University of British Columbia

Advanced Machine Learning Tools for Engineers

EECE 571T

Stock Price Forecasting

Authors:
Menghong Huang
Zhaosheng Li

April 29, 2021

Abstract

This paper implements various machine learning (ML) models to forecast the stock price of an American gaming merchandise retailer, namely, *GameStop Corporation*. The objective of this paper is to compare the performance of different ML models when being trained with financial time-series data. The stock market is a constantly changing and complicated environment. This attribute makes stock price forecasting become a challenging task in both industry and academia, because there exists numerous factors that cannot be predicted by the prediction models but significantly affect the actual stock price. But a prediction with high accuracy is still of paramount importance, since an accurate forecast on the stock price can dramatically mitigate the financial risks for stockholders and investors. This motivates researchers and scholars to work in this field. Inspired from the previous relevant work, this paper proposes several ML models, which include linear regression (LR), support vector regression (SVR), backpropagation neural network (BPNN), simple recurrent neural network (RNN), long short-term memory (LSTM) network, and gated recurrent units (GRU) network. According to the final experimental results in this work, the LR and LSTM outperform with relatively lower root mean square error (RMSE) and higher coefficient of determination (R^2), when compared to the other models.

1 Background

Stock price forecasting is regraded as one of the most challenging applications in modern time-series prediction. This is mainly due to the extreme complexity and volatile nature of the stock market. This attribute leads people to look for effective and accurate predictions on the stock price to guide their stock trading. A stock price forecast with high precision can identify the patterns in the stock price movement and is fairly necessary, especially for the stock traders and investors. An accurate forecasting of the future stock prices not only helps them make appropriate investments, but significantly mitigates the financial risks for them. Therefore, a robust and reliable tool, which can help stockholders and investors gain insight of the stock price movement, is of great importance.

With the rapid development of artificial intelligence, a variety of ML models have been rolling out. Some of those models are capable of capturing the pattern in the time-series data. Plus, there exists some nonlinear relationships in the stock price data, and some ML models are acknowledged as a good fit for identifying the nonlinear characteristics. In a sense, utilizing ML models to predict the stock prices becomes a promising option, and the application of ML in stock price forecasting becomes an active area of research.

Over the years, many scholars have committed to forecasting the stock price by using various ML models. [1] compares the performance of different kernel functions when applying the SVR to forecast option prices. [2] uses SVR with windowing function to predict stock market prices. [3] illustrates that SVR is a promising alternative to stock market prediction. [4] proposes to apply the SVR with adaptive parameters for accommodating to the dynamic changes in stock market. [5] utilizes an improved backpropagation (BP) algorithm and compares the single-input and multi-input BPNN for stock prices prediction. [6] deploys both SVR and BPNN on a financial time-series data set to do the forecasting and shows that SVM outperforms the BPNN, whereas [7] proves BPNN beats SVR when training the two models with different data sets. [8], [9], [10], and [11] utilize LSTM to predict the financial time-series data and finally get promising results. [12] uses LSTM to predict the next-day stock index price according to the historical data. [13] designs a LSTM-based model to simultaneously predict the open price, the lowest price, and the highest price of a stock on the next day.

The data set we are using in this work is the GameStop historical stock prices, which is publicly available on *Kaggle*. In this data set, the price of the *Gamestop* stock (GME) is dated from 2002-02-13 to 2021-01-28. Each entry has the key features of daily stock trading, which are open price, highest price, lowest price, trading volume, close price, and the adjusted close price. We set the close price as the target variable for all the ML models in this work. The data set has 4773 entries in total. We separate the first 80% entries into the training set, and the rest 20% data is used for testing purpose.

The rest of the paper is organized as follows: Section 2 describes the problems we are solving in this work. Section 3 presents detailed information of all the methods we are using in this project. Section 4 demonstrates all the experimental results and provides an in-depth discussion on the results. Section 5 concludes the paper along with an outlook on the future work.

2 Proposal

In this work, we propose to implement six ML models to forecast the price of GME, which include LR, SVR, BPNN, simple RNN, LSTM, and GRU. After training these models with the training set, we use the trained models to conduct two types of forecasts on the test set. The two kinds of forecasts are the next-day forecast and the 30-day forecast. The performance of each model is compared based on several evaluation metrics. Observing the results of performance, we find out the most appropriate model for each forecast scenario.

2.1 Next-day Forecast

The target variable in next-day forecast is the GME close price right after today. Most of the ML models in this work have the stock prices in the latest consecutive four days as input, which are depicted on Figure 1. The exceptions are the LR model and SVR model. Since those two models only accept 1-dimension (1D) vector as input, the close price on today is considered as input feature when applying LR and SVR to conduct the next-day forecast. An accurate next-day forecast can tell the investors if the stock deserves an investment on the next day.

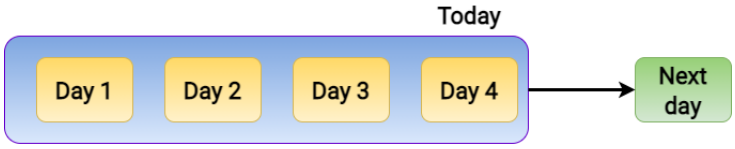


Figure 1: Next-day forecast

2.2 30-day Forecast

In the 30-day forecast scenario, the target variable switches to the GME close price on the $30th$ day after today. To accommodate to the increasing time range, we change the input of most ML models into the close prices on the latest thirty days in a row, as illustrated on Figure 2. Once again, the input to LR model and SVR model is still the close price on today because of their intrinsic attribute aforementioned. A precise 30-day forecast can provide investors and stock traders with insight about the stock price movement a month ahead.

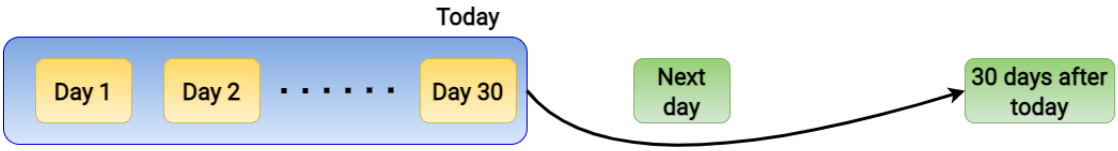


Figure 2: 30-day forecast

2.3 Evaluation Metrics

The performance of the models are based on the test set and quantified with the following indicators:

- **Root Mean Square Error (RMSE):** Measuring the RMSE can tell us the extent of difference between the actual price and predicted price. Equation 1 shows the formula to calculate RMSE, where $\hat{y}_i$ refers to the predicted GME close price, y_i is the actual GME close price, and N is the size of test set.

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(y_i - \hat{y}_i)^2}{N}} \tag{1}$$

- **Coefficient of Determination (R^2):** Measuring R^2 can tell us goodness of fit between the actual stock price and the predicted stock price movement. Equation 2 is the formula for computing the R^2, where $\overline{y}$ refers to the mean value of the test set.

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \overline{y})^2} \tag{2}$$

- **Trend prediction accuracy:** According to [2], trend forecasting becomes an essential topic in the stock market business. Thus, we add the trend prediction accuracy into the evaluation metrics. This indicator can tell us the performance of each model on trend forecasting. Equation 3 shows the formula for calculating the accuracy.

$$Accuracy = \frac{\sum_{i=1}^{N} \mathbb{1}\left[\text{sign}(y_{i+1} - y_i) = \text{sign}(\hat{y_{i+1}} - \hat{y}_i)\right]}{N} \tag{3}$$

3 Methods Used and Explained

This section illustrates the reason behind applying each ML model in this work. Also, this section elaborates on how are we implementing each model by providing a number of pseudocode algorithms.

3.1 *LR*

The LR model is a fairly straightforward ML model. It is directly imported from `sklearn` python module and is widely used in the realm of statistics. It is needless to tune any parameter in the LR model but feeding the input into the model. The LR model determines the best fit model with optimal coefficients of intercept and slope, when receiving the input stream. In essence, the optimization process is minimizing the cost function through gradient descent, and the cost function is usually referring to the mean squared error (MSE) function. Equation 4 shows the MSE function, where $\hat{y}_i$ refers to the predicted GME close price, y_i is the actual GME close price, and N is the size of test set.

$$MSE = \sum_{i=1}^{N} \frac{(y_i - \hat{y}_i)^2}{N} \tag{4}$$

Algorithm 1 depicts the pseudocode algorithm of utilizing linear regression model in the training phase. The LR model is working as a reference for the SVR model in this work.

Algorithm 1 LR algorithm

Input: Training set $D = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^{N}$. Close price on each day. $\dim(\mathbf{x}_k) = 1$
Output: $\hat{\mathbf{y}}_k$. Predicted close price in the next day ***OR*** after one month. $\dim(\hat{\mathbf{y}}_k) = 1$
1: Feed input into the model
2: **repeat**
Minimize MSE function by modifying intercept and slope coefficients
3: **until** The gradient stops descending

3.2 *SVR*

The SVR model is known for its ability to capture the pattern in time-series data, and it is a combination of support vector machine and regression model. The LR model is trying to minimize the cost function to its minimum value, whereas the SVR model is decreasing the cost function into a threshold range instead. The threshold range is determined by the kernal function in the SVR model. We choose the radial basis function (RBF) as the kernel because this kernal function just needs us to tune two parameters, which are the penalty parameter and γ. The penalty parameter, which is denoted by C, is essentially a regularization parameter, while the γ refers to the reciprocal of the distance from the support vector to the regression line. According to [14], the RBF function transfers the data from a lower dimension into a higher dimension to help the model better fit the data. Plus, a large value of C and a small value of γ can guarantee a small error threshold range and take fairly number of data points into account simultaneously. Algorithm 2 shows how are we implementing the SVR model in this work.

Algorithm 2 SVR algorithm

Input: Training set $D = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^{N}$. Close price on each day. $\dim(\mathbf{x}_k) = 1$
Output: $\hat{\mathbf{y}}_k$. Predicted close price in the next day ***OR*** after one month. $\dim(\hat{\mathbf{y}}_k) = 1$
1: Set $C = 1000$ and $\gamma = 0.1$
2: Feed input into the model
3: **repeat**
Minimize cost function into the threshold range determined by C and γ
4: **until** The cost function value is within the threshold range

3.3 *BPNN*

Besides the widely used regression models as above, we also manage to utilize deep learning techniques to identify the price movement pattern. BPNN, the fully connected neural network, is the fundamental one and has good ability to identify non-linear relationship. Figure 3 depicts the BPNN model we have built for stock price forecasting.

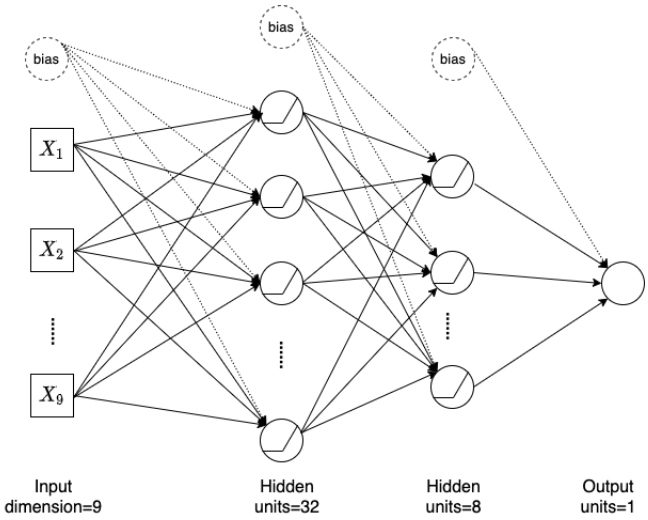


Figure 3: BPNN structure

For input layer in next-day forecasting, it takes into all features of the stock price on today, including open price, high price, low price, close price, volume and adjusted close price. As stock price is a kind of time-series data, the next-day price is probably related to the prices on today, yesterday and even several days before. In order to help BPNN learn the correlation of the time sequences, we look back on the close prices of previous 3 days and add them as 3 more inputs with the current 6 features. Thus, the input dimension is 9. To predict the close price after 30 days, previous 30-day close price are taken as 30 inputs to the BPNN model. Since BPNN model is sensitive to the input dimension and easy to overfit if too many features are involved, we only add close price as extra input when let BPNN look back previous information. Further, we apply z-score normalization to normalize all the inputs and the target variable to mitigate the effects of outliers. Equation 5 shows the z-score normalization, where μ is the mean value and σ is the standard deviation of the training set.

As for the hidden layers, both of them have rectified linear unit (ReLU) function as the activation function. The first hidden layer, which has 32 neurons, connects to the input layer and the second hidden layer, which has 8 neurons. While for the output layer, there is only one unit for the next day close price or the price after one month.

$$x_{scaled} = \frac{x - \mu}{\sigma} \tag{5}$$

With sufficient training and proper hyperparameter setting, the minimized loss can be reached and closest predicted price to actual value can be obtained. Algorithm 3 explains the implementation of BPNN model.

Algorithm 3 BPNN algorithm

Input: Normalized training set: $D = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^{N}$.
Next-day forecast: The 6 stock price features today and 3 close prices from latest consecutive days
$\dim(\mathbf{x}_k) = 9$
30-day forecast: Close prices of last 30 consecutive days
$\dim(\mathbf{x}_k) = 30$
Output: $\hat{\mathbf{y}}_k$. Predicted close price in the next day ***OR*** after one month. $\dim(\hat{\mathbf{y}}_k) = 1$
1: Weights initialization: ${w_{jk}}^{l}$ random normalized value in [0,1]
2: Set Relu as activation function in hidden layer
3: Set MSE as loss function and 'Adam' as optimizer.
4: Set number of training epochs as 100
5: **repeat**
6: Forward propagation to compute $\hat{y}$
7: Back propagation to optimize the weights and bias
8: **until** Model converged or total epochs reached

3.4 *RNN*

3.4.1 *Simple RNN*

As stock price movement is a time-sequence related problem, we manage to better utilize the dependencies in temporal direction. RNN is appropriate to process time-series data and identify pattern based on the correlation of previous trend. In addition to the input ($\mathbf{x}$) in time step t, the previous hidden state in last time step ($\mathbf{h}_{t-1}$) is also included to compute current hidden state ($\mathbf{h}_t$). Thus, it helps the model utilize the previous information as reference, like the mechanism of memory.

We build the simple RNN model with a many-to-one structure, as shown in Figure 4, because our goal is to predict the single price on next day or after one month for each entry. As RNN can input several days of transaction data in sequence, we include all features of previous data into input, which gives much richer previous information compared with BPNN. There are 3 hidden layers in our RNN model with both 64 units of hidden neurons, as depicted by Figure 5.

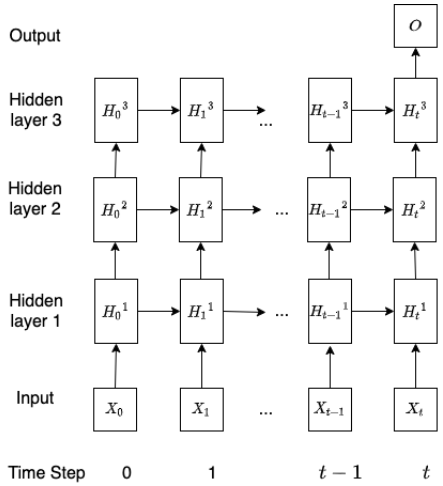


Figure 4: Many-to-one RNN structure

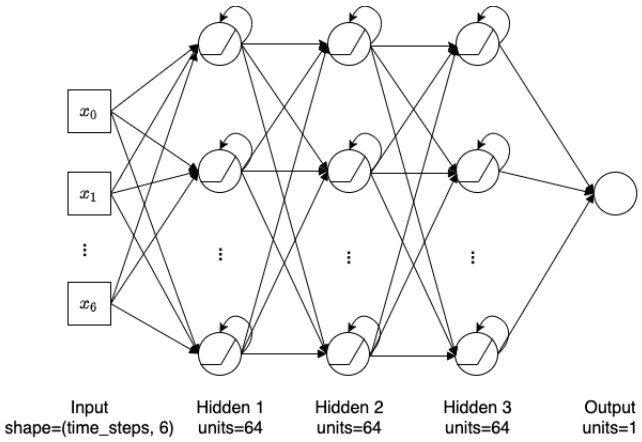


Figure 5: Layers in RNN Structure

For the 30-day price forecast, we take previous 30-day data in time sequence as input and close price after one month as target to train the model. Before feeding these data into the model, the value of all features are scaled via z-score normalization as well. In order to mitigate the gradient vanishing problem, we set recurrent dropout rate of 0.2 in hidden layers. The pseudocode for simple RNN is shown in Algorithm 4.

Algorithm 4 Simple RNN algorithm

Input: Normalized training set $D = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^{N}$.
Next-day forecast: All the stock price features in the latest four consecutive days, including today.
$\dim(\mathbf{x}_k) = (timesteps = 4, features = 6)$
30-day forecast: All the stock price features in the latest thirty consecutive days, including today.
$\dim(\mathbf{x}_k) = (timesteps = 30, features = 6)$
Output: $\hat{\mathbf{y}}_k$. Predicted close price in the next day ***OR*** after one month. $\dim(\hat{\mathbf{y}}_k) = 1$

```
Weights initialization: W_hx, W_hh, W_qh get random normalized value in [0,1]
Set Relu as activation function in hidden layer
Set recurrent dropout as 0.2
Set MSE as loss function and 'Adam' as optimizer
Set EarlyStopping(monitor='val_loss',mode='min',patience=10)
Set number of training epochs as 100
repeat
    Forward propagation to compute ŷ
    Back propagation through time to update weights and bias
until Early stop condition or total epochs reached
```

3.4.2 *LSTM and GRU*

To combat the gradient vanishing problem we encounter in the simple RNN model, we bring the advanced architectures of RNN into the table, which are the LSTM and GRU. To be on the same page with RNN, we still deploy the many-to-one structure and the number of units in each hidden layer is 64 as well. We build three stacked LSTM/GRU layers for the next-day forecast, whereas we designed four stacked LSTM/GRU layers for the 30-day forecast. Figure 6 and Figure 7 depict the structure of both LSTM and GRU on the next-day forecast and 30-day forecast respectively.

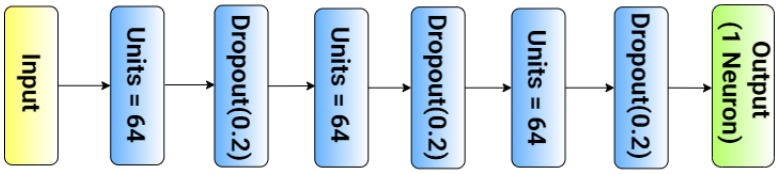


Figure 6: LSTM and GRU structure for the next-day forecast

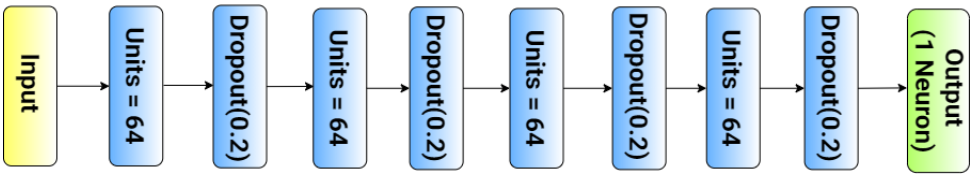


Figure 7: LSTM and GRU structure for the 30-day forecast

Since the activation function in LSTM and GRU is the sigmoid function or hyperbolic tangent function by default, we normalize all the data into the range between 0 and 1 to avoid overfitting in the training

process. To further avoid overfitting, we add a dropout layer with dropout rate of 0.2 in every hidden layer. Algorithm 5 reveals the pseudocode for employing LSTM and GRU in this work.

Algorithm 5 LSTM and GRU algorithm

Input: Normalized training set $D = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^{N}$.
Next-day forecast: All the stock price features in the latest four consecutive days, including today.
$\dim(\mathbf{x}_k) = (timesteps = 4, features = 6)$
30-day forecast: All the stock price features in the latest thirty consecutive days, including today.
$\dim(\mathbf{x}_k) = (timesteps = 30, features = 6)$
Output: $\hat{\mathbf{y}}_k$. Predicted close price in the next day ***OR*** after one month. $\dim(\hat{\mathbf{y}}_k) = 1$
1: Set MSE as loss function and 'Adam' as optimizer.
2: Set number of traning epochs as 100
3: Feed input into the model
4: **repeat**
Train the LSTM/GRU model
5: **until** The number of epochs reaches the 100

4 Results and Discussion

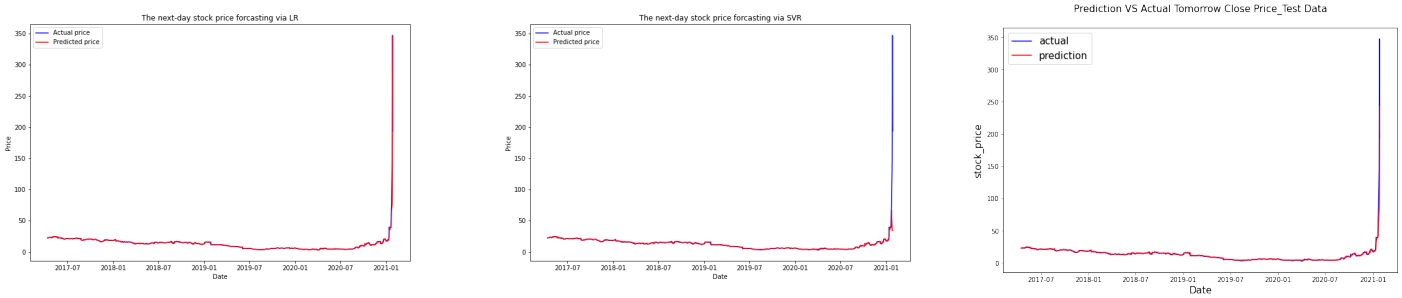
This section demonstrates all the experimental results in the format of table and figure. We test the performance of each trained model on the test set, and this section also includes an in-depth discussion on the test results we get in this work.

4.1 Next-day Forecasting Result

Table 1 shows all the results of evaluation metrics for each model. The RMSE is quite large, and the R^2 is below our expectation as well. This is due to some outliers in the test set, which will be talked about in the upcoming section. To improve the data visualization, we also plot the performance of each model on Figure 8.

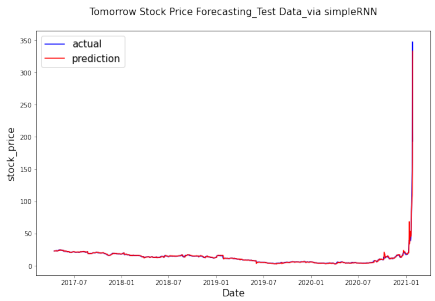
	LR	SVR	BPNN	Simple RNN	LSTM	GRU
RMSE	8.537	12.01	8.273	7.705	8.407	8.307
R^2	0.668	0.344	0.689	0.730	0.678	0.686
Accuracy	50.1%	48.6%	47.8%	50.9%	49.8%	50.1%

Table 1: Next-day forecast result

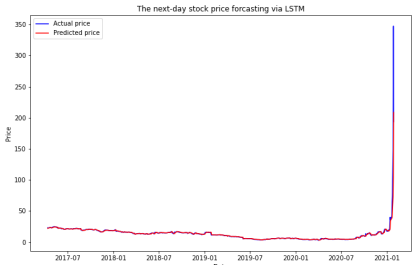


(a) LR (b) SVR (c) BPNN

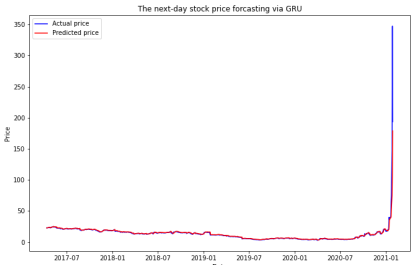
Figure 8: Next-day forecasting result vs actual price on test data



(d) Simple RNN



(e) LSTM



(f) GRU

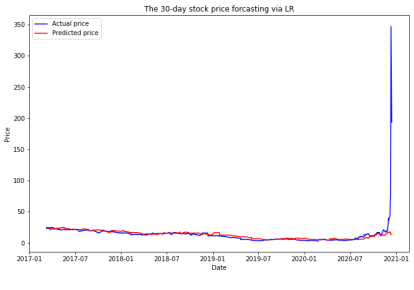
Figure 8: Next-day forecasting result vs actual price on test data (cont.)

4.2 30-day Forecasting Result

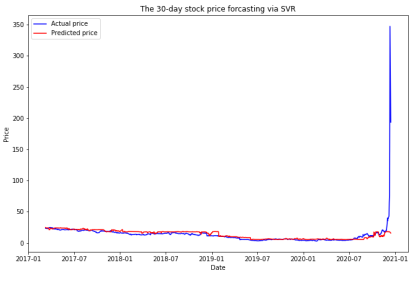
Table 2 shows the performance of each model on 30-day forecast scenario. It is totally acceptable that the performance of each model degrades as the length of time step increases. But with RMSE beyond 13 and R^2 lower than 0.2 are far more below our expectation. Observing all the plots on Figure 9, we see none of the models successfully predict the ending part of the test set, which is an unexpected spike. The spike is a significant reason for the inferior performance on both next-day forecast and 30-day forecast.

	LR	SVR	BPNN	Simple RNN	LSTM	GRU
RMSE	13.60	13.52	13.69	13.62	13.71	13.74
R^2	0.161	0.171	0.153	0.162	0.150	0.145
Accuracy	51.3%	48.3%	51.2%	49.4%	48.7%	50.3%

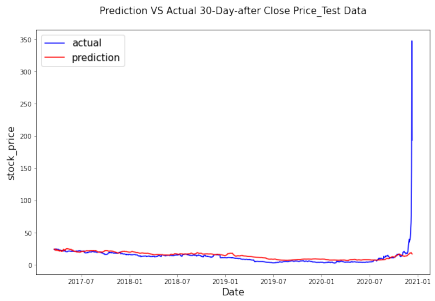
Table 2: 30-day forecast result



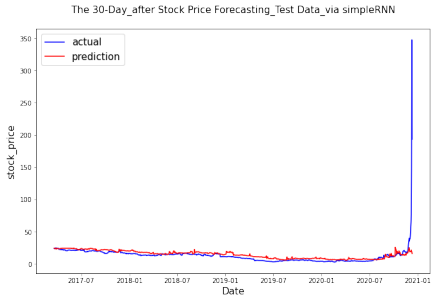
(a) LR



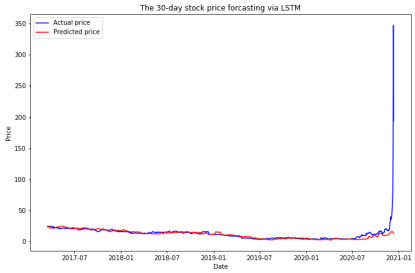
(b) SVR



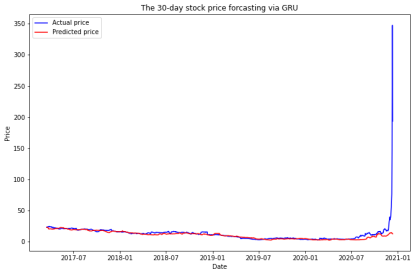
(c) BPNN



(d) Simple RNN



(e) LSTM

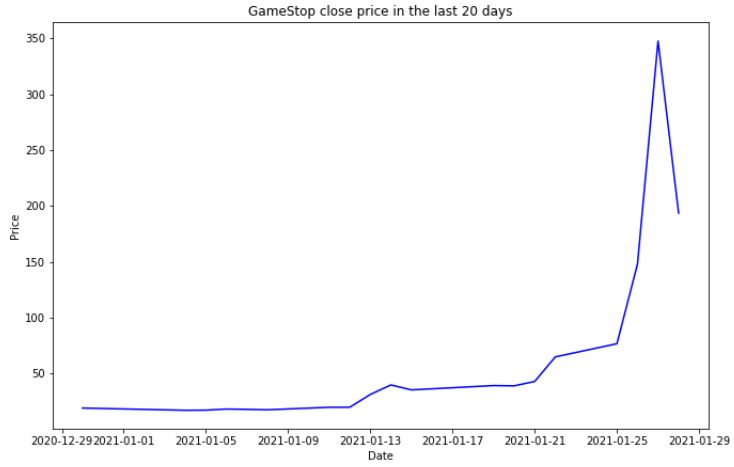


(f) GRU

Figure 9: 30-day forecasting result vs actual price on test set

4.3 Turning Point

We look up some mainstream media to find out why there exists a spike at the end of test set. Reading the pieces of news on [15] and [16], we acknowledge that there was a malicious competition on the GME price. That spiteful competition was mainly triggered by some short investors on `Reddit`. If we plot the GME close price on the last 20 entries on test set, as shown on Figure 10a, we can easily tell that the abnormal increasing happened since around January $13th$, 2021. Figure 10b further proves that the GME close price started rocketing after January $12th$, 2021, as the close price increased nearly 60% over 24 hours only. Since this is an uncommon phenomenon in the stock market, we decide to update our test set by eliminating the last 11 entries and test our trained models again with the updated test set.



(a) Plot

2021-01-04	17.25
2021-01-05	17.37
2021-01-06	18.36
2021-01-07	18.08
2021-01-08	17.69
2021-01-11	19.94
2021-01-12	19.95
2021-01-13	31.4
2021-01-14	39.91
2021-01-15	35.5
2021-01-19	39.36
2021-01-20	39.12
2021-01-21	43.03
2021-01-22	65.01
2021-01-25	76.79
2021-01-26	147.98
2021-01-27	347.51
2021-01-28	193.6

(b) Table

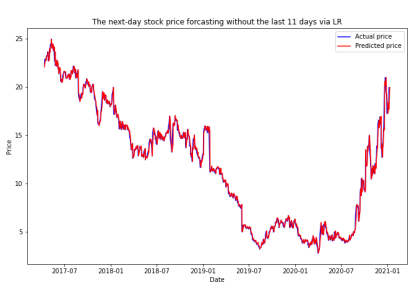
Figure 10: Last 20 entries on test set

4.4 Updated Next-day Forecasting Result

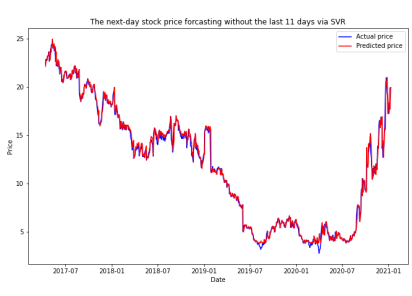
After eliminating the outliers from the test set, we find the performance on normal case is quite promising. Table 3 shows that all six models have much smaller RMSE and higher R^2 score that is close to 1. That means models can reliably predict the next-day price within acceptable tolerance. However, all models are not able to have good prediction on daily trend of direction with only around 50% accuracy. As shown in Figure 11, all models have good generalization on test data, especially for the LR model. But simple RNN has the biggest RMSE as there are several prediction with large distance to actual value. During its training process, the model loss is still high and stop decreasing at the middle period. SVR and BPNN both miss off prediction in the lowest area of prices below around 5 dollars. The potential reason is that they can hardly distinguish the minute difference of input as these smallest values after normalization is extremely close to 0, impacted by the far larger max values in the last 11 days. Among the 6 models, LR, LSTM and GRU are more effective to identify the pattern of price movement and LR have the best performance on the precise prediction of actual price values. Thus, there is a strong linear relationship between adjacent two days for GME stock.

	`LR`	`SVR`	`BPNN`	`Simple RNN`	`LSTM`	`GRU`
`RMSE`	0.502	0.528	0.557	0.669	0.583	0.663
R^2	0.993	0.993	0.992	0.988	0.991	0.988
`Accuracy`	50.1%	48.6%	47.6%	50.9%	49.6%	49.8%

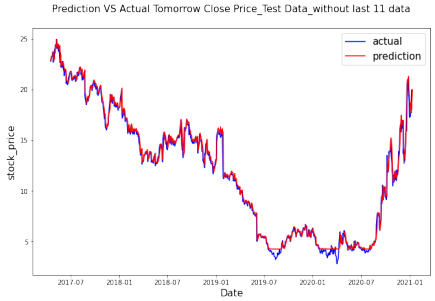
Table 3: Updated next-day forecast result



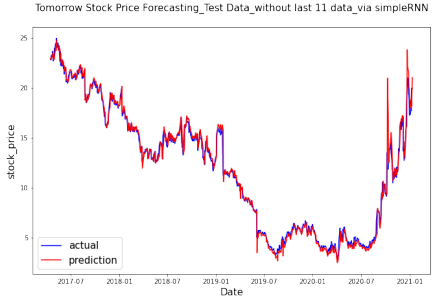
(a) LR



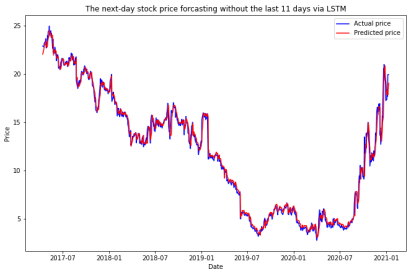
(b) SVR



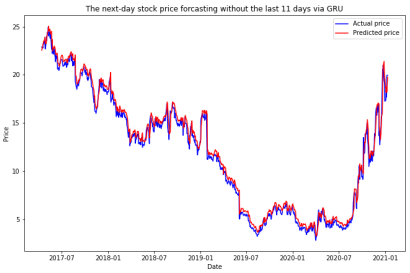
(c) BPNN



(d) Simple RNN



(e) LSTM



(f) GRU

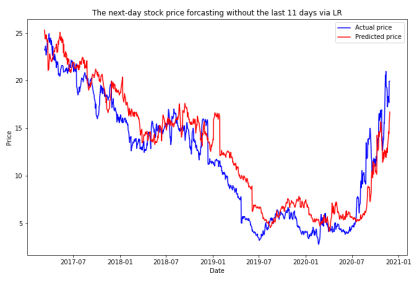
Figure 11: Next-day forecasting result excluding the last 11 entries

4.5 Updated 30-day Forecasting Result

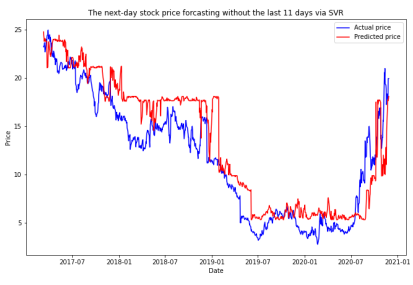
After excluding the last 11 data, we can see how these models learn and predict the nature of price movement after one month. Table 4 indicates all models have less precise prediction on the value of prices than that for the next-day prediction, as 30-day forecasting is more challenging. However, the predicted results from some models, like LR, LSTM, and GRU, are worth referring to since R^2 are beyond 0.8. As exhibited in Figure 12, LR, LSTM, and GRU have better goodness of fit, while BPNN and simple RNN have larger distance between the predicted price and actual price. SVR has the lowest capability to identify the price movement pattern because it has looser tolerance on the prediction error. That yields the predicted price is not that centralized to actual price compared with LR. BPNN has drawbacks on 30-day prediction since the data in next 29 days are unknown and not taken into input. Without the reference from the data of the closest days, BPNN lacks of the ability on precise long-term prediction. Simple RNN has large RMSE as the final model loss is still high in training. Since 30-day is a quite long sequence, the RNN model has the problem of gradient vanish. The loss value is not decreased after the only sharp decrease at the very beginning. LSTM and GRU overcome the vanishing gradient problem and have much smaller RMSE and larger R^2. LSTM slightly outperforms GRU both in next-day forecasting and 30-day prediction.

	LR	SVR	BPNN	Simple RNN	LSTM	GRU
RMSE	2.365	2.809	3.539	3.509	2.407	2.438
R^2	0.848	0.786	0.654	0.660	0.841	0.836
Accuracy	51.3%	48.3%	51.1%	49.1%	48.9%	50.3%

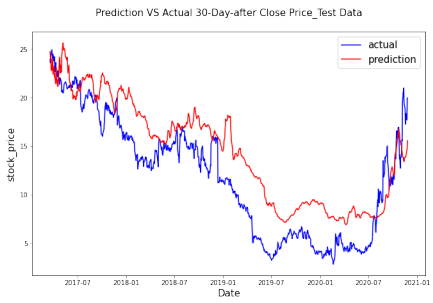
Table 4: Updated 30-day forecast result



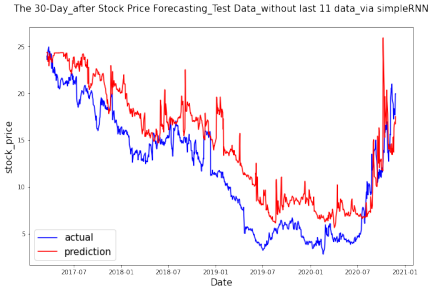
(a) LR



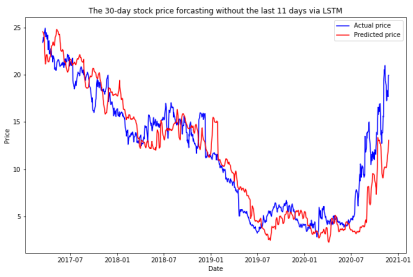
(b) SVR



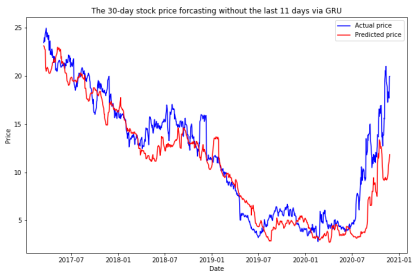
(c) BPNN



(d) Simple RNN



(e) LSTM



(f) GRU

Figure 12: 30-day forecast result excluding the last 11 entries

5 Conclusions and Future Work

In recent years, using ML techniques to predict stock price movement becomes a hotspot research topic in the field of quantitative trading. In this work, we investigate six ML models to conduct stock price forecasting on GME price in two different tasks, namely, the next-day forecast and the 30-day forecast. LR model and LSTM model eventually outperform when compared to the other models based on RMSE and R^2. The LR beats SVR is mainly because the SVR model has loose tolerance for the error. We also conclude that the BPNN has difficulty on longer-term prediction, and the simple RNN is prone to vanishing gradient problem when the length of time step is fairly large. Both of them are especially the case when we are applying BPNN or simple RNN to do the 30-day forecast. Plus, the LR model has an outstanding performance is because of the linear relationship in the GME close price, but this may only apply to the data set we have used here. Last but not least, an unexpected and unusual spike in the stock price can highly degrade the performance of prediction models, which has been proved by the last 11 entries on the test set.

As for the future development on this work, we plan to increase the trend prediction accuracy. As all the models in this work have that accuracy around 50%, we decide to change the type of optimizer and loss function that lead the prediction models to improve their performance on that accuracy. Also, we are thinking about enlarging the set of input features by adding some categorical variables, such as if there exists governmental issues, breaking news on the website, or some malicious competition in the stock market. Since ML models cannot learn directly from the categorical variables, some techniques, such as one-hot encoding, need to be taken into account at this situation. Lastly, since both next-day forecast and 30-day forecast belong to the short-term forecasting, switching to medium-term forecasting (prediction for 1 to 2 years) or long-term forecasting (prediction beyond 2 years) is another option on future work.

Acknowledgement

We would like to appreciate Dr. Matthew Yedlin and Dr. Bhushan Gopaluni for their feedback and suggestions during this work. Also, we are grateful to the Kaggle community and Yahoo finance website for generously offering the daily data of GameStop stock exchange we have used here.

References

[1] B. Madhu, A. K. Paul, and R. Roy, "Performance comparison of various kernels of support vector regression for predicting option price," *International Journal of Discrete Mathematics*, vol. 4, no. 1, p. 21, 2019.

[2] P. Meesad and R. I. Rasel, "Predicting stock market price using support vector regression," *2013 International Conference on Informatics, Electronics and Vision (ICIEV)*, pp. 1–6, 2013.

[3] K. Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, pp. 307–319, 2003.

[4] Y. Guo, S. Han, C. Shen, Y. Li, X. Yin, and Y. Bai, "An adaptive svr for high-frequency stock price forecasting," *IEEE Access*, vol. 6, pp. 11 397–11 404, 2018.

[5] W. Ma, Y. Wang, and N. Dong, "Study on stock price prediction based on bp neural network," *2010 IEEE International Conference on Emergency Management and Management Sciences*, 2010.

[6] F. E. Tay and L. Cao, "Application of support vector machines in financial time series forecasting," *Omega*, vol. 29, no. 4, p. 309–317, 2001.

[7] Y. Song, Y. Zhou, and R. Han, "Neural networks for stock price prediction," *arXiv e-prints*, p. arXiv:1805.11317, May 2018.

[8] J. Qiu, B. Wang, and C. Zhou, "Forecasting stock prices with long-short term memory neural network based on attention mechanism," *PLoS ONE*, vol. 15, 2020.

[9] A. Moghar and M. Hamiche, "Stock market prediction using lstm recurrent neural network," *Procedia Computer Science*, vol. 170, p. 1168–1173, 2020.

[10] S. Mehtab, J. Sen, and S. Dasgupta, "Robust analysis of stock price time series using cnn and lstm-based deep learning models," *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 1481–1486, 2020.

[11] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. Menon, and K. Soman, "Stock price prediction using lstm, rnn and cnn-sliding window model," *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1643–1647, 2017.

[12] P. Gao, R. Zhang, and X. Yang, "The application of stock index price prediction with neural network," *Mathematical Computational Applications*, vol. 25, p. 53, 2020.

[13] G. Ding and L. Qin, "Study on the prediction of stock price based on the associated network model of lstm," *International Journal of Machine Learning and Cybernetics*, vol. 11, pp. 1307–1317, 2020.

[14] D. Scatterday, "Walking through support vector regression and lstms with stock price prediction," Sep 2019. [Online]. Available: https://towardsdatascience.com/walking-through-support-vector-regression-and-lstms-with-stock-price-prediction-45e11b620650

[15] A. Morrow, "Gamestop's stock is going through some stuff. you can thank reddit," *CNN Business*, Jan 2021. [Online]. Available: https://www.cnn.com/2021/01/25/business/nightcap-gamestop-budweiser-super-bowl-ads/index.html

[16] T. Adinarayan, "Gamestop frenzy explained: How small investors on reddit took on wall street," *Global News*, Jan 2021. [Online]. Available: https://globalnews.ca/news/7607462/gamestop-frenzy-explained/

Appendix

Please click here to access to our code on GitHub. The GitHub repository contains the final version of code in Python, trained models, images of experimental results, documentation for this project, and the data set used in this work. In case the previous link does not work, we also attach the link to the repository here: `https://github.com/zhaoshengEE/Stock_Price_Forecasting`