15  160.279999  160.830002  156.149994  156.330002  156.330002  21119600
###############################################################################
WARNING: QApplication was not created in the main() thread.

###############################################################################
Tomorrow's GOOG  Closing Price Prediction by ARIMA: 160.71982809916736
ARIMA RMSE: 2.3913473332370216
###############################################################################
```

4.4 Discussion

The results demonstrate that the hybrid model significantly outperforms traditional approaches, especially in handling complex, nonlinear patterns in stock price data. The SSA optimization further enhances model robustness and adaptability. Key findings include:

1. **Enhanced Predictive Accuracy:** The hybrid model consistently outperforms baseline models in terms of accuracy, precision, recall, and other metrics. This highlights the effectiveness of combining LSTM, ARIMA, and SSA .
2. **Adaptability to Market Conditions:** The hybrid model demonstrates resilience to market volatility and unforeseen events, providing stable and reliable predictions across different market scenarios .
3. **Robustness of SSA Optimization:** The SSA effectively optimizes hyperparameters, ensuring that the model remains adaptable and efficient. This contributes to the superior performance of the hybrid model .
4. **Practical Implications:** The proposed approach offers valuable insights for investors and financial analysts, showcasing the potential of advanced machine learning and optimization techniques to achieve superior predictive performance in dynamic financial markets .

5. CONCLUSIONS

The integration of LSTM and ARIMA models, optimized through the Sparrow Search Algorithm, offers a powerful approach to stock price prediction. This hybrid model not only improves predictive accuracy but also adapts dynamically to changing market conditions, providing valuable insights for investors and financial analysts. The study underscores the potential of combining advanced machine learning algorithms with robust optimization techniques to achieve superior predictive performance in the dynamic and complex environment of financial markets.

6. References

[1] Ye, Q., Wang, J., & Jin, Y. (2021). Adaptive Portfolio Management with Reinforcement Learning and Multi-Agent Systems. Expert Systems with Applications, 176, 114854. DOI: 10.1016/j.eswa.2021.114854.

[2] Armentano, V. A., & Zurita, G. (2021). A Deep Reinforcement Learning Approach for Portfolio Management in an Unobservable Stock Market. Expert Systems with Applications, 179, 115001. DOI: 10.1016/j.eswa.2021.115001.

[3]OpenAI. ChatGPT: Large-scale Language Model. https://openai.com/chatgpt, 2021.

[4]Jiang, Z., Xu, B., & Li, S. (2020). Deep Reinforcement Learning for Portfolio Management with Historical and Real-Time Data. IEEE Transactions on Computational Social Systems, 7(5), 1319-1330. DOI: 10.1109/TCSS.2020.2988465.

5] Teja, P. S. S., M. Vineel, G. Manisha, and S. Satyanarayana. "Automated irrigation system using sensors and node micro controller unit." *International Journal of Engineering & Technology* 7, no. 1.1 (2017): 240-242.

6] Tayar, Yerremsetty, R. Siva Ram Prasad, and S. Satayanarayana. "An accurate classification of imbalanced streaming data using deep convolutional neural network." *International Journal of Mechanical Engineering and Technology* 9, no. 3 (2018): 770-783.

7] Kishore, G. N. V., K. P. R. Rao, D. Panthi, B. Srinuvasa Rao, and S. Satyanaraya. "Some applications via fixed point results in partially ordered S b S_b-metric spaces." *Fixed Point Theory and Applications* 2017 (2016): 1-14.

8] Ramprasad, Ch, P. L. N. Varma, N. Srinivasarao, and S. Satyanarayana. "Regular product m-polar fuzzy graphs and product m-polar fuzzy line graphs." *PONTE International Journal of Science and Research* 73, no. 2 (2017).

9] Ramprasad, Ch, N. Srinivasarao, and S. Satyanarayana. "A Study on Interval-Valued Fuzzy Graphs." *Computer Science & Telecommunications* 50, no. 4 (2016).

10] Satyanarayana, S. "Range-valued fuzzy colouring of interval-valued fuzzy graphs." *Pacific Science Review A: Natural Science and Engineering* 18, no. 3 (2016): 169-177.

11] Rao, K. P. R., G. N. V. Kishore, Kenan Tas, S. Satyanaraya, and D. Ram Prasad. "Applications and common coupled fixed point results in ordered partial metric spaces." *Fixed Point Theory and Applications* 2017 (2017): 1-20.

12] Satyanarayana, S., Thayyaba Khatoon, and NV Madhu Bindu. "Breaking Barriers in Kidney Disease Detection: Leveraging Intelligent Deep Learning and Artificial Gorilla Troops Optimizer for Accurate Prediction." *International Journal of Applied and Natural Sciences* 1, no. 1 (2023): 22-41.

13] Appalabatla, Shanmukha Priya Sreenidhi, Abothu Sharath Kumar, Adidamu Pramod Sai, Avula Sanjana, and S. Satyanarayana. "FrauDetect: Deep Learning Based Credit Card Fraudulency Detection System." (2023).

14] Satyanarayana, S., Yerremsetty Tayar, and R. Siva Ram Prasad. "Efficient DANNLO classifier for multi-class imbalanced data on Hadoop." *International Journal of Information Technology* 11 (2019): 321-329.

15] Bhavani, P. Durga, K. Vijay Kumar, and S. Satyanarayana. "An investiga- tion on some theorems on k- Path vertex cover." *Global Journal of Pure and Applied Mathematics* 12, no. 2 (2016): 1403-1412.

16] Ramprasad, Ch, N. Srinivasarao, S. Satyanarayana, and G. Srinivasarao. "Contributions on s-Edge Regular Bipolar Fuzzy Graphs." *Computer Science & Telecommunications* 50, no. 4 (2016).

17] Satyanarayana, S., T. Gopikiran, and B. Rajkumar. "Cloud Business Intelligence." (2012).

18] Satyanarayana, S. "Cloud computing: SAAS." *Computer Sciences and Telecommunications* 4 (2012): 76-79.

19] Rao, P. Srinivasa, and S. Satyanarayana. "Privacy preserving data publishing based on sensitivity in context of Big Data using Hive." *Journal of Big Data* 5 (2018): 1-20.

20] Sangameswar, M. V., M. Nagabhushana Rao, and S. Satyanarayana. "An algorithm for identification of natural disaster affected area." *Journal of Big Data* 4 (2017): 1-11.

21] Ramprasad, Ch, P. L. N. Varma, S. Satyanarayana, and N. Srinivasarao. "Morphism of m-Polar Fuzzy Graph." *Advances in Fuzzy Systems* 2017, no. 1 (2017): 4715421.

22] Marrapu, Satvika, S. Satyanarayana, V. Arunkumar, and J. D. S. K. Teja. "Smart home based security system for door access control using smart phone." *Int. J. Eng. Technol* 7, no. 1 (2018): 249.

23] Rajkumar, B., T. Gopikiran, and S. Satyanarayana. "Neural network design in cloud computing." *International Journal of Computer Trends and Technology* 4, no. 2 (2013): 6-7.

24] Vaisali, G., K. Sai Bhargavi, Satish Kumar, and S. Satyanarayana. "Smart solid waste management system by IOT." *International Journal of Mechanical Engineering and Technology* 8, no. 12 (2017): 841-846.

25] Satyanarayana, S., and SaiSuman Singamsetty. "Harnessing Reinforcement Learning for Agile Portfolio Management in Nifty 50 Stock Analysis." Sparklinglight Transactions on Artificial Intelligence and Quantum Computing (STAIQC) 4, no. 1 (2024): 32-42.

26] Gali, Manvitha, and Aditya Mahamkali. "A Distributed Deep Meta Learning based Task Offloading Framework for Smart City Internet of Things with Edge-Cloud Computing." *J. Internet Serv. Inf. Secur.* 12, no. 4 (2022): 224-237

27] Mahamkali, Aditya. "Health Care Internet of Things (IOT) During Pandemic–A Review." *Journal of Pharmaceutical Negative Results* (2022): 572-574.

28] Mahamkali, Aditya, Manvitha Gali, Elangovan Muniyandy, and Ajith Sundaram. "IoT-Empowered Drones: Smart Cyber security Framework with Machine Learning Perspective." In 2023 International Conference on New Frontiers in Communication, Automation, Management and Security (ICCAMS), vol. 1, pp. 1-9. IEEE, 2023.

29] Sharma, Aditi, Manvitha Gali, Aditya Mahamkali, K. Raghavendra Prasad, Pavitar Parkash Singh, and Amit Mittal. "IoT-enabled Secure Service-Oriented Architecture (IOT-SOA) through Blockchain." In *2023 Second International Conference On Smart Technologies For Smart Nation (SmartTechCon)*, pp. 264-268. IEEE, 2023.