MATH 509 - Data Structures and Platforms

University of Alberta

Final Project: Multiple Time Series Forecasting

Authors:
Juxin Fa (ID: 1786677)
Shiran Ye (ID: 1459814)
Mengxuan Liu (ID: 1573548)

Date: December 9, 2022

Contents

1 Introduction

With the advent of the big data era, information becomes increasingly mixed, and a single analysis model is no longer sufficient to meet modern society's data analysis needs. In this project, we want to compare two essential time series forecasting models in solving one problem. We obtained a dataset from the Kaggle website. It is a sales record for the thousands of product families offered by Ecuadorian Favorita outlets. There are several features, such as Date, kind, shop number, the family of items, etc. We will utilize two distinct mechanical models to estimate future earnings using ARIMA(Autoregressive Integrated Moving Average models) and LSTM(Long Short-Term Memory) with the time series that comes with the database, concentrating on the sales of each item in each store.

2 Methods

2.1 ARIMA

Autoregressive Integrated Moving Average models (ARIMA) are widely used in time series forecasting. The goal of ARIMA models is to describe autocorrelation in the data. Before introducing ARIMA models, we will first discuss Autoregression models (AR) and Moving Average models (MA).

An AR model forecasts the variable of interest using a linear combination of past values of the variable [3]. An AR model of order p, denoted by $AR(p)$ can be written as

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} \cdots \phi_p y_{t-p} + \epsilon_t, \tag{1}$$

where ϵ_t is white noise. This is like a multiple regression but with lagged values of y_t as predictors. Changing ϕ_i results in different time series patterns. The variance of ϵ_t will only change the scale of the series, not the patterns [3]. Usually, Stationary time series are used to build AR models.

An MA model uses past forecast errors in a regression-like model [3]. An MA model of order q, denoted by $MA(q)$, is written as

$$y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q}, \tag{2}$$

where ϵ_t is white noise. Notice that each value of y_t can be thought of as a weighted moving average of the past few forecast errors. Changing θ_i results in different time series patterns. The variance of ϵ_t will only change the scale of the series, not the patterns [3].

If we combine differencing with autoregression and a moving average model, we obtain a non-seasonal ARIMA model:

$$y'_t = c + \phi_1 y'_{t-1} + \phi_p y'_{t-p} + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t, \tag{3}$$

where y'_t is the differenced series. This model is denoted by $ARIMA(p, d, q)$, where

p = order of the autoregressive part;

d = degree of first differencing involved;

q = order of the moving average part.

We can also rewrite this equation using backshift notation B, where $By_t = y_{t-1}$:

$$\underbrace{(1-\phi_1 B - \cdots - \phi_p B)}_{AR(p)}\underbrace{(1-B)^d y_t}_{d \text{ differences}} = c + \underbrace{(1+\theta_1 B + \cdots + \theta_q B^q)\epsilon_t}_{MA(q)} \tag{4}$$

$(1-\phi_1 B - \cdots - \phi_p B)$ represents $AR(p)$, $(1-B)^d y_t$ represents d differences and $(1+\theta_1 B + \cdots + \theta_q B^q)\epsilon_t$ represents $MA(q)$.

To determine the order of an ARIMA model, Akaike's Information Criterion (AIC) can be used [3]. Its formula is

$$\text{AIC} = -2\ln(L) + 2(p+q+k+1), \tag{5}$$

where L is the likelihood of the data and $k = \begin{cases} 0 & c = 0 \\ 1 & c \neq 0 \end{cases}$.

For ARIMA models, the corrected AIC can be written as

$$\text{AICc} = \text{AIC} + \frac{2(p+q+k+1)(p+q+k+2)}{T-p-q-k-2}, \tag{6}$$

Good models are obtained by minimizing the AIC or AICc.

Once the order has been identified, we estimate the parameters:c, ϕ and θ via maximum likelihood estimation (MLE) [3]. For ARIMA models, MLE is similar to the least squares estimates. Therefore, MLE would be obtained by minimizing

$$\sum_{t=1}^{T} \epsilon_t^2. \tag{7}$$

2.2 LSTM

Based on the time series prediction with Recurrent Neural Network (RNN), Hochreiter and Schmidhuber developed a more efficient method, Long Short-Term Memory (LSTM) [2]. RNN is a chain-like network, passing each state output as the next state input. The basic formula for RNN is

$$h_t = f_w(h_{t-1}, x_t), \tag{8}$$

where h_t is the current output for the current cell C_t, h_{t-1} is the previous output, x_t is the current input. tanh is often used as f_w.

However, RNN only has the ability to process the most relevant past value to the future value. To predict future value based on the far-end past value, LSTM provides both long-term memory and short-term memory. Based on RNN, LSTM contains four parts: forget gate, input gate, cell state and output gate [6].

The forget layer takes the previous output h_{t-1} and current input x_t, and outputs a numeric value f_t between 0 and 1 controlled by sigmoid, to control the degree of forgetting. The formula of f_t is written by

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{9}$$

where W and b denote the weight and the bias respectively.

The next part input gate is combined by a pointwise multiplication operation of a sigmoid layer and a tanh layer. i_t decides which value is updating, and $\widetilde{C}_t$ creates the value for updating, as same as in RNN.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{10}$$

$$\widetilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{11}$$

The cell state combines the forget gate and the input gate to update the old state C_{t-1} to C_t by multiplying the forget degree to the previous cell, and add the multiplication operation of the sigmoid layer and tanh layer

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C}_t \tag{12}$$

The last part is the output gate, the sigmoid layer o_t filtered the value for output and through the tanh layer map to between -1 and 1.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{13}$$

$$h_t = o_t * \tanh(C_t) \tag{14}$$

3 Analysis

3.1 Dataset and Data pre-processing

The data set is from the Kaggle competition "Store Sales - Time Series Forecasting"[5].

In the train data that we cleaned, there are 3,000,887 numbers of data that contains 54 different stores with 33 different products group and their sales data in the time range from 2013-01-01 to 2017-08-15. For each day, there are 54 * 33 equals 1782 sales data.

	Store	Product	Sales	Day
Number of classes	54	33	1	150

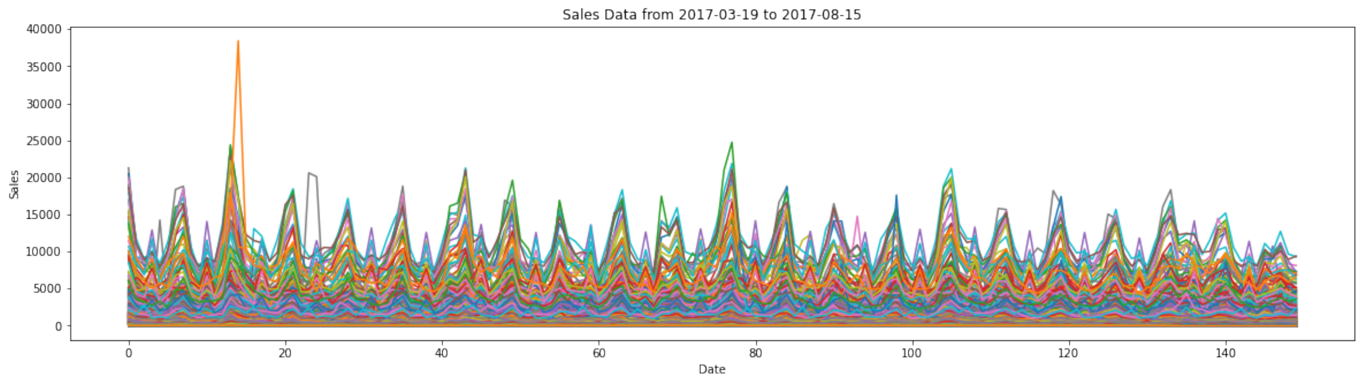

Figure 1: Sales data from 2017-01-01 to 2017-08-15, 1782 time series in total

The evaluation by the requirement of the Kaggle competition is using Root Mean Squared Logarithmic Error (RMSLE) [5]:

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(1+\hat{y}_i) - \log(1+y_i))^2} \tag{15}$$

Initially, the data need to be pre-processed. First, We split *train.csv* into two parts: the training data and the validation data, with a ratio of 99%. Originally, the shape of training data is (150, 1782). The window size of the training data is 16 days, the same as the predicted data. For both "X_train" and "y_train", the shape is (149, 16, 1782), and for "X_valid" and "y_valid", the shape is (1, 16, 1782).

3.2 ARIMA Method

Since all time series require their own ARIMA models, an AUTO-ARIMA process, implemented by the function pmdarima.arima.auto_arimam, is used to prove efficiency [7]. The function is based on a variation of the Hyndman-Khandakar algorithm [3]. It is a stepwise procedure to search in the model space and choose the model with the lowest AICc and MLE values [4]. We have the algorithm as follows.

Hyndman-Khandakar algorithm for automatic ARIMA modelling

Step 1: The number of differences $0 \leq d \leq 2$ is determined using repeated KPSS tests
Step 2:
(a) try four possible models to start with

- ARIMA(2, d, 2)
- ARIMA(0, d, 0)
- ARIMA(1, d, 0)
- ARIMA(0, d, 1)

If $d = 2$, $c \neq 0$; if $d \leq 1$, $ARIMA(0, d, 0)$ without a constant
(b) The best model (with the smallest AICc value) fitted in step (a) is set to be the "current model".
(c) Variations on the current model are considered:

- vary p or q from the current model by ± 1
- include/exclude c from the current model.

The best model considered so far (either the current model or one of these variations) becomes the new current model.
(d) Repeat Step 2(c) until no lower AICc can be found.

Using auto_arima, 1782 ARIMA models are built based on the training set, then the corresponding prediction values can be obtained.

3.3 LSTM Method

Since the output requires 16 days of prediction, the LSTM model selects the previous 16 days as input and predicts the future 16 days. The window size is 16 days for both training and prediction, and it moves from day 0, day by day, to the end of the date.

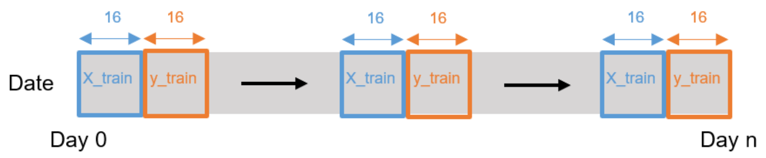

Figure 2: Training and Validation windows

The loss function for the LSTM model training selects Mean Squared Error (MSE), Where n is the number of data, y_i is the real value, and $\widehat{y}_i$ is the predicted value. The model does not choose RMLSE as the loss function due to the bad empirical results.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{16}$$

The architecture of the LSTM model is designed in three main layers, each layer contains one LSTM layer, one batch normalization layer and one dropout layer. Finally, add a time-distributed layer.

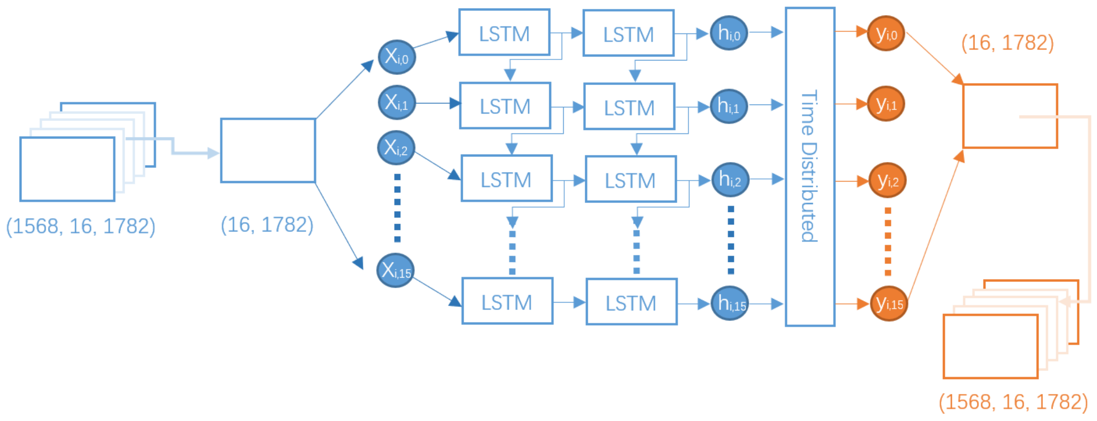

Figure 3: LSTM Model Layer

4 Prediction Results

The results obtained by ARIMA and LSTM are shown in Figures 4 and 5 respectively. By comparing the two plots, we can intuitively see that overall the inference of LSTM will outperform ARIMA. We can see from ARIMA's conclusion that while the model captures the broad trend, it does substantially

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 16, 50)	366600
dropout (Dropout)	(None, 16, 50)	0
lstm_1 (LSTM)	(None, 16, 50)	20200
dropout_1 (Dropout)	(None, 16, 50)	0
time_distributed (TimeDistributed)	(None, 16, 1782)	90882
Total params: 477,682		
Trainable params: 477,682		
Non-trainable params: 0		

Table 1: Model Summary: LST

worse in terms of precision. Meanwhile, LSTM operates admirably throughout the early stages of data prediction, but with time, the loss function of LSTM begins to exhibit a significant change.

We set the last 16 days in the training set as the baseline and compare the results of ARIMA, LSTM and baseline via RMSLE, which are shown in Table 2. A similar conclusion can be reached: when compared to the baseline, ARIMA only improves by 23.02%, whereas LSTM improves by up to 34%.

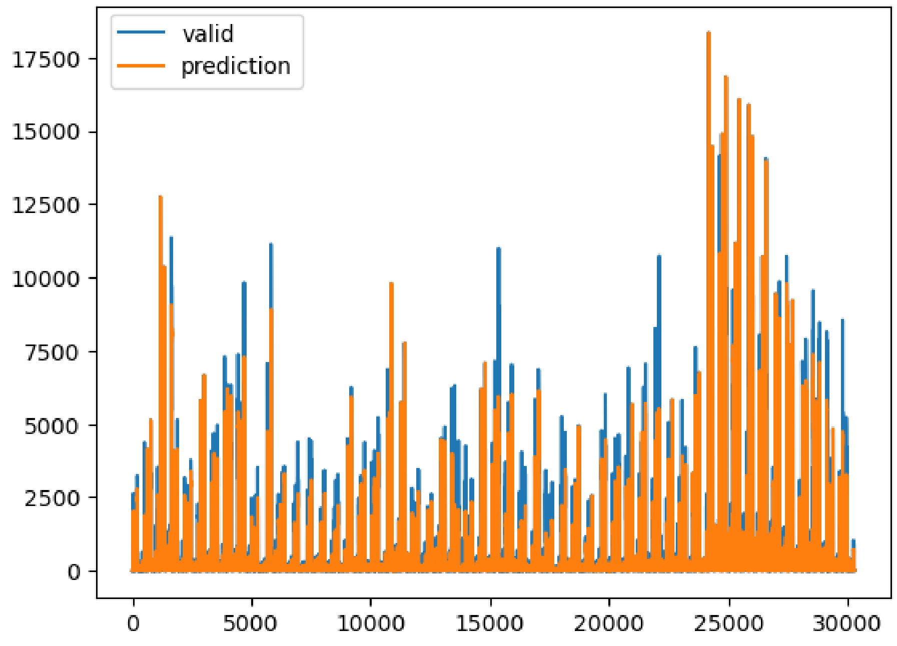

Figure 4: ARIMA validation set vs prediction

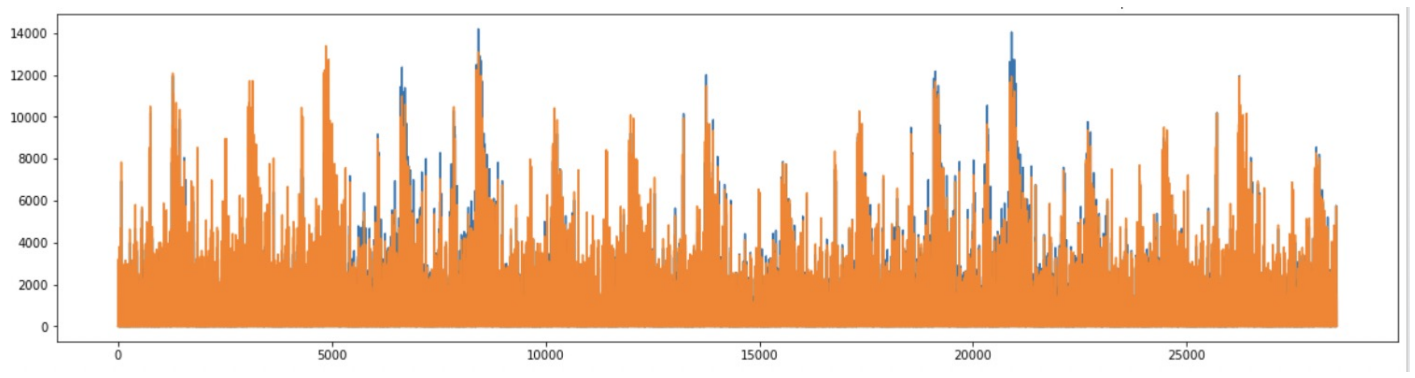

Figure 5: LSTM validation set vs prediction

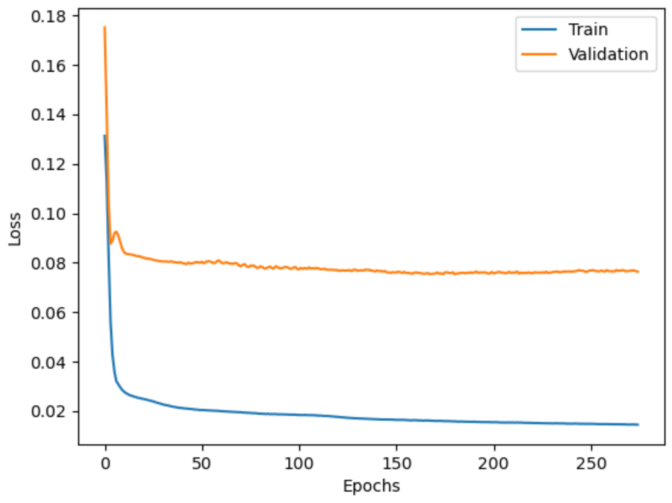

Figure 6: LSTM Train and Validation Loss

Model	Valid-MSE	Valid-RMSLE	Test-RMSLE
Empty Entry	-	-	4.37969
BaseLine	-	-	0.73163
ARIMA	413764	0.545413	0.56323
LSTM	94258	0.469778	0.47928

Table 2: Result Comparison

5 Conclusion

To a certain degree, ARIMA models can predict future sales, but the results are not satisfying. Here are some possible reasons. First, not all models built by auto_arima are optimal. Second, some time series may show strong seasonality, and seasonal ARIMA may be a better choice. However, the running time for seasonal models is much higher than non-seasonal models, we did not finish building these models. Third, the sales are also influenced by some exogenous variables such as oil prices, holidays and store locations.

LSTM model shows great performance on time series prediction. However, using only time series to predict time series might lose some important features, and LSTM is not capable of multi-time series and multi-feature prediction. This reaches the model's limit and cannot provide a more robust prediction through multiple time series. Adding more features may increase prediction accuracy. Moreover, using CNN to extract the time series data features may also help increase the performance. Recently, a study showed that "Attention" is more suitable for time series prediction [1].

6 Appendix

You can find the code in this GitHub repository.

References

[1] M. del Pra. Time series forecasting with Deep Learning and Attention Mechanism, 2020. Accessed: 2022-11-30.

[2] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[3] R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2 edition, 2021.

[4] Rob J. Hyndman and Yeasmin Khandakar. Automatic time series forecasting: The forecast Package for R. *Journal of Statistical Software*, 27(3), July 2008.

[5] Kaggle. Store sales - time series forecasting, 2022. Accessed: 2022-10-30.

[6] C. Olah. Understanding LSTM networks, 2015.

[7] T. G. Smith. pmdarima.arima.auto_arima, 2017. Accessed: 2022-11-20.