Time Series Prediction on Sales Data

Juanxi Li
Hongliang Shi

1 Introduction

Sales prediction is a typical time series forecasting problem. According to Box, George EP, et al, “A time series is a sequence of observations taken sequentially in time [1].” Thus, time series forecasting would be making predictions about the future of such observations and in our case, the total number of items sold by a company every day. While predicting the future sounds intriguing, time series forecasting attracts less attention in data mining than classifications or regressions. This is partly because of the difficulties of these problems, especially when dealing with multivariate data.

1.1 Problem Descriptions

In this project, we use the sales data from January 2013 to September 2015 predict the sales of every day in October 2015. The data, collected from a Kaggle competition [2], contain information about the amount of sold products, the price of each product, and the shops where the products are sold every day. These data are provided by a Russian software company called 1C Company. However, we are not doing the exact same task as the proposed challenge. While the competition asks participants to predict the total number of items sold in the last month, we challenge ourselves to figure out the sale of each single day in that month. This prediction is possible because the data we collected have a certain trend and seasonality, which we will explain in detail in the next section. In addition, we also assume that the sales of one day are correlated with the sales of the previous several days. In other words, we assume that the statistical properties of a time series in the future is the same as that in the past.

1.2 Coding Contributions

We used multiple Python packages in this project, including Keras, Tensorflow, Pandas, Numpy, sklearn, matplotlib, StatsModel, seaborn, graphviz, and googletrans. While model construction in sklearn and StatsModel does not need much coding, building the long short-term memory (LSTM) model requires a lot of customized codes. In addition, a large amount of our coding invested in the preprocessing of data and feature selection, such as date formats converting, stationarity analysis, detrending, as well as calculating moving average, exponentially weighted moving average.

1.3 Group Member Contributions

Juanxi Li: Implemented the LSTM models, analyzed the data, and wrote part of the report.
Hongliang Shi: Implemented the ARIMA model, analyzed the data, and wrote part of the report.

2 Data Set

In this section, we will discuss the shape, the stationarity, and the seasonality of the original data.

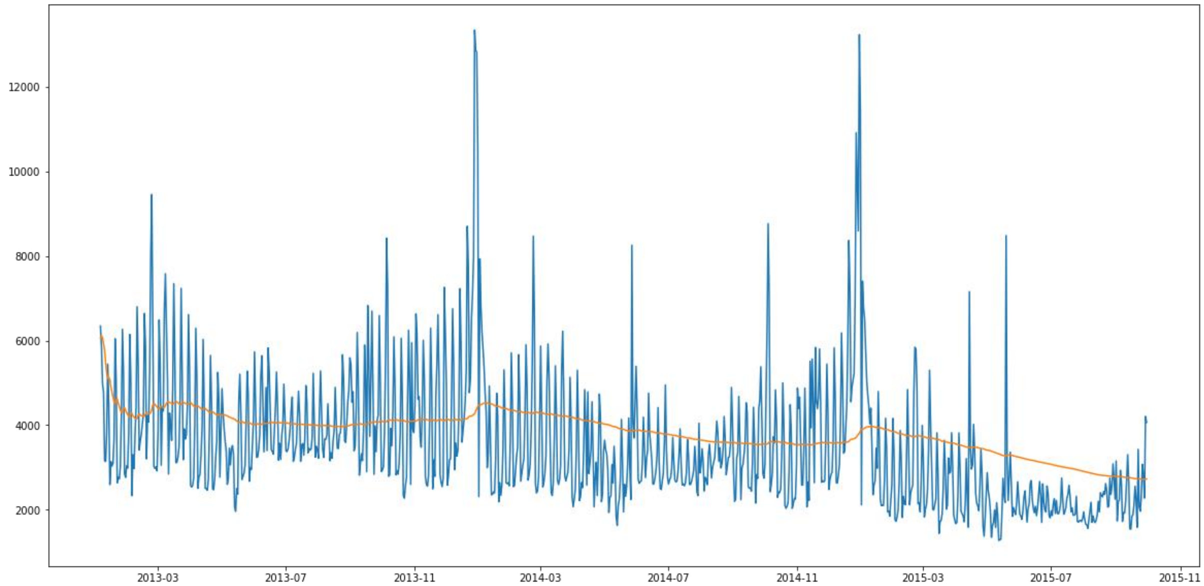


Figure 1: The blue line represents the original data while the yellow line represents the 365-day exponentially weighted moving average (EWMA).

There are apparently two peaks in Figure 1, both of which locate in the period of December and January. This coincides with the consuming habit that people would like to go shopping during the Christmas and New Year sales period. Apart from this peak, there are also three lower peaks in 2014 and two in 2013. In 2013, the first lower peak is in February and the second one is in October. The similar things happened in 2014 except that there is an extra peak in June. However, this pattern is less obvious in 2015, which increase the difficulty of our task.

In addition, there is a trend of decrease in sales over the years as we can see in the yellow line of Figure 1. The EWMA is calculated as follows:

$$EWMA(t) = \lambda Y_t + (1 - \lambda)EWMA(t - 1) \text{ for } t > 0,$$

where Y_t is the observation at time t and $0 < \lambda < 1$ is a parameter that determines how far in the past EWMA can memorize. It is usually easier to think of λ as a function of *span*, which corresponds to the common expression of "N-day EW moving average". The relation between λ and *span* is as follows:

$$\lambda = 2/(s + 1), \text{ for } s \geq 1.$$

The EWMA with a large *span* can very well capture the trend of the data. Considering that the economy in Russia is has not been so well in recent years, it is reasonable that the sales of this Russian company would also slightly decrease over the years.

Moreover, the distance between two local maximum is roughly the same, about 2 days. This result is calculated through Fourier transformation, which can capture the frequency of sequential data. We also noticed that the fluctuation of data becomes smaller after May in 2015, compared to the same period of time in the previous two years.

2.1 Stationarity

One of the most important characteristics of a time series is stationarity. A time series can be considered as stationary if its statistical properties don't change over time [3]. Our prediction works on the assumption that the time series is stationary, so we need to test its stationarity. One way to do this is to draw moving averages (MA) and moving variances of our data to see how they change over time but this method is too intuitive, so we use the Dickey-Fuller Test to check the stationarity. If the test statistics is bigger than the critical value, we can reject the assumption that the data is stationary. According to the test statistics shown in Table 1, it's clear that the original data is stationary.

Test Statistic	*P*-value	Lags Used	Critical Value (1%)	Critical Value (5%)	Critical Value (10%)
-4.0456	0.0012	21	-3.4368	-2.8644	-2.5683

Table 1: The results of the Dickey-Fuller Test on the original data.

2.2 Detrending

While the data is stationary, it is still important to remove the trend of the data in a prediction task because it can enlarge other patterns of the data, such seasonal and cyclical components. There are many ways to detrend a time series and we experimented with three of them, including log-transform, MA and weighted moving average (WMA). The log-transform simply means taking the log of original data, which penalizes higher values more than smaller values. The MA is a calculation to analyze data points by creating series of averages of different subsets of consecutive points in the whole data set [4]. Moreover, the WMA assigns weights to all the previous values with a decay factor, which means it gives higher weights to the more recent points.

Figure 2: The figure on the left shows the rolling mean (green line) and rolling standard deviation (red line) of the log-transformed data (blue line). The figure in the middle shows the rolling mean (green line) and rolling standard deviation (red line) of the MA (blue line). In addition, the figure on the right shows the rolling mean (green line) and rolling standard deviation (red line) of the WMA (blue line).

At the first glance, Figure 2 shows that the MA did the best at detrending the data, but we still need more statistics to validate this observation. Thus, we take the Dickey-Fuller Test again on the three series.

Method	Test Statistics	Lags Used	Critical Value (1%)	Critical Value (5%)	Critical Value (10%)
Log-transform	-3.1307	21	-3.4368	-2.8644	-2.5683
MA	-8.6252	22	-3.4370	-2.8644	-2.5683
WMA	-4.0462	21	-3.4368	-2.8644	-2.5682

Table 2: The result of the Dickey-Fuller Test on the decomposed data.

As we can see in Table 2, the log-transform method makes the test statistics value bigger, which indicates that the data is less detrended. Meanwhile, the MA and WMA methods reduce the test statistics value and thus could be helpful with our prediction.

3 Data Preprocessing

In this section, we discuss how we process the data before feeding them to the models. This includes scaling and selecting new features.

3.1 Scaling

The data in the series vary from 1274 to 13343, which can be problematic in a prediction task, so we need to scale them into the interval [0,1]. In this way, we can reduce the influence of the larger values in the series. Notice that we developed our scaler only with the training data and then perform the same transformation to the testing data so that we can isolate the testing data during the training process and get a reasonable result of the models.

3.2 Feature Selection

The goal of this project is to utilize the number of the daily sold items to predict the future sales situation and it is hard to make a precise prediction only with one feature. Therefore, we need to collect more information through feature engineering. Feature engineering is the transformation of raw data into features that better represent the underlying problem to the predictive models. First, we took in the the data of the previous five days as features. Then, we converted the date of every record into four attributes: “year”, “month”, “day of a week”, and “day of a month”. For the “day of the week” feature, we need to transform this variable into dummy variables since it only states the categories. In addition, we calculate the mean, median, and standard deviation of the original data. Similarly, we collected the mean, median, and standard deviation of the product prices every day as features. We also considered the EWMA of the previous day.

4 Models

In this section, we discuss the models for predicting the future sales and how we built these models. After experimenting with several models, we will report two of them: the ARIMA and the LSTM.

4.1 ARIMA

The autoregressive integrated moving average (ARIMA) model is a well-known time series prediction model in the field of statistics and econometrics. The ARIMA model has three parameters which are often denoted as (p, d, q). Specifically, p states order of autoregressive (AR) terms, q states the order of MA terms and d states the degree of differencing. It is important to determine the values of this parameters.

We utilize the Bayesian information criterion (BIC) to determine the most appropriate parameters [6]. Low BIC values indicate that the model is suitable for the data. In addition, we provide autocorrelation function graph (ACF) and partial autocorrelation function (PACF) graph to justify our choice of these parameters. ACF can measure the correlation between the time series with a lagged version of itself and PACF measures the same correlation but removes the variations explained by intervening version. We pick the value of p by finding the point where the PACF first goes through the 95% confidence interval. Similarly, we select the value of *q* by picking the point where the ACF first goes through the 95% confidence interval. These graphs are shown in Figure 3-5. As we mentioned previously, we detrended the data with three methods, including log-transform, MA, and WMA. Here, we compare the performance of these methods.

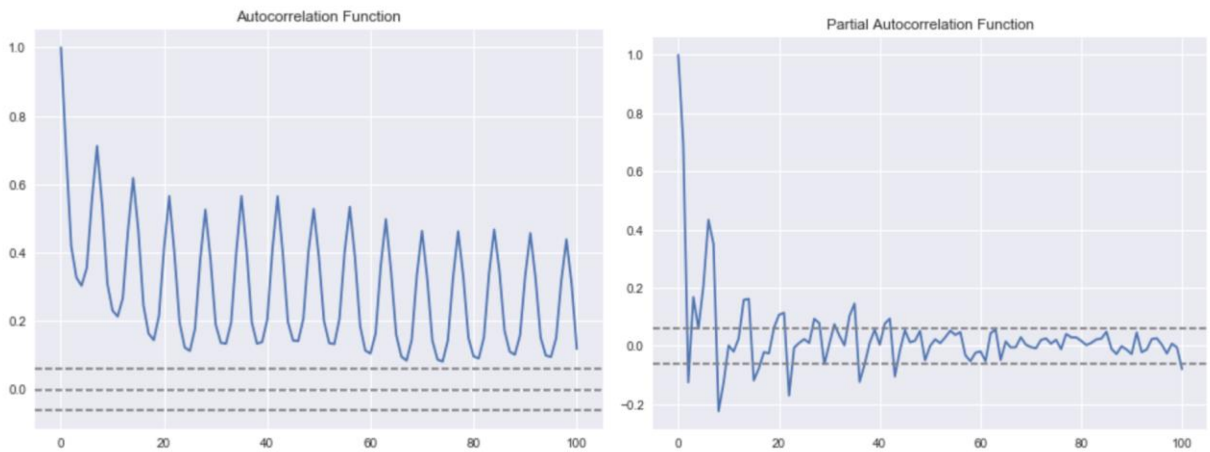


Figure 3. On the left is the ACF graph of the log-transformed data and the PACF graph is on the right.

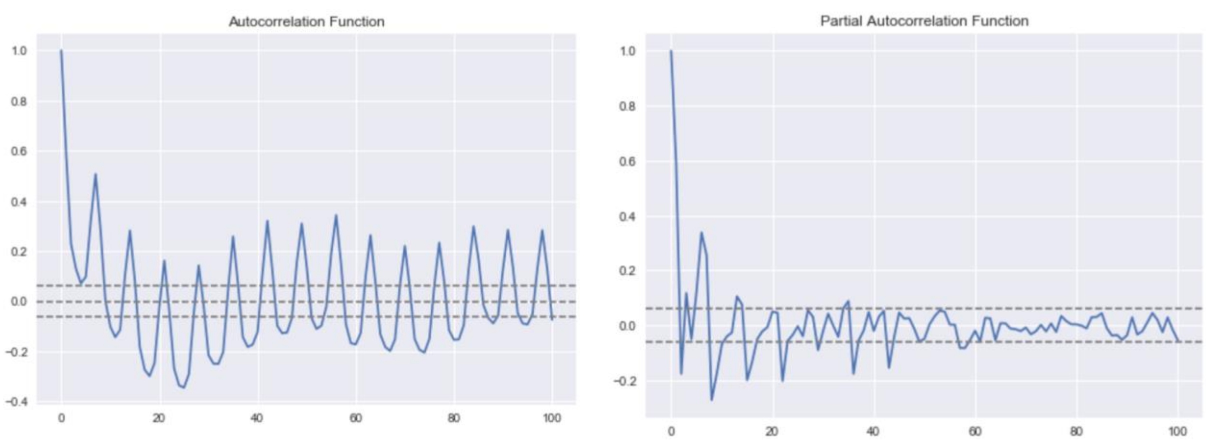


Figure 4. On the left is the ACF graph of the MA data and the PACF graph is on the right.

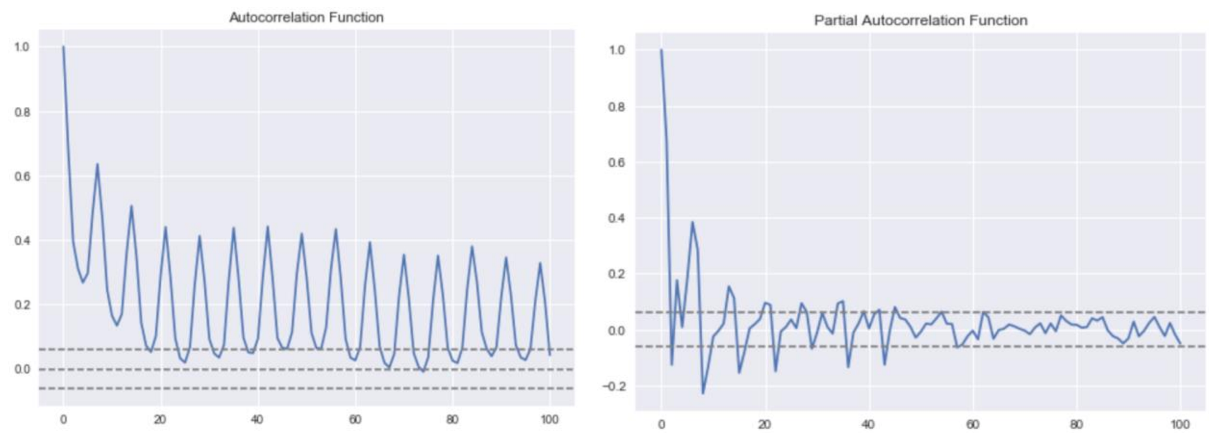


Figure 5. On the left is the ACF graph of the WMA data and the PACF graph is on the right.

The two dotted lines on either side of 0 show the 95% confidence intervals. These can be used to determine the *p* and *q* values. For the log-transformed data, the lag value in the PACF graph first crossed the upper confidence interval when $p = 2$, the lag value in the ACF graph never crossed the upper confidence interval, so we need to consider the BIC information to determine the *q* value. Similarly, we have $p = 3$, $q = 4$ for the MA data and, $p = 3$, $q = 18$ for the WMA data. The values of *d* are determined through multiple trials.

For the log-transformed data, we choose the parameters (2, 0, 8). The results are as follows.

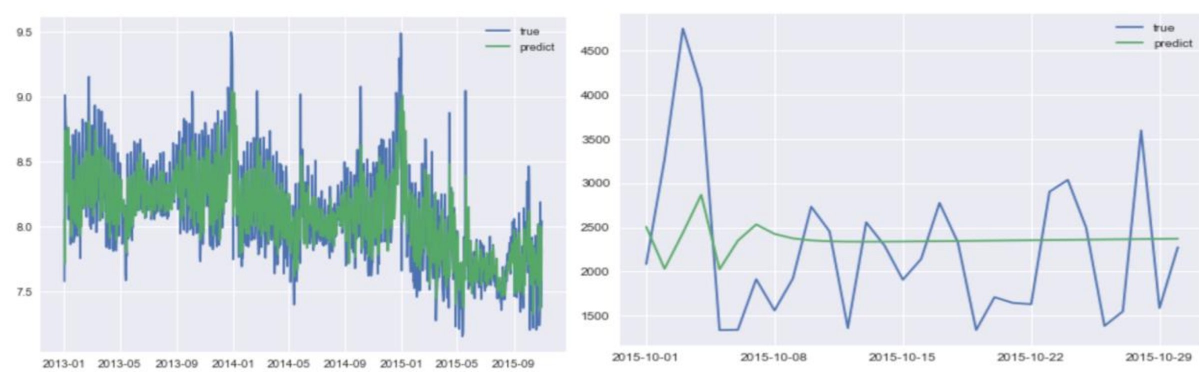


Figure 6: On the left, we have the result of log-transformed data on the training set and the result of the testing set is on the right. In both figures, the blue line represents the true values and the green line represents the predicted values.

We can see that the log-transformed data do not produce a good performance. In fact, all the predicted values are lower than the original values and the predictions bear little resemblance to the true values.

As for MA and WMA, we chose the parameters (3, 0, 4) and (3, 0, 18) respectively.

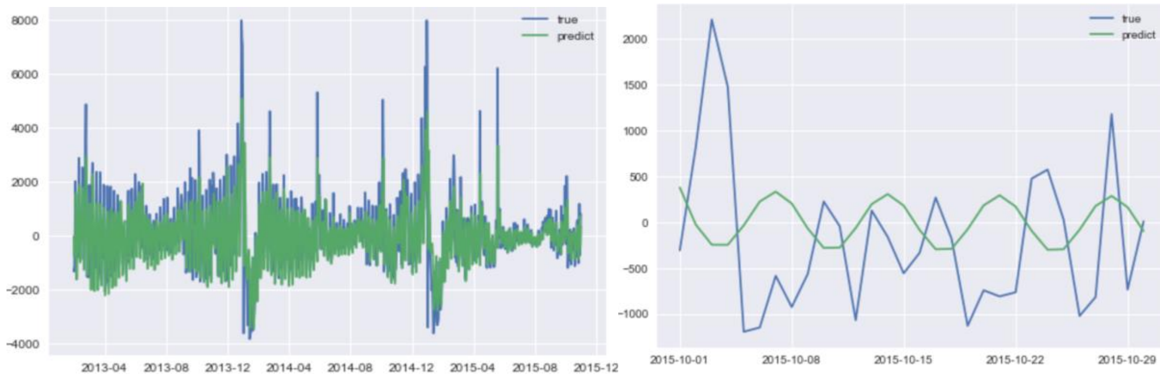


Figure 7: The figure on the left shows the result of MA on the train data. The one on the right shows the result of MA on the testing data. In both figures, the blue line represents the true values and the green line represents the predicted values.

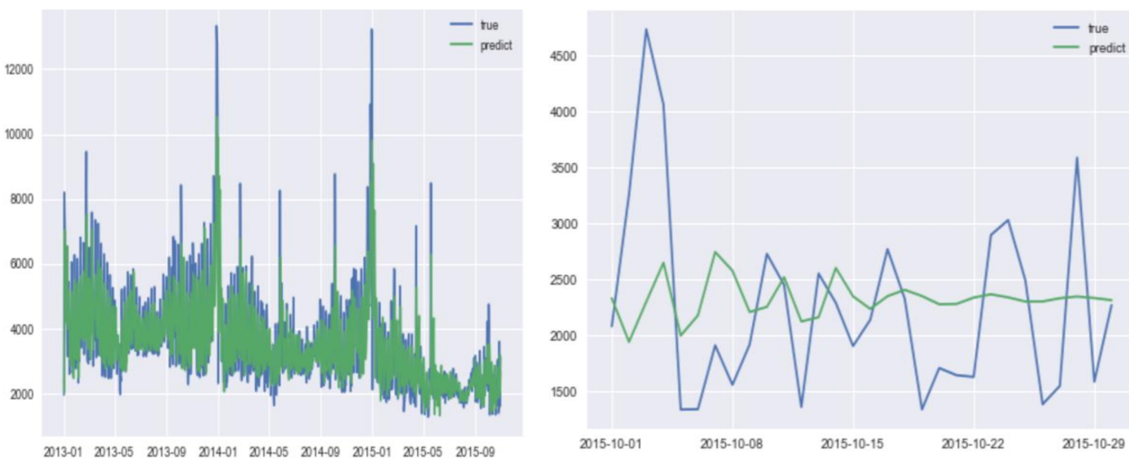
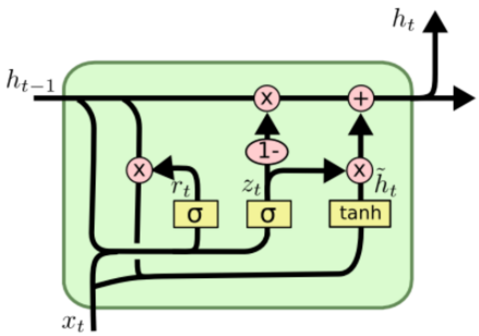


Figure 8: The figure on the left shows the WMA of the training set and the one on the right shows the WMA of the testing set. In both figures, the blue line represents the true values and the green line represents the predicted values.

The MA and WMA both have better performances than the log-transformed data. In the training set, the prediction behaviors are similar to the original data. In testing set, the predictions of MA go up and down almost at the same time as the true value but we can see that the errors are not small. It is frustrated that the prediction of WMA becomes flat in the later period of the testing process. This shows that we did not model the fluctuation very well.

4.2 LSTM

We can treat this problem as a supervised learning problem where the target is the number of products sold each day and the features are those we discussed in the feature selection section. In this way, a multilayer perceptron can handle such a task. In addition to the ARIMA model, we want to explore whether we can apply deep learning methods such as Recurrent Neural Networks(RNN) to this task. We pick RNN because it is very good at handling sequential data such as texts and speeches. The LSTM, a popular type of RNN, is composed of a cell, an input gate, an output gate, and a forget gate as shown in Figure 9 [3].



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 9: An LSTM cell.

The structure of our model is shown in Figure 10. It has four layers. The LSTM layer takes in a sequence with a length of four each time, meaning it looks at the data of the previous four days before making a prediction. The input of each day has a dimension of six, representing the six features used in this model. The dropout layer randomly ignores a proportion of the values coming to it, which helps prevent overfitting. The proportion, called the dropout rate, is a hyper-parameter that can be changed during tuning. After the dropout layer, there is a fully-connected layer which gives only one output as the prediction.

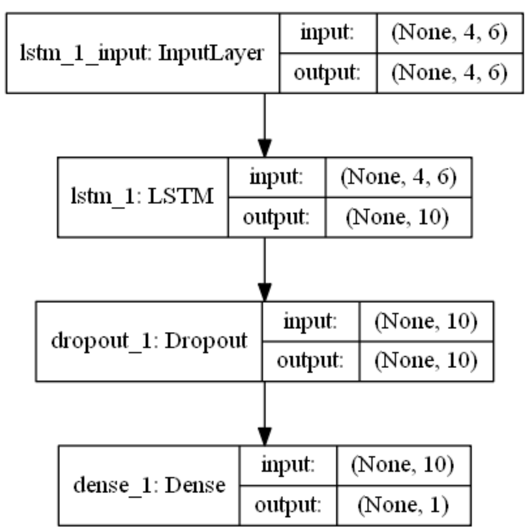


Figure 10: The LSTM model.

During training, we use the MSE as the loss function, setting the learning rate at 0.001, the dropout rate at 0.5, and the number of epochs at 100.

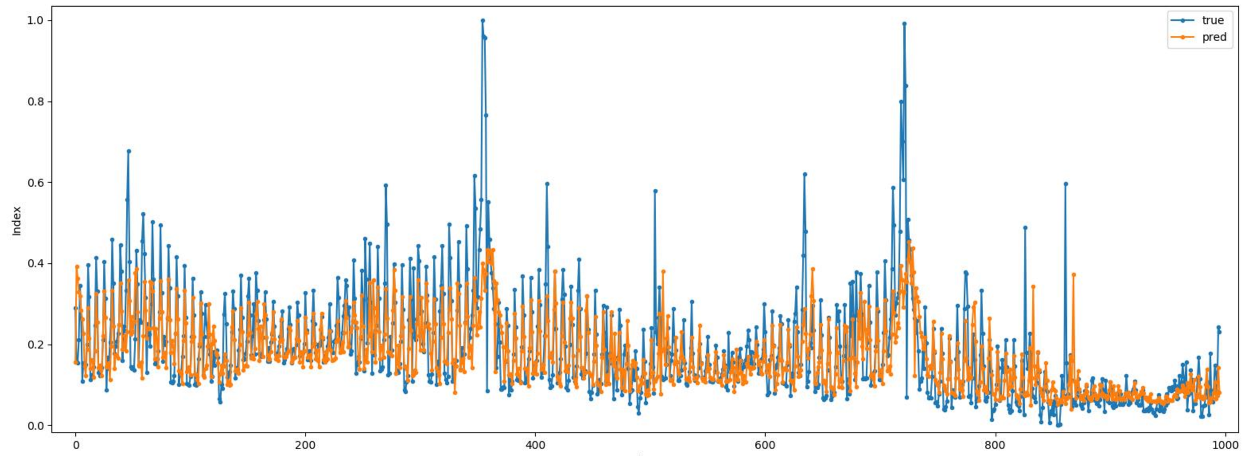


Figure 11: The training results of the LSTM model. The blue line represents the true values and the orange line represents the predicted values.

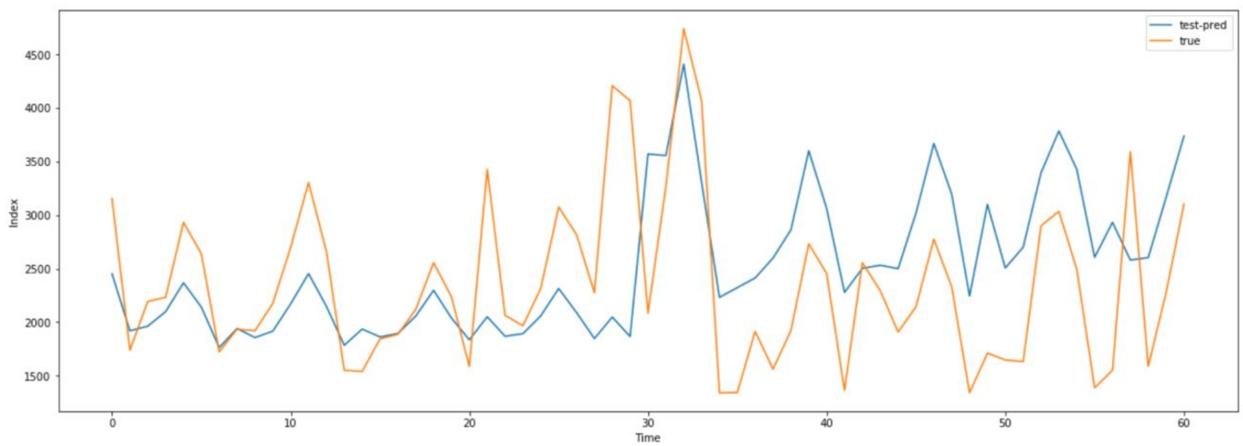


Figure 12: The testing results of the LSTM model. The blue line represents the true values and the orange line represents the predicted values.

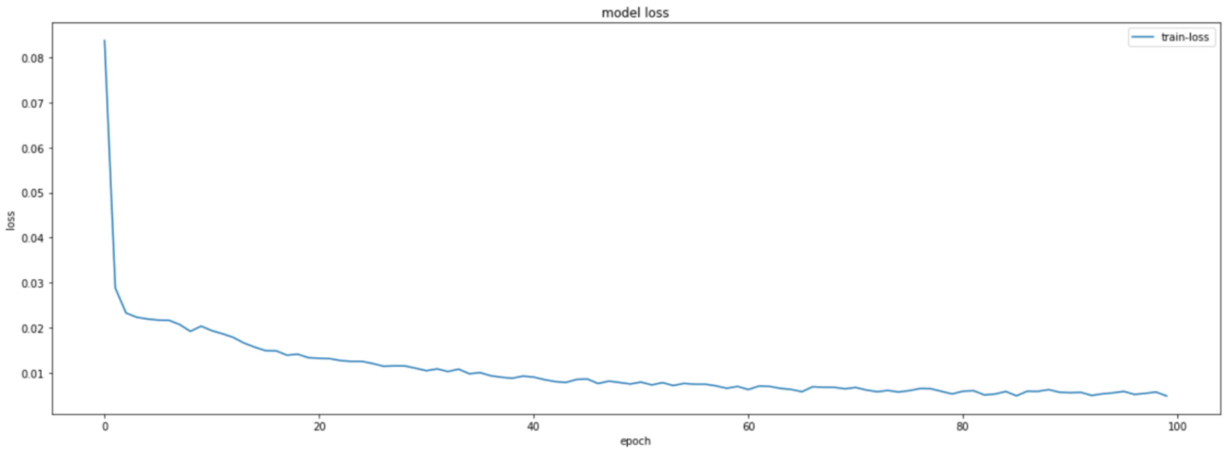


Figure 13: The change of the training loss.

We make longer predictions with the LSTM model and discovered that the predicted values are very similar to the true values in that the fluctuation of the two series look synchronized. According to the Figure 13, the training loss reduces rapidly in the first 20 epoch and becomes very low and stable in the next 80 epochs. The results shown in Figure 11 and 12 indicate that the deep learning method seems to obtain better performance than the ARIMA model. We will talk more about these graphs in section 5.

4.3 The Application of LSTM on Another Time Series Dataset

In order to see how well the LSTM model can adapt to another dataset, we decided to introduce another economic dataset, the Bitcoin dataset. This dataset contains the highest daily price, lowest daily price and the closed daily price of the Bitcoin. What we want to the is to predict the highest price of the future days based on the history, which can give people some guidance with respect to buying Bitcoins. The results are shown in Figure 14, 15 and Table 3. We will provide deeper analysis in Section 5.

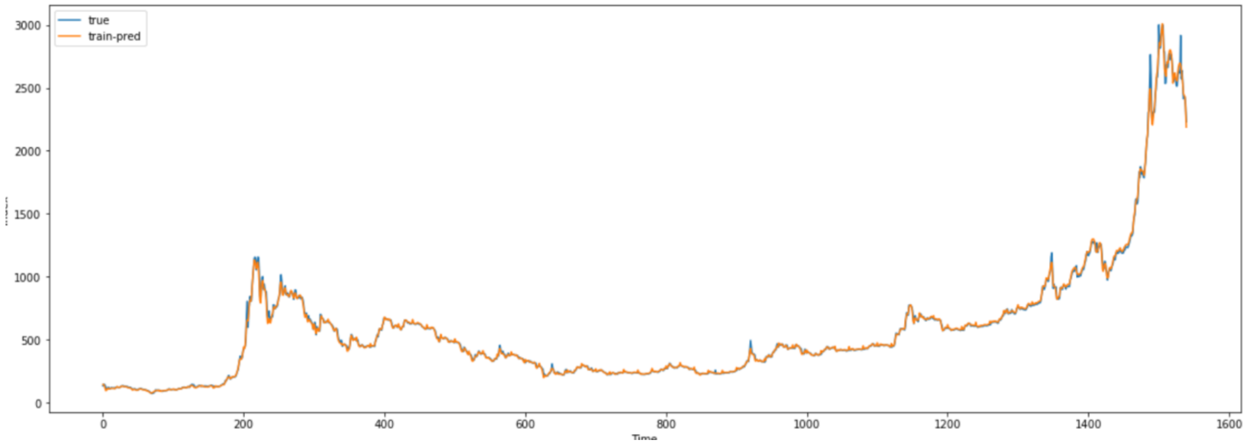


Figure 14: The training results. The blue line represents the true values and the orange line represents the predicted values.

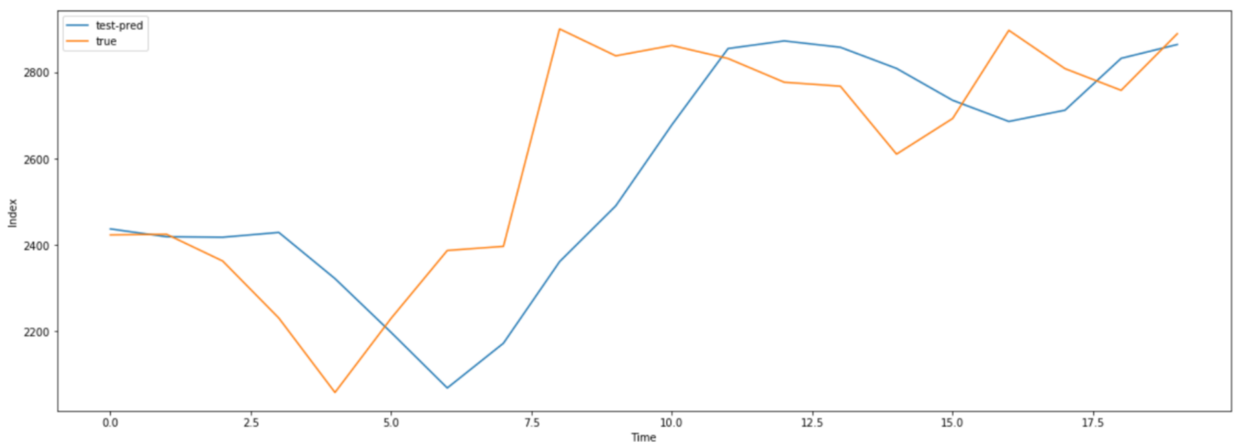


Figure 15: The testing results. The blue line represents the true values and the orange line represents the predicted values.

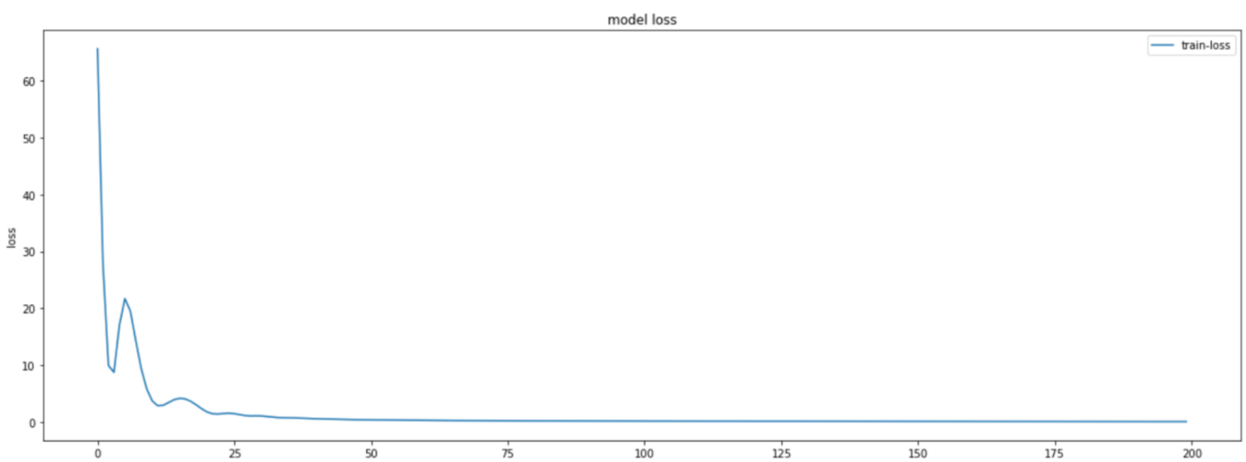


Figure 16: The change of the training loss.

5 Evaluation and Comparison

We use three metrics to evaluate the performance of our models, including root mean square error (RMSE), mean average error (MAE), and mean absolute percentage error (MAPE).

$$RMSE = \sqrt{mean((y_{pred} - y_{true})^2)}$$

$$MAE = mean(abs(y_{pred} - y_{true}))$$

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{y_{pred}(t)-y_{true}(t)}{y_{true}(t)}\right|$$, where n is the length of y_{pred}.

These statistics are shown in Table 3 and 4.

method	RMSE (Train)	RMSE (Test)	MAE (Train)	MAE (Test)	MAPE (Train)	MAPE (Test)
Log-transform	1034.3049	1012.2720	647.3656	663.8641	17.7985	32.6026
MA	933.4927	950.0631	588.3841	812.5495	16.8305	23.0449
WMA	967.0826	1032.2966	614.1245	664.4654	17.7977	31.9192
LSTM	619.8631	731.6216	411.2765	463.1651	16.1926	21.4815

Table 3: The evaluation metrics for the sales data set.

	RMSE	MAE	MAPE
train	19.3284	9.3928	1.7115
test	203.5678	151.9092	5.8590

Table 4: The evaluation metrics for the Bitcoin data set.

We can see that our ARIMA model is not performing as well as the LSTM model. For the MA data and WMA data, the ARIMA model can predict the trend of ten days in the future. After that, the prediction accuracy declines very rapidly. And the ARIMA model is not sensitive to the extreme values. The predicted values only fluctuate around the average value.

As for the LSTM approach, the predicted values have almost the same behavior as the true values. However, in the first 30 days, the predicted values are usually smaller than the true values while in the last 30 days, the predicted values tend to be bigger than the true values.

This is probably because the true values have a decreasing trend, but our model cannot depict it very well. In general, the results reflect the learning power of the LSTM model.

As for the Bitcoin price prediction, we also achieved very good training and testing results. Figure 14 shows that the training results are almost the same as the true values. The RMSE, MAE and MAPE values for the training data are also very small. In addition, in the testing set, the predicted values have a lag ahead of the true values, which is a common problem in time series prediction.

6 Conclusion

In this report, we discussed the performance of ARIMA model and LSTM model on the sales data and experimented on the Bitcoin data. There are several things that one can do to further improve the performance. For example, one can collect more information about the customers, the statistics of the local economy, etc. Unfortunately, we are not able to do all that in this project. Time series prediction is not a trivial thing, we also need more statistical and mathematical knowledge to better this project.

7 Reference

[1] Box, George EP, et al. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
[2] https://www.kaggle.com/c/competitive-data-science-predict-future-sales/data
[3] http://colah.github.io/posts/2015-08-Understanding-LSTMs/
[4] https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc442.htm
[5] https://www.investopedia.com/articles/technical/060401.asp
[6] https://www.sciencedirect.com/topics/medicine-and-dentistry/bayesian-information-criterion