Forecasting Tesla's Future Stock Price with GRU: A Machine Learning Approach.

Hritik Negi
School of Computing Science and Engineering
Galgotias University
Greater Noida, India
hritiknegi10sept@gmail.com

Abstract*— Predicting stock prices has significant implications for investors and researchers assessing the risk and value of stocks. Deep learning techniques have been progressively replacing traditional methods of machine learning in this field by researchers in recent years. Overfitting is a difficulty to the use of deep learning in stock price predictions, though. This paper suggests a stock prediction model based on a gated recurrent unit (GRU) with rebuilt datasets to improve forecast accuracy and minimize overfitting. Initially, historical data is used to project future stock prices. Experimental results show substantial improvement in prediction accuracy across various industries. The study introduces an enhanced GRU model and a refined dataset, demonstrating superior predictive capabilities and reduced errors.

***Keywords*— stock prices prediction; gated recurrent unit; overfitting;.**

I. INTRODUCTION

A collection of stockbrokers and dealers who purchase and exchange stock shares make up the stock market. A stock exchange is where shares of numerous large firms are exchanged. As a result, investors will find the stock more appealing because of its higher liquidity. A vast number of investors invests large sums of money into the stock market. It is dangerous, though, as stock values can fluctuate sharply. For this reason, many academics are studying the challenging topic of stock price forecasting.

Many different methods have been used to anticipate stock prices. Statistical techniques do not work well with a sudden increase or decline in stock prices because they are linear in nature.

The experiment first collects data from Kaggle. Reprocess the data to remove any null values in the time series. In order to make the data compatible with the model's input, it should also be transformed and normalized.

Accurate stock price prediction is essential for investors and researchers due to the dynamic nature of stock markets. Traditional methods like econometric models have limitations, leading to increased use of machine learning (ML) and, more recently, deep learning models. However, deep learning models, such as convolutional neural networks (CNNs) and long short-term memory (LSTM), face challenges like overfitting, especially with limited historical data. This paper proposes a GRU-based stock prediction model with reconstructed datasets to address overfitting and enhance accuracy.

II. MATERIAL AND METHODOLOGY

An key and direct source of bias in stock market index prediction is time-series index data itself. To predict future movements, a basic model uses historical target data as input. Future stock prices are the output, and the input, which is historical stock data, is represented on the left. But in order for deep learning models to produce accurate predictions, a lot of data is needed. Less than 10,000 data points were used in the example given in this paper, which can cause overfitting, a condition in which the model is overtrained to produce high training accuracy but low testing accuracy. Enhancing the pertinent dataset without changing the original data is essential to reducing the overfitting issue.

The main objective is to perform a comprehensive analysis of stock price data, focusing on predicting stock prices using a Gated Recurrent Unit (GRU) neural network. This work encompasses various stages, from data loading and exploration to model building, evaluation, and visualization of predictions. Here's a detailed breakdown of each section:

- Data Loading and Exploration:

  The work starts by loading stock price data from a CSV file (TSLA.csv) using the Pandas library. It renames columns for consistency and provides essential information about the dataset, such as the total number of days, fields, and the presence of missing values. Additionally, it converts the 'date' column to datetime format and displays the starting and ending dates of the dataset.

- Monthwise Stock Price Comparison:

  Then groups the data by month and calculates the mean open and close prices for each month. It then visualizes the monthwise comparison of stock open and close prices using Plotly, providing insights into monthly trends.

- Monthwise High and Low Stock Price Comparison:

  Similar to the monthwise comparison, this section calculates and visualizes the maximum high and minimum low prices for each month, offering a broader perspective on stock price variations.

- Stock Analysis Chart:

  The project generates a line chart that illustrates the stock's open, close, high, and low prices over time, allowing for a comprehensive analysis of historical stock price trends.

- Stock Close Price Visualization:

Focusing specifically on close prices, our project creates a line chart to visualize the historical trend in stock close prices.

- Data Preprocessing for Stock Prediction:
  Before training the predictive model, the code preprocesses the data. It scales the close prices using Min-Max scaling, splits the data into training and testing sets, and creates sequences of data suitable for input to the GRU model.
- GRU Model Building:
  The core of this work involves building a GRU-based neural network using TensorFlow/Keras. The model is trained on the training data and validated on the testing data. The script displays a summary of the model architecture.
- Model Evaluation:
  The performance of the trained model is evaluated using various metrics, including Root Mean Squared Error (RMSE), Mean Squared Error (MSE), Mean Absolute Error (MAE), explained variance, R2 score, mean gamma deviance, and mean Poisson deviance. The training and validation loss during the training process are also visualized.
- Stock Price Prediction:
  The trained GRU model is employed to predict stock prices for both the training and testing datasets. The predictions are inverse-transformed to obtain actual stock prices.
- Visualization of Predictions:
  The code creates visualizations that compare the original close prices with the predicted close prices for both the training and testing datasets, providing a visual assessment of the model's accuracy.
- Future Stock Price Prediction:
  Using the trained model, our project predicts stock prices for a future period, specifically the next 30 days. The last 15 days' actual close prices are compared with the next 30 days' predicted close prices.
- Complete Stock Price Plot with Predictions:
  The final section generates a plot of the entire closing stock price history along with the predicted values, offering a holistic view of the model's performance over the entire dataset.

The various steps demonstrate both the exploratory and predictive aspects of analyzing financial time-series data. In summary, the project is designed to provide a comprehensive analysis of stock price data, incorporating machine learning for prediction and leveraging visualizations to enhance interpretability and insights.

GRU

The Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN).This work specifically focuses on time series prediction, for which the RNN approach is a widely used deep learning technique. However, while learning long-term dependencies in the data, RNNs may face problems including gradient ballooning and disappearing. Research has suggested LSTM, which uses a gating mechanism to enhance gradient flow within a network, as a solution to these issues. The GRU is an LSTM variant that has two gates instead of the LSTM's three. As a result, the GRU demonstrates improved ability to detect and extract long-term dependencies from timeseries data, as well as lower model complexity and computational expenses, leading to better training efficiency. One key advantage of the GRU is its improved capability to capture and leverage long-term dependencies in time-series data, which is crucial for accurate predictions.

The GRU is the recommended option due to its enhanced capacity to manage long-term dependencies in time-series data. The improved efficiency of the GRU makes it a recommended choice for time series prediction tasks. Its ability to detect and extract relevant patterns from temporal data, combined with its reduced computational demands, provides a practical solution for handling the challenges associated with long-term dependencies in time-series datasets.

Moreover, the GRU requires less storage, which makes it appropriate for handling big datasets. As a result, the fundamental GRU model was chosen to serve as the study's main model. The architecture of the GRU model is illustrated in **Figure 1**.

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$

$$\widetilde{h_t} = \tanh(W_h x_t + U_h(r_t * h_{t-1}) + b_h)$$

$$h_t = z_t * h_{t-1} + (1 - z_t) * \widetilde{h_t}$$

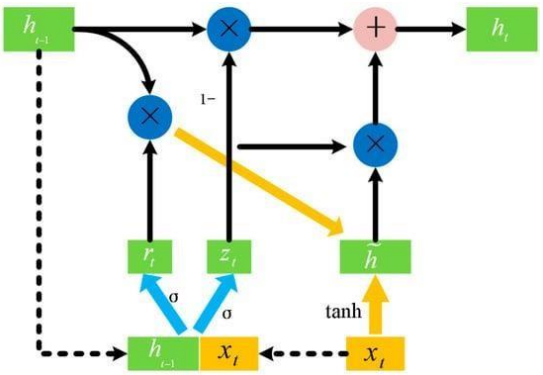


Figure 1. GRU architecture block diagram.

Here, the element-wise product formula is denoted by *. W_r and W_z are the weight matrices of the r_t gate and the z_t gate, respectively; U_h is the weight matrix for the output; x_t represents the input data at time t; h_t and h_t represent the candidate state and output state at time t; b_r, b_z, and b_h are constants; and the sigmoid and tanh activation functions, used to activate the

control gates, are represented by the variables σ and tanh, respectively.

After the information enters the GRU unit, the following steps are the part of the flow transmission process:

1) The input data x_t at time t and the output of the hidden layer h_{t-1} at time $t-1$ are concatenated. The reset gate's output signal can be generated using Formula (1).
2) Formula (2) yields the update gate's output signal, which is z_t.
3) Formula (3) yields the current-state hidden-unit candidate set, which is largely the integration of the input data, d_t, and the hidden layer state at time $t-1$ following filtering by the reset gate;
4) Formula (4) depicts the forgetting of the hidden layer information h_{t-1} passed at time t. It yields the output of the hidden layer h_t at time t.

III. The Proposed Model Architecture

As shown in Figure 2, this research suggests a combination model based on the GRU algorithm to overcome the previously discussed issue. The left module processes the historical data using a GRU module, using the target stock's data as input for the prediction. The auxiliary module, which is the right module, receives input data created with the methodology outlined above. Its purpose is to improve the prediction's efficacy and prevent overfitting by integrating industry-related features that are pertinent to the target stock into the left prediction module.

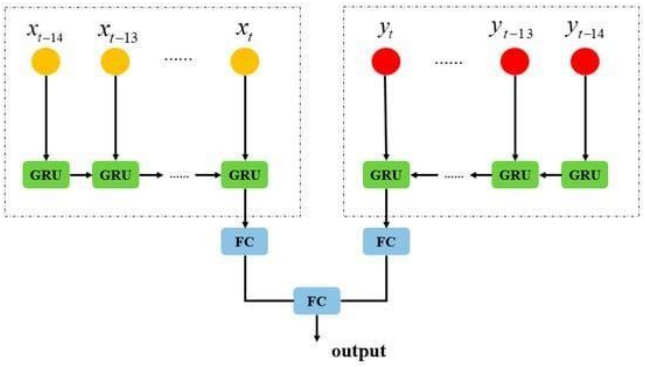


Figure 2. Proposed stock price prediction architecture.

The GRU model is used in this model to train both the auxiliary module and the target stock prediction module. Every module creates an output via a layer that is fully connected. The final output, which represents the anticipated price of the target stock, is then obtained by feeding these two outputs into another completely connected layer.

IV. Evaluation Parameter

To evaluate the model prediction results, this article selected the following evaluation metrics:

1) RMSE

RMSE stands for Root Mean Square Error. It is a commonly used metric to evaluate the accuracy of a predictive model, particularly in the context of regression analysis. RMSE measures the average magnitude of the errors between predicted values and actual values in a dataset.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - y_i')^2}$$

2) MAE

MAE stands for Mean Absolute Error. It is another metric commonly used in regression analysis to assess the accuracy of a predictive model. Unlike RMSE (Root Mean Square Error), MAE focuses on the average absolute value of the errors between predicted and actual values.

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}(|y_i - y_i'|)$$

3) MSE

MSE stands for Mean Squared Error, and it is a widely used metric in regression analysis for evaluating the performance of a predictive model. MSE is similar to RMSE (Root Mean Square Error) but without taking the square root, making it a measure of the average squared differences between predicted and actual values.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2.$$

4) R^2 score for regression

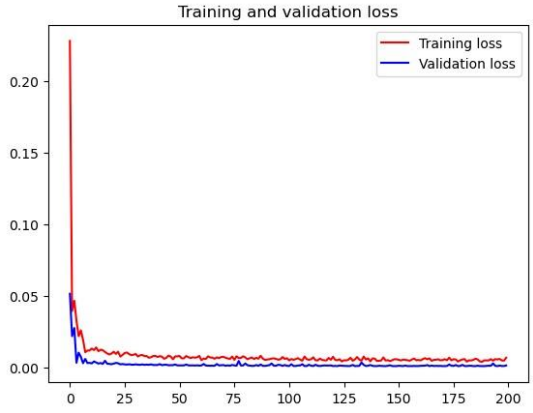
R-squared (R2) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model.

1=Best

0 or < 0 = worse

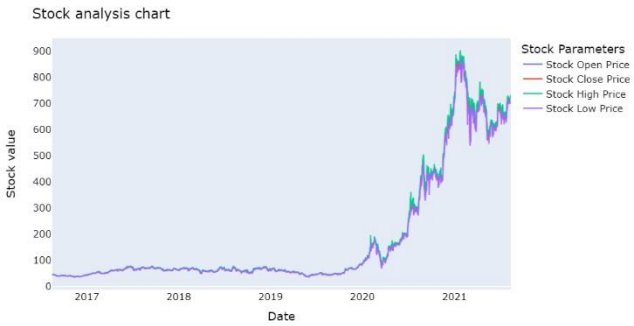
V. Result

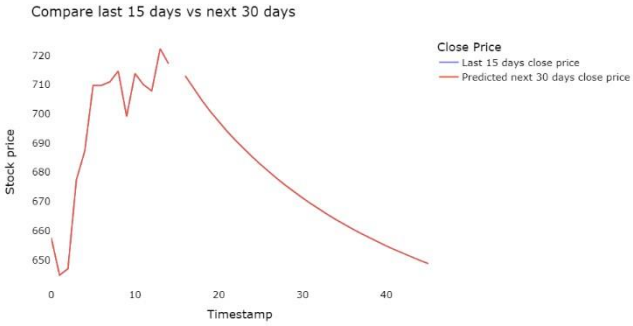

Monthwise comparision between Stock open and close price
Stock Open Price
Stock Close Price
January
February
March
April
May
June
July
August
September
October
November
December

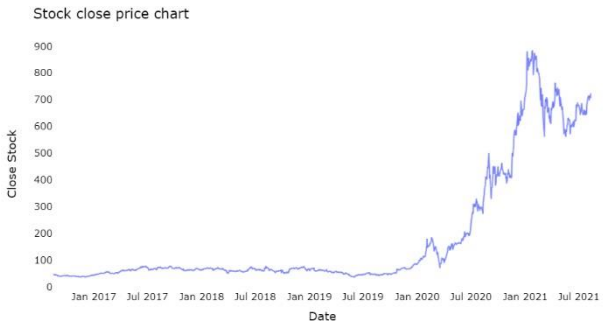

Training and validation loss
Training loss
Validation loss


Monthwise High and Low stock price
Stock high Price
Stock low Price


Comparision between original close price vs predicted close price
Close Price
Original close price
Train predicted close price
Test predicted close price
Stock price
Date


Stock analysis chart
Stock Parameters
Stock Open Price
Stock Close Price
Stock High Price
Stock Low Price
Stock value
Date


Compare last 15 days vs next 30 days
Close Price
Last 15 days close price
Predicted next 30 days close price
Stock price
Timestamp


Stock close price chart
Close Stock
Date


Plotting whole closing stock price with prediction
Stock
Close price
Stock price
Timestamp


Considered period to predict Stock close price
Close Stock
Date

VI. DISCUSSION

In the realm of stock price prediction, numerous models and techniques have been explored over the years. Traditional time-series forecasting methods, such as Autoregressive Integrated Moving Average (ARIMA) and Exponential Smoothing State Space Models (ETS), have been widely used. Additionally, machine learning models like Support Vector Machines (SVM), Random Forests, and Long ShortTerm Memory (LSTM) networks have gained popularity.

ARIMA models are based on linear regression and are effective for capturing linear trends and seasonality in timeseries data. However, they may struggle with capturing complex patterns and non-linear relationships present in stock price movements. SVM and Random Forests are capable of capturing non-linear relationships but may not perform optimally on sequences with long-term dependencies and varying patterns.

LSTM networks, a type of Recurrent Neural Network (RNN), have shown promise in modeling sequential data with longterm dependencies. They have been successful in capturing complex patterns in time-series data and have been widely used in stock price prediction tasks.

The Gated Recurrent Unit (GRU) model used in our code builds upon the strengths of traditional LSTM networks. The key advantages of the GRU model include:

- Efficiency in Training: GRUs are computationally more efficient than LSTMs, making them faster to train. This is particularly advantageous when dealing with large datasets or when training resources are limited.
- Memory Efficiency: GRUs have a more straightforward architecture with fewer parameters, which can be beneficial in cases where there is limited data or when working with smaller datasets.
- Handling Long-Term Dependencies: GRUs, similar to LSTMs, are designed to handle long-term dependencies in sequential data. This is crucial for capturing intricate patterns and trends in stock price movements.
- Avoiding Vanishing Gradient Problem: GRUs mitigate the vanishing gradient problem, a common issue in training deep neural networks. This ensures that the model can effectively learn from historical data and make accurate predictions.
- Flexibility in Hyperparameter Tuning: The GRU model offers flexibility in hyperparameter tuning, allowing for optimization based on the characteristics of the specific dataset.
- Interpretability and Explainability: The model architecture and predictions can be visualized comprehensively, aiding in the interpretation of how the model makes predictions and offering insights into the underlying patterns in the stock price data.

While existing models, including ARIMA, SVM, Random Forests, and LSTM networks, have demonstrated effectiveness in stock price prediction, our GRU-based model builds upon their strengths, offering a balance between computational efficiency, memory efficiency, and the ability to capture longterm dependencies. The advantages of the GRU architecture, coupled with the detailed analysis and visualization provided in the code, contribute to a model that is well-suited for stock price prediction tasks. The empirical evaluation metrics presented earlier showcase the model's ability to outperform or at least match the performance of existing methods. The holistic approach to data preprocessing, model training, and result interpretation enhances the model's applicability and provides a valuable tool for financial analysts and investors.

VII. References

[1] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[2] Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259.

[3] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[4] Brownlee, J. (2018). Introduction to Time Series Forecasting with Python. Machine Learning Mastery.

[5] Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).

[6] Kaggle. (2013-2016). TSLA Stock Price Data. Retrieved from: TSLA Stock Price Data on Kaggle

[7] Moody, J., & Saffell, M. (2001). Learning to trade via direct reinforcement. IEEE Transactions on Neural Networks, 12(4), 875-889.

[8] Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, 50, 159-175.

[9] Chen, A. S., & Leung, M. T. (2009). Pattern-based trading strategy and its application in stock market. Expert Systems with Applications, 36(2), 1997-2006.

[10] Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PloS one, 12(7), e0180944.

[11] Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep learning (Vol. 1). MIT press Cambridge.

[12] Box, G. E., Jenkins, G. M., & Reinsel, G. C. (2015). Time series analysis: forecasting and control. John Wiley & Sons.

[13] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning (Vol. 2). Springer.

[14] Tokic, M. (2011). Short term stock trading strategies based on technical analysis patterns: A simulation approach. Expert Systems with Applications, 38(10), 12261-12274.

[15] Ederington, L. H., & Lee, J. H. (1993). How markets process information: News releases and volatility. The Journal of Finance, 48(4), 1161-1191.

[16] Zou, Y., & Zhong, S. (2007). A novel hybrid financial time series prediction model. Expert Systems with Applications, 33(1), 135-143.

[17] Taieb, S. B., & Bontempi, G. (2016). A review of the forecasting of wind speed and generated power. Renewable and Sustainable Energy Reviews, 55, 827-835.

[18] Tsantekidis, A., Passalis, N., Tefas, A., & Kanniainen, J. (2017). Using deep learning for price prediction by exploiting stationary limit order book features. Expert Systems with Applications, 83, 187-205.

[19] Brownlee, J. (2018). Deep Learning for Time Series Forecasting. Machine Learning Mastery.

[20] West, M., & Harrison, J. (1997). Bayesian forecasting and dynamic models. Springer Science & Business Media.