Exploring some better methods for Stock Prediction based on deep learning

Li Wenxuan

Abstract - This paper is using Apple stock historical data to complete stock price forecasts, but the traditional methods are not good enough, so we try some new methods and models to see whether they can perform better. We built up **ARIMA** and **GRU** models based on Apple's historical stock price data. In addition, we used the S&P 500 Index Market data as auxiliary information to construct the **ResCNN+LSTM** model as our innovation and attempt. According to the evaluation result, ARIMA has the best performance, but the applicability is poor. When predicting new dates, the model needs to be rebuilt and retrained. The performance of GRU is slightly worse than that of ARIMA, but its applicability is stronger. When predicting a new date, you can use the model directly to make predictions. ResCNN+LSTM performed the worst, but we think our thinking is correct. After our analysis, we believe that we have misselected auxiliary information. In the future work, we will try to find the data of companies and enterprises that are more closely related to Apple as auxiliary information to assist prediction, which may make our assumptions become reality.

Keywords: ***Rolling window, ARIMA, GRU, ResCNN, LSTM***

1 Introduction

The stock market is a place where the stocks of listed companies are traded. The volatility of the stock market makes it an area that requires a lot of analysis of old data. In this field, there are many algorithms that can help us predict stock prices. These include traditional linear models: linear regression, ARMA, ARIMA, ARCH, and GARCH. These methods have been around for a long time, but they still occupy an indispensable position in the field of price forecasting due to their applicability and accuracy. With the continuous development of technology, machine learning and deep learning have become hot spots in many fields, and the stock market is no exception. More and more predecessors are applying machine learning models: Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory Neural Network (LSTM) and Gated Recurrent Unit (GRU) to this field with good results.

Predicting stock prices using historical stock data is not a new topic. Our purpose is to try to use new models and methods to complete the prediction of stock prices for better performance. Our thought process is as follows:

1. We first try the ARIMA model. According to our survey results, it will perform well, but its biggest problem is poor usability. When we make predictions for a new day, we need to rebuild and train the model.

2. We try to improve the applicability of the model while keeping its performance as constant as possible. We choose GRU model which belongs to RNN.

3. ARIMA has a good performance, and GRU has a strong applicability, which ARIMA does not have. Both have their strengths and weaknesses, but we want to have both. Therefore, we will keep the strong applicability of the model while improving the performance as much as possible. We try to add the data of S&P 500 Index market as auxiliary data and build ResCNN+LSTM neural network model.

This is the overall idea and general result of our project. The detailed modeling process, evaluation model, and conclusions will be described in detail later.

2 Related Work

2.1 Traditional Time Sequential Models

The traditional time sequential models contain **AR**, **MA**, **ARMA**, **ARIMA**, **ARCH** and **GARCH**. The full name of AR model is Auto Regression model, which using the historical data to do the regression predicting the future value. MA (Moving Average) model using the error, ϵ, to do the regression. ARMA is a method combining both AR and MA models, and ARIMA is adding a differential step on the basis of ARMA model. ARCH and GARCH models are used to predict the variance, or we say, the risk of the prediction.

But these traditional sequential models have a serious problem that each time we have new data, we should train for a new model. And in fact, these models are not convincible for the suggestions – every time you predict a new price, as long as your new predictions are not too different from the last data, whatever go up or down, like tossing a coin, the MAE and MSE won't be high, but the prediction is not helpful. We need a constant model to fit all data, and when meet new data can predict directly instead of training a new model.

2.2 Machine Learning Models

In some papers, people use **SVM**, **Random Forest**, **XGBoost** and some other traditional machine learning models to do the auto regression. Although we use machine learning method, the performance still seems to be poor.

2.3 Deep Learning Models

Recent years, as the deep learning getting more and more hot, people use it in the financial area more and more frequently. And the most common models are **LSTM** and **GRU** models, which are models that having memory for the past data and auto forgetting some useless parts of them. But they are still not good enough to fit the new data, so in this paper, we will try to add some extra market information to help predict the new price.

3 Data Preparing

3.1 Data Source

We got Apple Inc. and S&P 500 Index Market stock prices from Yahoo from December 31, 2015 to December 30, 2021. Since the stock market closed on the weekend, there are 1511 non-null data in total. Figures 1 and 2 are visualizations of historical stock price data for the S&P 500 Index Market and Apple, respectively.

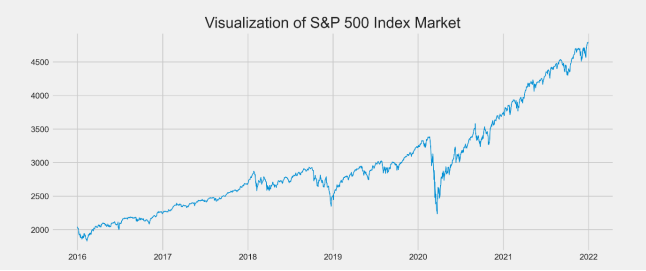

FIGURE 1. Visualization of S&P 500 Index Market

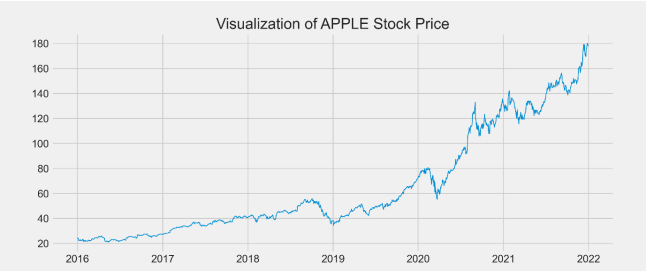

FIGURE 2. Visualization of Apple's stock price

3.2 Data preprocessing

There are no nulls in our dataset and the time range of stock price data for all is from December 31, 2015 to December 30, 2021. So, we don't need to process the data. Beyond that, we believe that all observations in the data, whether outliers or not, should be part of the model fit. The existence of these possible outliers is also one of the reasons why the prediction of stock prices is not accurate enough, and in fact we cannot fully correctly identify which values are outliers. Therefore, we do not choose to denoise the data and remove outliers.

For our purposes, we need to predict the stock price one day in the future based on the company's historical stock price data. Therefore, we need to segment the data according to a certain time span. We used a rolling window algorithm. This is a specified window width and step size, and the data is divided into several subsets by looping. The reason we need to segment the data is because the volatility of stock prices is difficult to analyze. There are too many external factors that can affect stock prices that we cannot fully account for or quantify. This leads to the fact that when we use a period of data to make predictions, we should narrow the scope of the prediction.

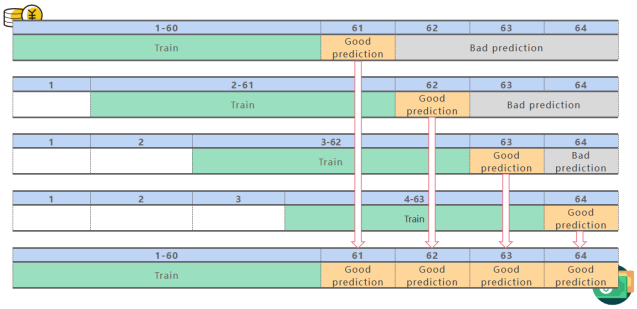

FIGURE 3. The Rolling Window Algorithm Workflow

Table 1 is the first window generated after we split. We can see that the training set time range of window 1 is from December 31, 2015 to March 29, 2016, a total of 60 data. The test set has only one data for March 30, 2016. In addition, the training set of window 2 is from January 3, 2016 to March 30, 2016, and the test set is the data of March 31, 2016.

TABLE I. Example of window with width 60, step size 1

Window 1			
Train Set		Test Set	
2015-12-31	24.199888	2016-3-30	25.325161
2016-01-04	24.220579		
2016-01-05	23.613626		
2016-01-06	23.151514		
⋮	⋮		
2016-03-28	24.315020		
2016-03-29	24.890591		

We have a total of 1511 pieces of data, so we use the

rolling window to segment and obtain a total of 1451 windows. When we use the ARIMA model to complete the prediction task, we will build an ARIMA model for each of these 1451 windows. We use the training set in each window to build and fit the model, and use the trained model to predict the test set. As shown in Figure 4.

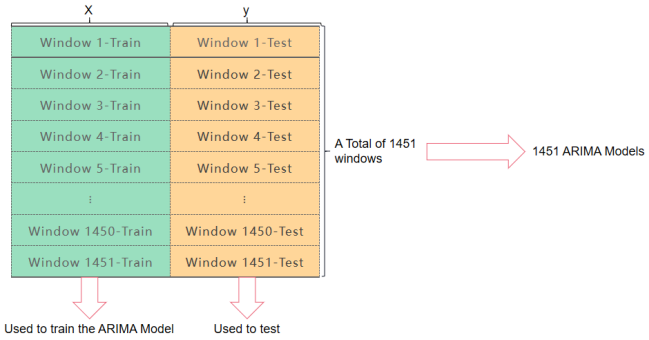

FIGURE 4. Data used for ARIMA

We all know that neural network models require a lot of data when training. Therefore, we set 80% of the 1451 windows as the training set, and the remaining 20% as the test set. As shown in Figure 5.

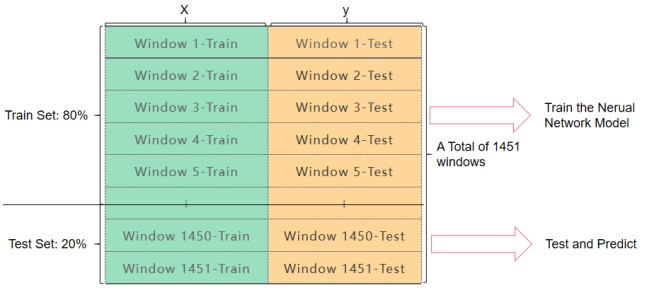

FIGURE 5. Data used for Neural Network Model

4 Methodology

4.1 ARIMA

A time series is a data model that uses plots of successive time intervals to showcase potential correlations. The *ARIMA* model is a time series with three components: an autoregressive component, a differencing component, and a moving average component. The autoregressive component correlates past data with successive data. The differencing component makes the data "stationary" in attempt to filter randomized data from correlated data. The moving average component takes into account past forecast errors to make the model adaptive. The structure diagram of the *ARIMA* model is as follows:

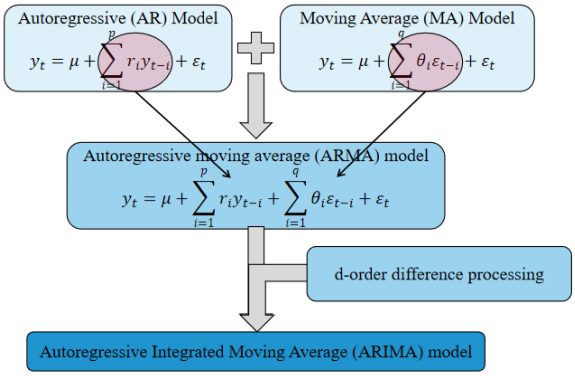

FIGURE 6. The structure diagram of ARIMA

While most *ARIMA* trading strategies are combined with additional linear regression and generalized autoregressive conditional heteroskedasticity (*GARCH*) models, this part only focuses on using *ARIMA* without any external regressors. So, the basic mathematical formula of the *ARIMA* model is:

$$y_t = \mu + \sum_{i=1}^{p} \gamma_i y_{t-1} + \sum_{i=1}^{p} \theta_i \varepsilon_{t-i} + \varepsilon_t \tag{1}$$

In the above formula:

- p: The order of the autoregressive (AR) model
- q: The order of the autoregressive process
- y_t: The current value
- μ: The constant term
- γ_i: The autocorrelation coefficient
- ϵ_t: The error
- θ_i: The moving average coefficient

We can see that the differencing component does not appear in the above formula. This does not mean that this component has no effect on the entire *ARIMA* model. The differencing component is reflected in the d-order difference processing of the data before building the model. So an *ARIMA*(*p*,*d*,*q*) model primarily deals with these three parameters respectively.

4.2 GRU

Recurrent Neural Network is extremely effective for processing time series. The Recurrent Neural Network can mine time series information in processed data. Based on this property, Recurrent Neural Network shine on problems like stock forecasting.

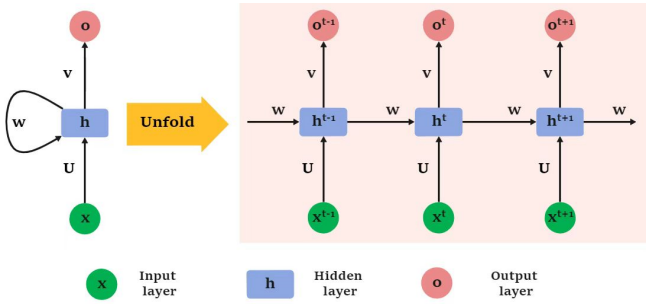

FIGURE 7. Basic Structure of RNN

Basic Recurrent Neural Network consists of 4 parts, which are "Input layer", "Hidden layer", "Recurrent layer" and "Output layer", as shown in Figure 7 left part. Expect for the Recurrent layer, the remaining three structures are not much different from ordinary neural networks. If we talk about the structural expansion analysis of the recurrent layer, we can understand why the recurrent neural network can effectively process sequence information from the data.

In Recurrent layer, the information at time t not only consider the input x_t, but also consider the information form S_{t-1}. So, $S_t = f(U*X_t+W*S_{t-1})$, here X_t is the adj close price at time t of stock and W is the weight matrix between each time point. So now we can understand why Recurrent Neural Network can handle the time series problem, it can remember information at any time t, and each hidden layer not only decided by input layer at time t, but also decided by hidden layer at time $t-1$.

GRU (Gate Recurrent Unit) is one of RNN's. And it is the same as LSTM (Long-Short Term Memory) function. Both these two algorithms solve the problems like long-term memory and gradients in backpropagation. Although these two algorithms have similar results, GRU is computationally cheaper.

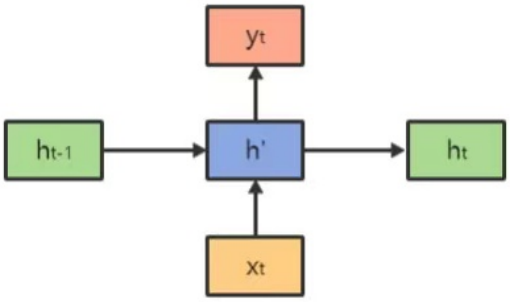

FIGURE 8. Basic structure of GRU (external)

The structure of input and output are the same as common RNN. Combine with x^t and h^{t-1}, GRU will get y^t and it transfer it to the next hidden state h^t. And here is the internal structure of GRU, as shown in Figure 9.

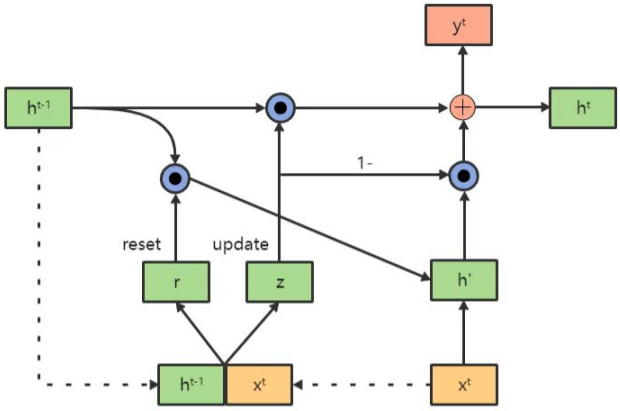

FIGURE 9. The internal structure of GRU

GRU has two main processes. One is resetting memory; another one is updating memory. GRU obtains two gated states through h^{t-1} and x^t. Here r is the reset gate, and the z is the update gate. Meanwhile GRU control gate states between [0,1] through sigmoid function. Though reset gate, we get the data after reset $h^{t-1'}=h^{t-1}.*r$. Next combine $h^{t-1'}$ and x^t by tanh method to get $h^{'}$. In addition, we can update memory after resetting memory, updating formula is that $h^t = (1-z).*h^{'}+z.*h^{t-1}$. If the z-gating signal approaches closely to 1, it means GRU remember much more information, otherwise forgets more information. In the algorithm, reset gate help model capture short-term memory and update gate can help reserve long-term memory. That's why GRU can solve vanishing gradient problem and exploding gradient. Besides, GRU obtains z-gating signal to forget and remember at the same time, so GRU has computationally cheaper. The architecture of our GRU model as shown as Figure 10.

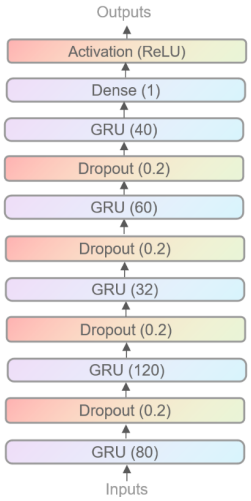

FIGURE 10. The architecture of GRU Model

Our GRU model has 11 layers expect input layer and output layer. And the number in GRU layer are the number of GRU cell. And each GRU layer is followed by a Dropout layer. Dropout layer reduce the risk of overfitting during

training. Through improving the randomness of the results by randomly discarding some weight information, thereby increasing the generalization ability of the model. The Dropout layer in the model is set to drop 20% of the random weight parameters.

4.3 ResCNN+LSTM

In the convolutional architecture, there are some parameter concepts like kernel size, strides, padding and filters. The kernel size means the size of the kernel; padding means add some rows to expand the whole picture; filters mean how many kernels we used in this layer; and the stride means steps, which is how many hops you dump each time you move. As the following example, our original picture is an 8×8 image, and we having padding equals 1, strides equal to 1 and 2, kernel size equals 3×3.

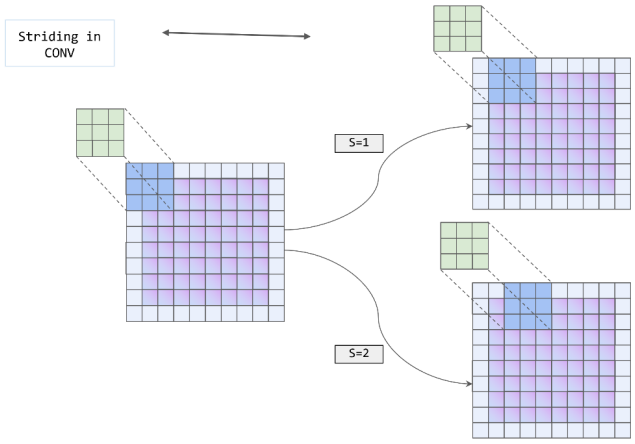

FIGURE 11. The Convolutional Architecture

And here is our architecture. Since our network is quite deep, so we design a residual block architecture to prevent the gradient vanished or exploded.

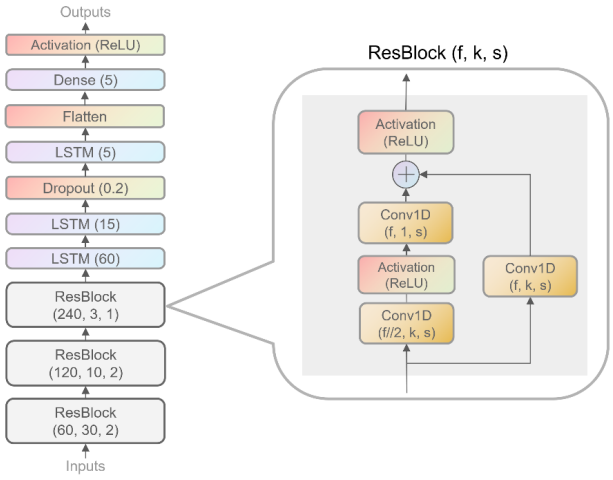

FIGURE 12. The Architecture of ResCNN+LSTM

And the Dense layer here means linear transforming with formula like:

$$f(x) = w^T x + b \tag{2}$$

The LSTM(n) is a layer of Long-short term memory network architecture with n neurons, which is a specific form of Recurrent Neural Network (RNN). The LSTM model is based on the RNN model, which makes the RNN really effective to use the long-range temporal information with 3 types of gates, which also solves the gradient dispersion problem in RNN.

LSTM model uses memory cells, like long-term and short-term memory modules, instead of using traditional neurons. Each memory cells have input gates, forgetting gates, memory units and output gates. The detailed structure of it is shown as follows:

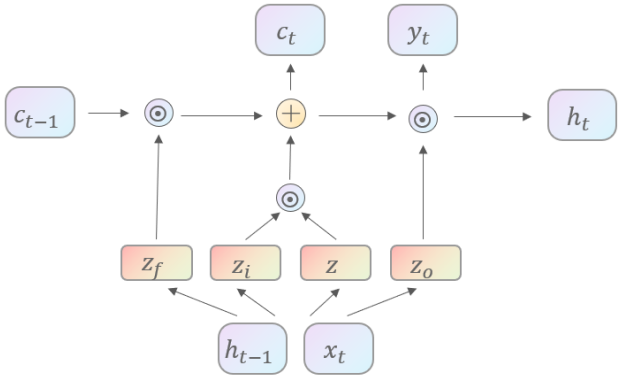

FIGURE 13. The detailed structure of LSTM

1. Input gate: It's used to memorize some information of the current input information with layers whose activating function are sigmoid and tanh. The sigmoid layer determines which value we will update, and the tanh layer is used to create new candidate values vector $\widetilde{C_t}$

$$i_t = \sigma(W_i \bullet [h_{t-1}, x_t] + b_i) \tag{3}$$

$$\widetilde{C_t} = tanh(W_C \bullet [h_{t-1}, x_t] + b_C) \tag{4}$$

2. Forgotten gate: It's utilized to selectively forget some information of the past. The gate will output a number between 0 to 1 to discard the degree of information where 1 means "completely reserved" and 0 means "completely discarded"

$$f_t = \sigma(W_f \bullet [h_{t-1}, x_t] + b_f) \tag{5}$$

3. Memory unit: It's like a bridge connecting the past and present information and combines these memories.

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t} \tag{6}$$

4. Output gate: It's used for selectively outputting some information with sigmoid layer and tanh layer where sigmoid layer determines which values can be output,

and the tanh layer shrinks the value to be ranging of -1 and 1.

$$o_t = \sigma(W_o \bullet [h_{t-1}, x_t] + b_o) \tag{7}$$

$$h_t = o_t \bullet tanh(C_t) \tag{8}$$

Among them, h_t is t-time hidden state, W_i, W_C, W_f, W_o are the weight metrics, and b_i, b_C, b_f, b_o are bias for the corresponding units. These are all model training parameters. (Notice that here $z = W \bullet [h_{t-1}, x_t]$)

5 Experiment

5.1 ARIMA

Building a complete ARIMA(p,d,q) model requires the following steps. We will use window1 as an example to show the complete process.

1. The Stationarity Test

The time series data required by ARIMA must satisfy stationarity, so we need to test the stationarity of the data. The current mainstream hypothesis testing method for stationarity is the unit root test, which tests whether there is a unit root in the series. We take the Augmented Dickey-Fuller (ADF) Test, which is one of the most commonly used unit root tests.

```
The stationarity test of the original data:

   Augmented Dickey-Fuller Results
═══════════════════════════════════
Test Statistic                -1.431
P-value                        0.567
Lags                               0
───────────────────────────────────

Trend: Constant
Critical Values: -3.55 (1%), -2.91 (5%), -2.59 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.

──────────────────────────────────────────────────────────

The stationarity test of the data after first-order difference:
   Augmented Dickey-Fuller Results
═══════════════════════════════════
Test Statistic                -6.066
P-value                        0.000
Lags                               1
───────────────────────────────────

Trend: Constant
Critical Values: -3.55 (1%), -2.91 (5%), -2.59 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
```

FIGURE 14. The stationarity Test of Window 1

According to the results shown in Figure 14, the p-value obtained when we perform ADF test on the original series is 0.567, which is larger than 0.05. This means that we cannot reject the null hypothesis (the unit root exists and the time series is non-stationary), which suggests that we need to do something with the original series. We performed the ADF test again after first-order differencing of the original series. The resulting p-value was less than 0.05, and based on the comparison of test statistics and Critical Values, we could reject the null hypothesis with 99% confidence. This means that the first differencing time series passes the stationarity test and we can set the differencing component (d) to 1.

2. White Noise Test

White noise testing is also known as randomness testing. ARIMA not only requires time series data to satisfy stationarity, but also to satisfy non-randomness. If the time series is white noise, then the time series is completely random, which means that past behavior has no influence on future development, so there is no need for further analysis. Therefore we need to perform a white noise test on the time series. Here, we use the Box-Pierce test together with the Ljung-Box test to complete this test. Because the original series did not pass the stationarity test, we will use the first-order difference series for the white noise test.

TABLE II. Result of Box-Pierce and Ljung-Box test

lag	lb-stat	lb-pvalue	bp-stat	bp-pvalue
1	0.892791	0.344722	0.848883	0.356869
2	1.237820	0.538531	1.171287	0.556747
⋮	⋮	⋮	⋮	⋮
27	12.902512	0.989855	9.746193	0.999051
28	12.988328	0.992952	9.789805	0.999430

According to our results, the p-values of both the Box-Pierce test and the Ljung-Box test are greater than 0.05, which means that the first-order difference series does not pass the test. But that doesn't fully explain that the first-order difference series failed the white noise test. We can determine this by looking at the ACF and PACF of the time series.

3. Draw ACF and PACF plots

ACF is a complete autocorrelation function that gives us the autocorrelation value for any series with lag values. In simple terms, it describes the degree of correlation between the current value of the series and its past values. The PACF is a partial autocorrelation function or a partial autocorrelation function. Basically, instead of finding the correlation of a lag like the ACF with the current, it finds the correlation of the residual (which remains after removing the effects already explained by the previous lags) with the value of the next lag.

Here, we draw the ACF and PACF plots of the original series, the first-order difference-processed series, and the second-order difference-processed series.

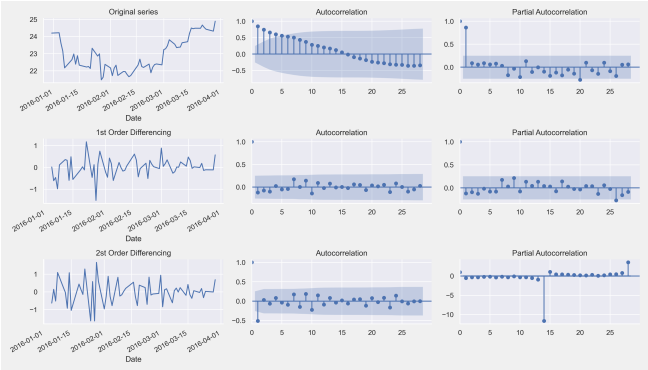

FIGURE 15. ACF and PACF of Time Series

Based on the results of our stationarity test, we set the differencing component to 1. In Figure 15, we can see that the first-order difference processing makes the series much more stationary compared to the distribution of the original series. Although the results of the second-order difference are not much different from those of the first-order difference, there is a gap in the effect of the PACF map. In addition, we can see that in the ACF and PACF diagrams of the first-order difference sequence, there are several function values corresponding to lag that approach or exceed the gray area. We concluded that the first-order difference sequence passed the white noise test. So we finally settled on the differencing component. By analyzing the ACF and PACF plots of the first-differenced series, we set both p and q to 0.

4. Build an ARIMA model and fit it with the data

So far, we have found the ARIMA parameters suitable for the data subset window1. We build an ARIMA (0,1,0) model and fit it. We visualize the fitting results and residuals.

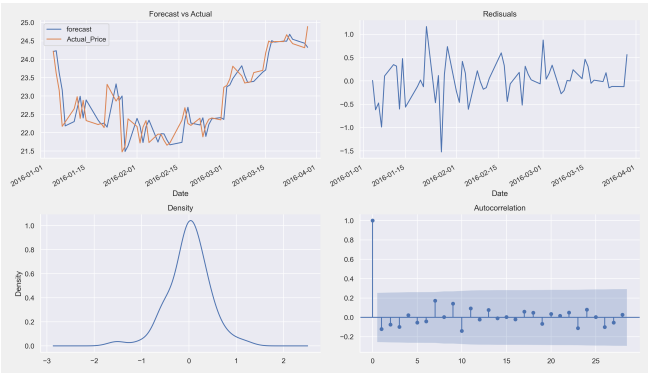

FIGURE 16. The fit result of model and Residuals distribution

According to Figure 16, we can see that (1) there is a certain lag between the data fitted by ARIMA and the real data. (2) The distribution of errors basically conforms to the normal distribution. (3) The ACF diagram of the residuals shows that the residuals may be white noise, so we performed a sequential white noise test on the residuals alone. According to the test results, we confirmed that the residuals are white noise, which means that there is no unextracted information in the residuals and the ARIMA model fits well.

TABLE III. Result of Residuals Box-Pierce and Ljung-Box test

lag	lb-stat	lb-pvalue	bp-stat	bp-pvalue
1	0.892791	0.344722	0.848883	0.356869
2	1.237820	0.538531	1.171287	0.556747
⋮	⋮	⋮	⋮	⋮
26	12.586499	0.987353	9.580416	0.998603
27	12.902512	0.989855	9.746193	0.999051

5. Evaluation the model

From the residuals we get, we can calculate the statistics used to evaluate the regression model, such as mean squared error (MSE), mean absolute error (MAE), and root mean squared error (RMSE).

TABLE IV. Basic Statistics for prediction result of window 1

Statistics	Value
MSE	0.1842
RMSE	0.42929
MAE	0.3042

Through the various residual-related statistic values used to evaluate the regression model in Table 4, we can see that these three statistic values are relatively small. In addition, the residual is determined to be white noise after the white noise test, which means that there is no unextracted time series information in the residual series. Therefore, we believe that the ARIMA(0, 1, 0) model is a good time series model for Window 1.

6. Predict using the ARIMA model

Finally, we found an ARIMA model suitable for Window 1 training data, and its parameters are 0, 1, 0. And after our evaluation, the fitting effect of this model is good, and it can be used to complete the prediction of Window 1 test data.

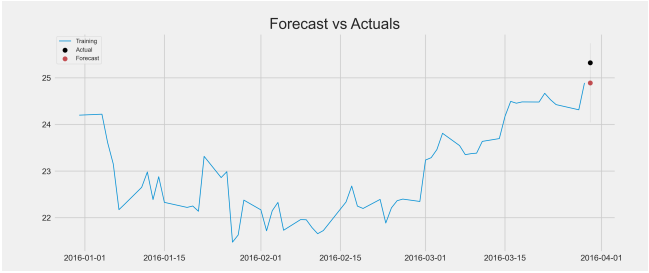

FIGURE 17. Prediction result of ARIMA based on Window 1

Through the observation of the above figure 17, the red points represent the results predicted by the ARIMA(0,1,0) model, and the black points represent the real values. Although there is a certain error between the two points, the black point still exists within the 95% confidence interval.

Therefore, we believe that the results of this prediction are acceptable.

Now we have completely completed the whole process of establishing the ARIMA model for Window 1 data. But we have a total of 1451 data subsets of the same size as Window 1. If we do the above complete operation for every subset of data, it would be a huge project. In order to speed up our modeling, we choose to use the auto arima function in the pmdarima package to complete the operation of building an ARIMA model for each data subset. The auto arima function is to find the ARIMA(p,d,q) model with the smallest AIC value by performing grid search within the specified parameter (p,d,q) value range. This can save us a lot of workloads, so we choose to use auto arima to complete the ARIMA modeling of the 1451 data subsets.

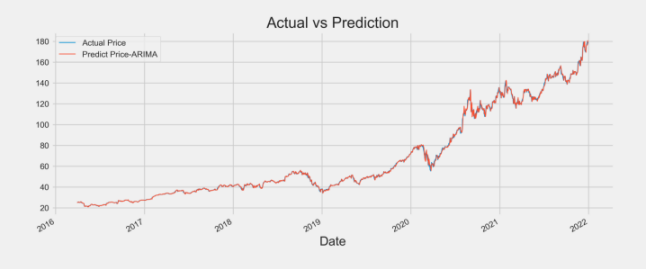

FIGURE 18. Prediction Result of Whole Test Set

By observation, we can see that the distribution of predicted values is very similar to the distribution of true values. Although there are many errors, these errors cannot be completely avoided. In order to measure the accuracy of these prediction results, we still use MSE, RMSE and MAE to measure the prediction effect.

TABLE V. Basic Statistics for the ARIMA Model

Statistics	Value
MSE	2.5191
RMSE	1.5872
MAE	0.9407

From the data in Table 5, it can be seen that using ARIMA to predict Apple's stock price is very good, and the residuals are within the expected range.

5.2 GRU

After constructing the network, we first randomly picked 80 percent of the data for training and 20 percent for testing. In our GRU model, we trained it with MSE loss and chose Adam as the optimizer. we set learning rate as 0.001 and the batch size as 64. During learning, we set 100 epochs to make sure error can be convergence. The training loss is shown as Figure 19. And the prediction result is shown as Figure 20.

FIGURE 19. The relationship between epoch and loss in GRU

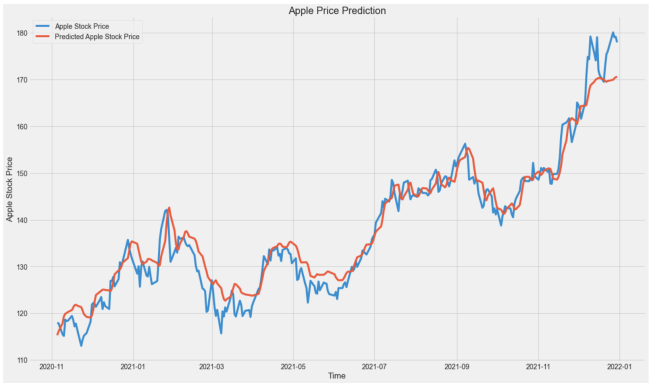

FIGURE 20. Prediction Result of GRU

5.3 ResCNN+LSTM

After constructing the network, we first randomly picked 80 percent of the data for training and 20 percent for testing. Then we trained it with MAE loss and chose Adam as our optimizer. At the very beginning, we set the learning rate to 0.01, batch size to 512 and trained for 30 epochs, then changed the learning rate to 0.001 and continued trained it for 50 epochs. Finally, we changed the learning rate to 0.0001 for fine tuning with 50 epochs. The training loss is shown as Figure 21. And the prediction result is shown as Figure 22.

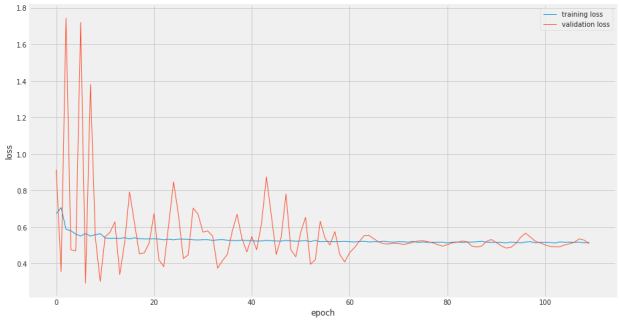

FIGURE 21. The relationship between epoch and loss in ResCNN+LSTM

FIGURE 22. Prediction result of ResCNN+LSTM

6 Result

6.1 Evaluation

Due to the difference of methods and models, the segmented data are processed differently in the process of building the model. In the ARIMA method, we make predictions on the test set in all 1451 windows. Therefore, the prediction range of the ARIMA model is from March 30, 2016 to December 30, 2021, with a total of 1451 prediction results. The two neural network models of GRU and ResCNN+LSTM do not predict all window test sets because they need a part of the data to train the model. We use 1161 windows as the training set of the model, 80% of all; and the remaining 290 windows as the test set. Therefore, the prediction range of the remaining two models is from November 5, 2020 to December 30, 2021, with a total of 290 prediction results.

In order to fairly evaluate the effects of the three models, we control the range of prediction results within the same interval, that is, from November 5, 2020 to December 30, 2021, a total of 290 days. We visualize the prediction results of the three models, as shown in Figure 23.

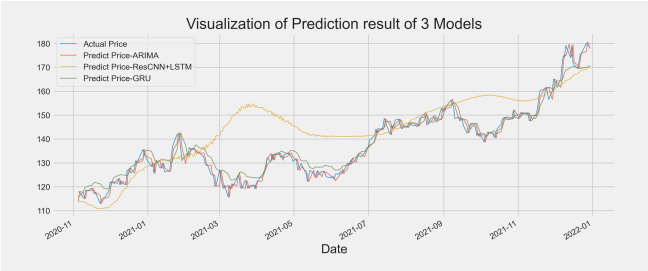

FIGURE 23. The prediction result of 3 Models and actual value

Through the image we can see the approximate fitting effect of the three models. The predicted value image made by ARIMA and GRU is close to the real value image, while the prediction result of the ResCNN+LSTM model is quite different from the real value. We calculated the MSE, RMSE and MAE of the three models respectively.

TABLE VI. Statistics for evaluate 3 models

Statistics	ARIMA	GRU	ResCNN+LSTM
MSE	4.9534	12.4826	69.1050
RMSE	2.2256	3.5330	8.3129
MAE	1.7121	2.7383	74.9526

6.2 Conclusion

Based on our evaluation of the predictions of the three models, we conclude as follows:

TABLE VII. Evaluation result of 3 models

Evaluation Items	ARIMA	GRU	ResCNN+LSTM
Performance	Best	Good	Not Good
Applicability	No	Yes	Yes
Single Model?	No	Yes	Yes

The ARIMA model has the best performance. This model produces the most accurate predictions with the smallest error. But since the ARIMA model is not composed of a single model, it is a composite model composed of many ARIMA models. Therefore, the generalization ability of ARIMA is poor. In order to avoid the generalization ability and applicability of the model, we use the GRU model. Compared with ARIMA, its performance is slightly worse, but overall the prediction results are more accurate. The most important thing is that the GRU model has the generalization ability that ARIMA does not have. Because of the applicability of the GRU model, when we predict a new window again, we do not need to use the training set in the window to retrain the model, but can directly use the model to complete the prediction according to the training set part. In addition, despite the poor performance of ResCNN+LSTM, we believe that our thinking is not wrong. We will do more further work based on this idea.

7 Future Work

Through our thinking and summary, we believe that the reason for the poor performance of the ResCNN+LSTM model is the use of S&P 500 Index Market data as auxiliary information. We all know that there are many factors that affect the price of a stock, and these effects may be positively correlated or negatively correlated. But we don't think the relationship between the S&P 500 data and Apple's stock price is purely positive or negative. Their relationship is also affected by many factors, so at different points in time, their relationship may change. Therefore, we will replace the auxiliary information and find relevant data of companies and enterprises that are more closely related to Apple as auxiliary information. In doing so, the ResCNN+LSTM model may perform better than the GRU.

8 Reference

1. Liu, F., Li, X., Wang, L. (2019, December). Exploring Cluster Stocks based on deep learning for Stock Prediction. In 2019 12th International Symposium on Computational Intelligence and Design (ISCID). (Vol. 2, pp. 107-110). IEEE.
2. Ariyo, A. A., Adewumi, A. O., Ayo, C. K. (2014, March). Stock price prediction using the ARIMA model. In 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation. (pp. 106-112). IEEE.
3. Sethia, A., Raut, P. (2019). Application of LSTM, GRU and ICA for stock price prediction. In Information and communication technology for intelligent systems (pp. 479-487). Springer, Singapore.
4. Sharma, N., Juneja, A. (2017, April). Combining of random forest estimates using LSboost for stock market index prediction. In 2017 2nd International conference for convergence in technology (I2CT). (pp. 1199-1202). IEEE.
5. Schöneburg, E. (1990). Stock price prediction using neural networks: A project report. Neurocomputing, 2(1), 17-27.
6. Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., Soman, K. P. (2017, September). Stock price prediction using LSTM, RNN and CNN-sliding window model. In 2017 international conference on advances in computing, communications and informatics (icacci) (pp. 1643-1647). IEEE.