EE140: Final Project

Team Member: Boyang Lyu, Tiange Wang

1. Introduction

Since a random process is a collection of random variables that are indexed by time [1], a time series is also called discrete-time random processes. It's a sequence of values collected over time on a particular variable.[2] Time series analysis has a wide range of applications, such as weather forecasting, economic growth forecasting, trading markets, hydrological, and even astronomy. For these values which involve time components, traditional forecasting methods are mainly based on statistical models, including Auto-Regressive Moving Average (ARMA) Models, Autoregressive Integrated Moving Average (ARIMA) Models and Exponential Smoothing Models (ETS).

In recent years, application of machine learning technology is developing rapidly in the time series forecasting area and this kind of problem is often regarded as a supervised problem[3]. Thus, the question which technique is more accurate and suitable motivates us to explore and compare the two types of methodologies.

More specifically, we would like to explore the statistical methods mentioned above, learn the theory behind them and do related simulation. Since Recurrent Neural Network(RNN) is widely used for modeling time series data as a machine learning method and among them, the Long Short Term Memory(LSTM) model is said to be designed for dealing with time components, we choose it as a comparison method [4].

Here we would like to do time series data forecasting by both stochastic and machine learning models. After forecasting, we'll compare their performances, including prediction accuracy and computational complexity. By doing this, we hope to learn more about essential stochastic process properties and get some insights of the advantages and disadvantages for both prediction methods.

2. Time Series Analysis Models

2.1 Auto Regressive(AR) Models

In a discrete time series, each data point can be expressed by a combination of a random error (perhaps a white noise) component and a linear combination of prior data points, such as Equation (1) below,

$$x_t = \sigma + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + \epsilon_t \tag{1}$$

where x_t is data point of variable *x* at time *t*, x_{t-p} (*p*=0,1,...) is previous data points of variable *x* at time *t-p*, and *p* is the order of auto regressive process; ϕ_p is autoregressive model parameter; ε_t is the random error component.

The AR model can be stable only if all the model parameters lie within a certain range. Otherwise, the previous data points will accumulate and the successive values of x_t will move toward infinity. Thus the time series will be non-stationary, which makes the model meaningless.

AR models have lots of limitations, as discussed above, the data should be stationary and can only be applied to predict phenomena which is related to previous processes (auto-correlation) [2].

2.2 Moving Average(MA) Models

Apart from the serial dependence of the data points as in case of AR processes, each point in time series can also be affected by the past random error than can't be affected by AR models. Thus, moving average models are generated to optimize time series analysis. The model equation can be expressed as below.

$$x_t = \mu + \sigma_t - \theta_1\sigma_{t-1} - \ldots - \theta_q\sigma_{t-q} \tag{2}$$

where μ is a constant or population mean; σ_{t-q}(q=0,1,...) are random error components of previous data points at time *t-p*. In MA models, the present data point x_t is't related to the previous data point, but only depends on a linear combination of historical white noises, which is different from the AR model.[2]

2.3 Autoregressive Moving Average (ARMA) Models

Auto-Regressive Mean Average model is a combination of AR and MA models and can be expressed as follows.

$$x_t = \mu + \phi_1 x_{t-1} + \phi_2 x_{t-1} + ... + \phi_p x_{t-p} + \sigma_t - \theta_1\sigma_{t-1} - ... - \theta_q\sigma_{t-q} \tag{3}$$

When q=0 and $p \geq 1$, equation (3) should be an AR(p) process. Whereas with p=0 and $q \geq 1$, it will be an MA(q) process.[2]

2.4 AR, MA, ARMA in System Identification

Assume that the random sequence $X(t)$ is generated as the output of an unknown filter with transfer function $H(z)$. Write $X(t)$ in discrete form $X[k]$ [1]. Input is the white noise $W[k]$ (corresponding to ϵ_t in Eq.1). The basic types of systems are AR, MA, and ARMA models. For a particular $x[k]$, we want to estimate parameters of $H(z)$. Thus, when the estimated system is driven by white noise, its output closely approximates $x[k]$ [1].

Due to input is white noise with variance σ_W^2, AR function of the output is the inverse z-transform of the following power spectral density function:

$$S_{YY}(z) = \sigma_W^2 H(z)H(z^{-1}) \tag{4}$$

AR, MA, ARMA models can also be viewed as filters.

AR model for random sequence $X[k]$ is described by the following equation:

$$X[k] = \sum_{n=1}^{N} a_n X[k-n] + W[k] \tag{5}$$

where N is the order of the model. This model is an infinite-impulse-response(IIR) filter operating on the input sequence white noise $W[k]$.

MA model is a finite sum of the weighted white input sequence as follows:

$$X[k] = \sum_{n=0}^{M-1} b_n W[k-n] \tag{6}$$

where *M* is the order. This model is an FIR filter.

Like section 2.3 above, ARMA is a combination of AR and MA models. The random sequence is as follows:

$$\begin{aligned} X[k] &= \sum_{n=1}^{N} a_n X[k-n] + \sum_{n=0}^{M-1} b_n W[k-n] \\ &= \mathbf{a}^T \mathbf{X}[k-1] + \mathbf{b}^T \mathbf{W}[k] \end{aligned} \tag{7}$$

where ***a*** and ***b*** are the same as previously defined.

2.5 Autoregressive Integrated Moving Average (ARIMA) Models

Though powerful and widely used, ARMA poses a strong assumption for the time series data, which is stationary. However, in practice, stationary can not always be guaranteed. For example, data related to business activities usually contains trend or periodic components and thus non-stationary [2]. To handle non-stationary cases, another model called Autoregressive Integrated Moving Average (ARIMA) Model is proposed in [5]. Generally speaking, ARIMA is an extension of the ARMA model. It gains the ability to handle non-stationary data by differencing it to make it stationary.

ARIMA consists of three main parts, AR, MA and Integration. The integration part will "difference" the data, for example, the first differencing value is obtained by subtracting previous time period from current time period. ARIMA can be formulated as

$$(1 - \sum_{i=1}^{p} \phi_i L^i)(1-L)^d y_t = (1 + \sum_{i=1}^{q} \theta_j L^j)\epsilon_t \tag{8}$$

where *p, d, q* are non-negative integers, *p* and *q* represent the order of autoregressive and moving average parts, d controls the level of difference.

2.6 Stationary checking

Different from other types of data, time series data has special features. Time series data, as the name indicates, has a close connection with the time component, which provides additional useful information. Thus, when building models, special characteristics of time series data should be considered as first. Otherwise, simply implementing the models, though perhaps the model predictions seem to follow the real value closely and show good accuracy, the results could be misleading and totally wrong.

Consider a random process *X*(*t*), we say it is *first-order* stationary if its probability density function (PDF) doesn't depend on time [1]:

$$f_{X(t)}(x) = f_X(x), \forall t \in \mathcal{T} \tag{9}$$

where $\mathcal{T}$ is a set of all time instants. Then, it can be said that *X*(*t*) is *N-order* stationary if the joint PDF at *N* time instants doesn't change by any time shift τ of all $\{t_1, ..., t_N\}$:

$$f_{X(t_1),...,X(t_N)} = f_{X(t_1-\tau),...,X(t_N-\tau)}(x_1, ..., x_N) \tag{10}$$

which is stronger than *first-order* stationary, and implies that all moments across time instants are unchanged due to a time shift. Moreover, *X*(*t*) is *strictly* stationary if it is *N-order* stationary for all *N* as $N \to \infty$.

However, it's difficult for real-world data to be strictly stationary. Thus, a process is said to be *wide sense stationary* if the mean is constant, and its autocorrelation function depends only on the time difference of two samples and not on specific time instants:

$$\mathcal{E}[X(t)] = \mu_X, \forall t \in \mathcal{T} \text{ and } C_{XX}(t_1, t_2) = C_{XX}(t_2 - t_1), \forall t_1, t_2 \in \mathcal{T} \quad (11)$$

On the other hand, if a series of data is non-stationary, such as some *random walk* data, it actually can't be predicted. Due to this, the idea of using historical data as the training data set in order to learn the behavior and predict future outcomes is impossible [6]. As a result, it's necessary to determine if the data is stationary before dealing with it, because stationary property has essential influence on selecting models.

There are several stationary checking methods, including:

(1) Look at plots: plot a time series, and visually check if there are any obvious trends [7]

(2) By statistical tools, check if the average of data is constant, if the variance always exists, and if the autocorrelation doesn't change over time.

(3) Use statistical unit root tests if expectations of stationarity are met or violated such as Dickey–Fuller(DF) test, Augmented Dickey Fuller(ADF) test

The basic idea of ADF test is that if the series is integrated then the lagged level of the series $X(t\text{-}1)$ will provide no relevant information in predicting the change in $X(t)$. The null hypothesis is that the time series has a unit root, implying that the time series is integrated, which implies non-stationary [9]. Here we'll examine the *p*-value of ADF test. The larger *p*-value is, the more tendency of non-stationary the data has, where *p*-value ranges from 0 to 1. For white noise(Gaussian process), it has zero *p*-value, which means white noise is a totally stationary random process.

2.7 Autocorrelation function and Partial Autocorrelation Function

Still let$\{x_t\}$ denote a series of random variables indexed by time *t*. As mentioned above, wide sense stationary time series have time invariant first and second moments. The autocorrelations of $\{x_t\}$ are defined by Equation (12) as follows.

$$\rho_j = \frac{cov(x_t, x_{t-j})}{\sqrt{var(x_t)var(x_{t-j})}} \quad (12)$$

where *j* implies the *j-th* order. A plot of ρ_j against *j* is called the Autocorrelation Function, which helps to define stationary time series. If $\{x_t\}$ is stationary, then $\{y_t\}$=$\{g(x_t)\}$ is also stationary for any function $g(\cdot)$.

What's more, Partial Autocorrelation Function is also a useful tool to help identify *p* values, which is related to the stationary test as mentioned above. The PACF is based on estimating the sequence of models. For example, for the AR models, there exists

$$\begin{aligned} y_t &= \phi_{11} y_{t-1} + \varepsilon_{1t} \\ y_t &= \phi_{21} y_{t-1} + \phi_{22} y_{t-2} + \varepsilon_{2t} \\ &\cdots\cdots \\ y_t &= \phi_{p1} y_{t-1} + \phi_{p2} y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_{pt} \end{aligned} \quad (13)$$

where $y_t = x_t - \mu$ is the demanded data. The coefficients are called partial autocorrelation coefficients. For an AR(*p*) all of the first *p* autocorrelation coefficients are non-zero, and the

rest are zero for $j > p$. The sample coefficients up to lag p are essentially obtained by estimating the above sequence of p AR models with least squares, and retaining the estimated coefficients.

3. Experiments

3.1 Dataset and Preprocessing

1. Dataset Introduction

Our dataset is stock data of General Electric during the year 2006-2018. Here is the visualization of our dataset. We will mainly focus on the closing price for each day over 12 years.

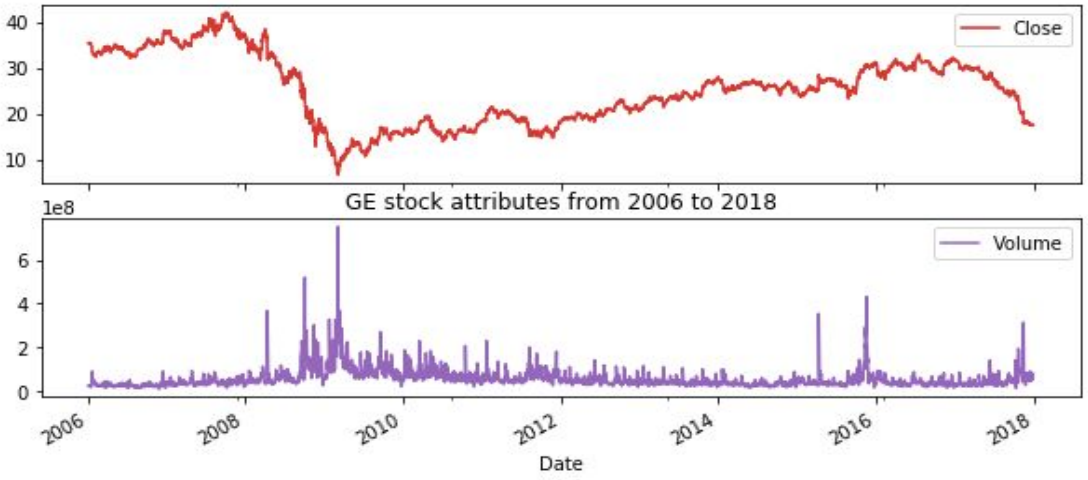


Figure 1. Stock Data of GE from 2006 to 2018

Figure.1 gives an overall scale of GE's stock data. Considering that volumes are too large to give an analysis which is easy to observe, we mainly discuss the closing price. Closing price generally refers to the last price at which a stock trades during a regular trading session (one day).

3.2 Stationary Checking and Differencing

In Fig.1, it's difficult to find if the closing price data is stationary by looking at the plot, though it can be discovered that during the whole time period, its mean value isn't constant. In order to verify if the data is stationary, two methods are employed here. As shown in Fig. 2 (top), the ACF of the data is around 1 and decreases slowly, which indicates that the data is non-stationary. An Augmented Dickey Fuller(ADF) test is also run and gives high probability (p-value is 0.55979) that there exists unit root, thus we can identify the data is non-stationary. Since the stochastic algorithm requires data to be stationary, we compute the differences between consecutive observations to make the non-stationary data stationary. After that, ADF test shows that the new p-value of the difference data is 3.71787×10^{-21} .

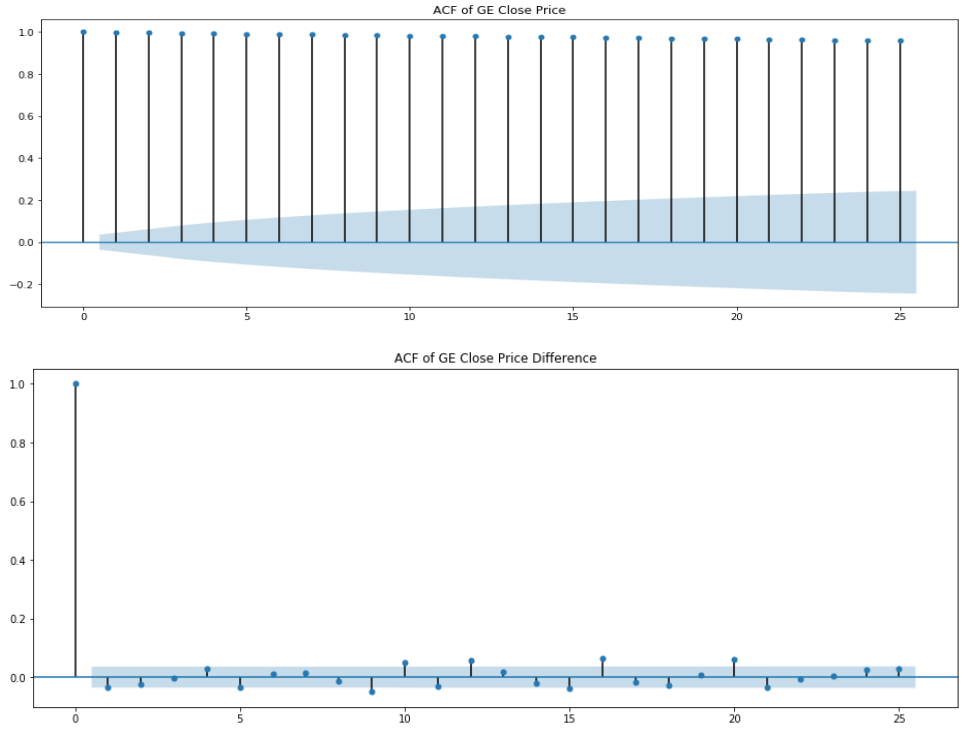


Figure 2. Autocorrelation function plot for GE close stock price (top)
and the difference of GE close stock price (bottom)

As shown in Fig.2 (bottom), after taking the difference, ACF is nearly unchanged around zero throughout time. That is, taking difference helps to reshape the data to be stationary.

3.2 Comparison Methods

In this report, Long Short Term Memory (LSTM) Recurrent Neural Network(RNN) model will be mainly applied to compare with statistical models.

In RNN algorithm, each neuron gets input from a set of other neurons or the problem input, weights each input to get a linear sum, and then applies an activation function to compute the corresponding output. Different from a traditional neural network, RNN contains recurrent inside. RNN also weights "past time" input. Their training procedure is a general way to propagate and distribute the prediction error to previous states of the network. The learning procedure consists of updating the model parameters by minimizing a suitable loss function, which includes the error achieved on the target task [10]. LSTM is a type of advanced RNN, which has more complex structure, and by controlling the "gates" which includes a sigmoid function, LSTM can add or subtract information. This characteristic enables it to deal with time series better [11].

Here 2 layers LSTM is applied, each of which has 32 hidden states, two dense layers are added after LSTM layers and to avoid overfitting, dropout rate is selected to be 0.2.

3.3 Training and predicting Procedure

As shown in Fig. 3, difference data is splitted into two parts, the first 95% data will be used to train the ARMA model and the last 5% is for testing. For LSTM and ARIMA models, we use raw data but the same train test split. For all three models, we use the model to make one-step

prediction, for the second step prediction, we pick the true value of the current time step from test data and feed the data to the model to get the next step prediction. This process will be repeated for multiple times until all predictions are gotten.

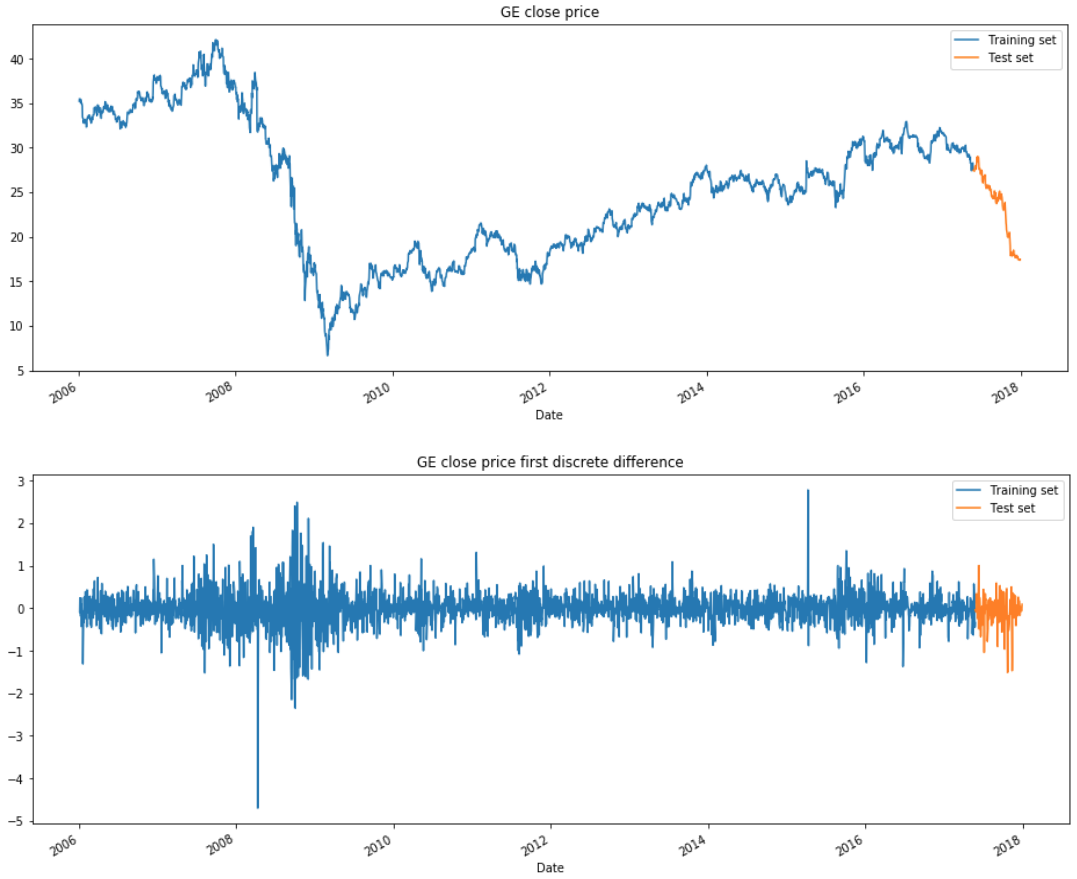
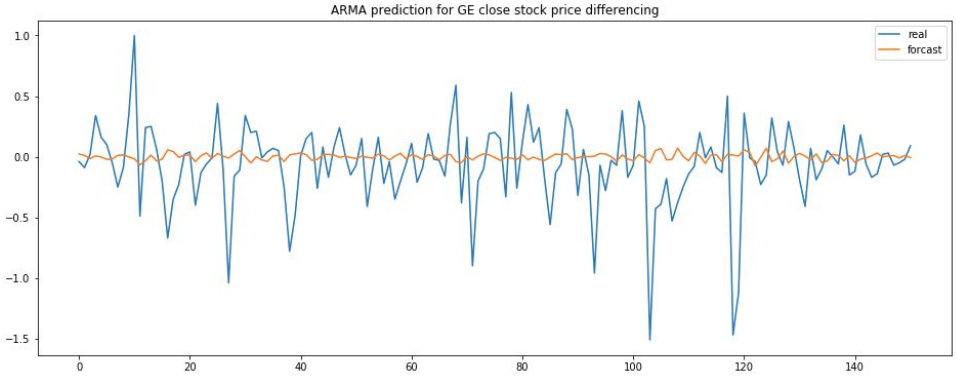


Figure 3. Training and testing data split for LSTM, ARIMA method (top) and ARMA method (bottom)

3.4 Performance of each model

3.4.1 ARMA models

Fig.4 shows ARMA forecasting of stock price (close). After predicting with the differencing values, an inverse transform is employed to show what the prediction value should be in the same form as the original data.



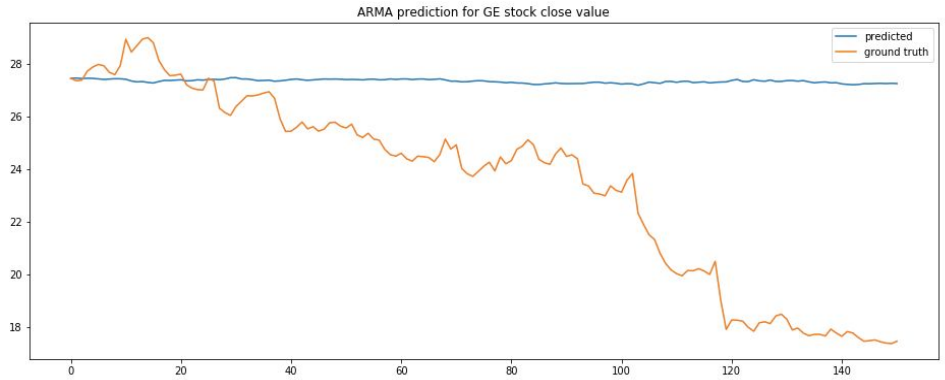


Figure 4. ARMA forecasting for the difference of stock price(top) and ARMA forecasting for the stock price (bottom)

As Fig.4 shows, with differencing value nearly unchanged during time, the forecast result via ARMA totally deviates from what it should be.

3.4.2 ARIMA Model

Fig.5 below provides prediction of stock price by ARIMA models. This prediction closely follows the true value. Actually, ARIMA has the best performance among the three models we applied.

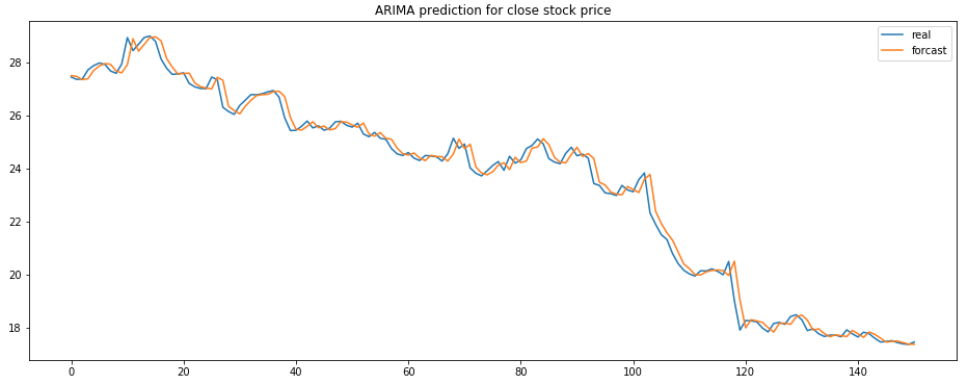


Figure 5. ARIMA forecasting for the stock price

3.4.3 LSTM model

Forecasting results by LSTM-RNN model is in Fig.6. The prediction curve also follows the trend of true value.

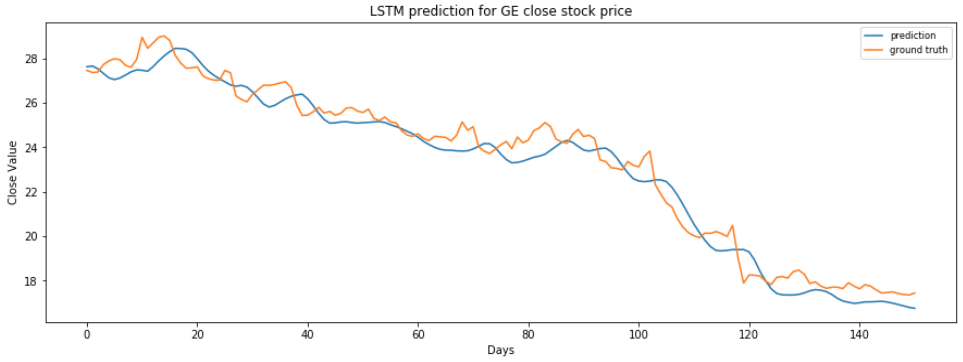


Figure 6. LSTM forecasting for the stock price

3.4.4 Test error

In order to examine accuracy of each model, Mean Square Errors(MSE) of forecasting results and testing data are computed and presented in Table.1.

The same as we discovered in Fig.4, Fig.5, and Fig.6, prediction of ARIMA model has the least MSE, which corresponds to its forecasting curve which follows the true value curve best. Next, LSTM also has a relatively low MSE, whereas its prediction curve just has a similar trend as true values, but in detail the points are slightly different. Last, ARMA model has a large MSE, and its prediction is also far away from the ground truth.

Table 1. Mean square error for test data

Method	ARMA	ARIMA	LSTM
MSE	27.60	0.120	0.166

4. Conclusions

Two stochastic methods ARMA, ARIMA and one machine learning method LSTM are used to do one-time step forecasting. Comparing the MSE and the prediction plots, we can see that ARIMA and LSTM methods achieve almost the same good results for one-step prediction. The result is consistent with the conclusion in [12], which means for the simple case as we used here, complex models are not necessarily better than simple models. In terms of computational complexity, since we only use a small amount of data, the speed of three methods don't vary too much.

There are several things we want to point out for the comparison. ARIMA works only if the difference of the data is stationary, which means it has more limitations than the LSTM model since the latter one doesn't pose such a requirement for the data, our experiment doesn't show this respect. Moreover, we only conduct one-time step prediction in the experiment, which may not be able to fully explore the models potential for prediction. Our test dataset lacks diversity, which may also affect the evaluation results.

We also found the stationary of the data has also been considered in the LSTM model as in [13], where the model proposed in the paper is inspired by ARIMA.

For next step work, diversity datasets from different areas should be tested and a more carefully hyper-parameter tuning procedure should be performed. Multi-step prediction should also be tested in order to see the limitations of different models.

Reference

[1] Shynk, John Joseph. Probability, Random Variables, and Random Processes Theory and Signal Processing Applications. Hoboken, NJ: Wiley, 2012.

[2] Machiwal, Deepesh, and Madan Kumar Jha. Hydrologic time series analysis: theory and practice. Springer Science & Business Media, 2012.

[3] Nielson, Aileen. Practical Time Series Analysis : Prediction with Statistics and Machine Learning. First ed. 2019.

[4] Makridakis, Spyros, Evangelos Spiliotis, and Vassilios Assimakopoulos. "Statistical and Machine Learning forecasting methods: Concerns and ways forward." PloS one 13.3 (2018).

[5] Box, George EP, et al. Time series analysis: forecasting and control. John Wiley & Sons, 2015.

[6] How to Check if Time Series Data is Stationary with Python:
https://machinelearningmastery.com/time-series-data-stationary-python/

[7] Time Series: Check Stationarity:
https://medium.com/@kangeugine/time-series-check-stationarity-1bee9085da05

[8] Upton, Graham, and Ian Cook. "Augmented Dickey–Fuller Test." *A Dictionary of Statistics* (2014): A Dictionary of Statistics.

[9] Tam, Pui Sun. "Finite-sample Distribution of the Augmented Dickey-Fuller Test with Lag Optimization." *Applied Economics* 45.24 (2013): 3495-511.

[10] Bianchi, Filippo Maria, Maiorino, Enrico, Kampffmeyer, Michael C, Rizzi, Antonello, and Jenssen, Robert. *Recurrent Neural Networks for Short-term Load Forecasting : An Overview and Comparative Analysis*. Cham: Springer, 2017. SpringerBriefs in Computer Science.

[11] Bianchi, Filippo Maria, Maiorino, Enrico, Kampffmeyer, Michael C, Rizzi, Antonello, and Jenssen, Robert. *Recurrent Neural Networks for Short-term Load Forecasting : An Overview and Comparative Analysis*. Cham: Springer, 2017. SpringerBriefs in Computer Science.

[12] Papacharalampous, Georgia, Hristos Tyralis, and Demetris Koutsoyiannis. "Comparison of stochastic and machine learning methods for multi-step ahead forecasting of hydrological processes." Stochastic environmental research and risk assessment 33.2 (2019): 481-514.

[13] Wang, Yunbo, et al. "Memory In Memory: A Predictive Neural Network for Learning Higher-Order Non-Stationarity from Spatiotemporal Dynamics." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.