Sale Prediction Using LSTM

Zening Li, Qingxu Wang

Abstract

Prediction methods are useful on nowadays merchandise. Predictions on price, sales can help merchants understand the trend of the market, so they can adjust the trading decisions to maximize profit and minimize cost. For example, PS2 was the best seller several years before, however the trend was going down, so the merchant would order less PS2 in case there’s fewer buyers.

This project is supposed to predict item sales amounts on November, 2015 for products from different shops in Russia. The dataset includes sales price, sales amount, date, shop_id and item_id on daily basis from January, 2013 and ends on October, 2015. We grouped the data by month and item and shops, and finally we can get two data frames where one is sales amount frame the other is price frame, combine those two frames into one and perform LSTM to get the prediction values.

The final prediction score on monthly sales amount per item per shop is around 1.2 to 1.3 where almost all kernel scores using LSTM fall at.

1. Introduction

Sales amount prediction is something really important in real world. It is a problem every merchant wants to solve so that they can understand the market and get as much money as they can.

The prediction methods are developed these years with the development of machine learning. Using machine learning and deep learning in python enhanced the accuracy rate a big deal than usual predictive skills. In our project, the method we used is LSTM, since we found that Arima is not a great method to predict more than one column of data.

We used Arima at first however the accuracy score was not good at all, because we have to ignore price effect due to the limitations of Arima method in time series. Also, the type and the trend of the data is not a good fit on Arima models.

Using LSTM in deep learning give us better prediction and can let us input more than one column of variables, the final prediction is more and more accurate.

The possible improvement that can be applied on this project is to distinguish the holidays and special days(such as black friday which may affect the sales amount a great deal), we don’t use it in our model yet, however it is possible to make the model better.

2. Related Works

We made our project after checking the method used in this github kernel: **https://www.kaggle.com/john850512/predict-future-sales-lstm**

However, in this kernel only one variable is used and in our project two variables(price and counts) are used in order to make the model more complicated and more caccurate. We basically use the similar method to clean the data, and the similar method to do the prediction. Since our purpose for this project is to learn the LSTM method and how to process time series data, we used some codes from the github kernel, however, most thoughts are from ourselves.

In the future, we could use some other way to do the prediction more accurately.

3. Details of Approach

This section is about the details of our model. First of all, we provide the principle of the method we use. Then, we show the description of the dataset. After that, we show the reason of using LSTM. Finally, we present the exploratory data analysis for the dataset.

3.1 Method

The method we use is LSTM. LSTM means long short-term memory. This is a gated RNN, which is the "networks based on the gated recurrent unit (Ian 2016, p397)." First of all, we are going to introduce what is RNN.

3.1.1 RNN

RNN means recurrent neural network. RNN is a network which have loops in it. It is a neural network which loops multiple times by time steps. Each time step data get into the loop will get an output. Thus, it is used for sequential data processing such as time-based data. Since our project is to predict future sales with sequential time-based dataset. It is convenient to use RNN to do the model. From the book "Deep Learning", it shows the design patterns for recurrent neural network which are: "1. Recurrent networks that produce an output at each time step and have recurrent connections between hidden units. 2. Recurrent networks that produce an output at each time step and have recurrent connections only from the output at one time step to the hidden units at the next time step. 3. Recurrent networks with recurrent connections between hidden units, that read an entire sequence and then produce a single output (Ian 2016, p368)." Follow up the design patterns, we know that we could have multiple features in our model. RNN can handle well on multiple features.

However, there are problems on RNN. It cannot deal with Long-Term Dependency. For example, if the output we want to predict have a long gap between one step which contains useful information for this prediction, regular RNN cannot figure out the connection between these two steps. LSTM can solve this problem really well.

3.1.2 LSTM

LSTM are used for avoiding the Long-Term Dependency problem. The reason is that LSTM have gate to control. There also exists cell state in LSTM. “The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions (Colah 2015).” Figure 1 is the basic structure of LSTM.

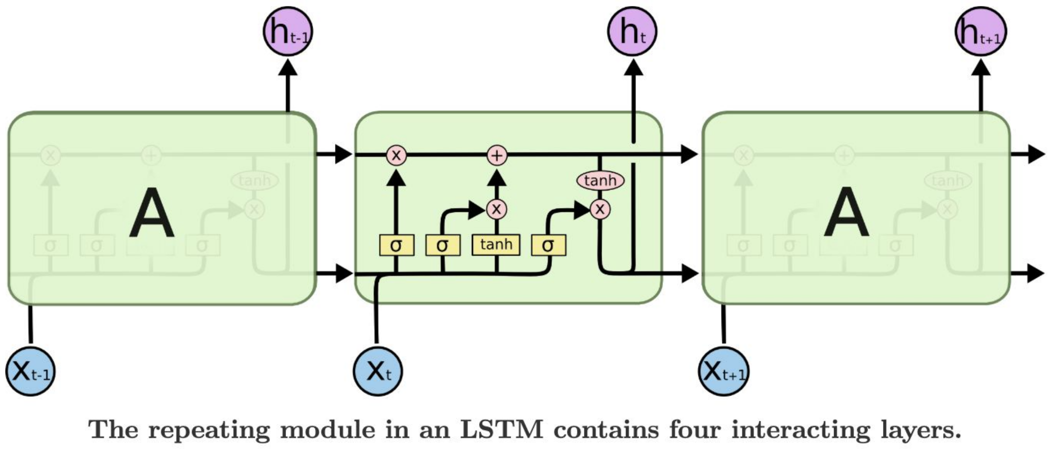

h_{t-1}
h_t
h_{t+1}
A
σ
σ
tanh
σ
tanh
A
X_{t-1}
X_t
X_{t+1}
The repeating module in an LSTM contains four interacting layers.

Figure 1: The structure of LSTM (Colah 2015)

From figure 1, there are four neural network layers which are “tanh layer, input gate layer, forget gate layer, and output gate layer (Colah 2015).” The forget gate layer loop at the previous output and the current input to present an output about whether keep the previous output or not. The input gate layer is to find out the value need to update. The tanh layer is “creates a vector of new candidate values to added to the state (Colah 2015).” The output gate layer is a sigmoid layer to “decides what parts of the cell state we’re going to output (Colah 2015).”

LSTM is basically following these layers and do the prediction. The first step is to use “ (Colah 2015)” to show the forget state. is the input, is the output, is the weight, and is the bias. The second step is to “decide what new information we’re going to store in the cell state (Colah 2015).” It contains two sub steps: use “ (Colah 2015)” to find the values to update; use “ (Colah 2015)” to create vector to add to the state. Then “combine these two to create an update to the state (Colah 2015).” is the input, is the output, and are the weight, and and are the bias. The third step is to update to by “(Colah 2015)”. is the “forgetting the things we decided to forget earlier (Colah 2015).” is “the new candidate values, scaled by how much we decided to update each state value (Colah 2015).” The last step is to find the output. There are two sub steps for this step. The first one is to “run a sigmoid layer to decide what parts of the cell state we’re going to output by (Colah 2015).” The second one is to “put the cell state through tanh (to push the values to be between −1−1 and 11) and multiply it by the output of the sigmoid gate by

(Colah 2015).” All the sigmoid functions return zero or one. These steps are LSTM basically work. Figure 2 is a graph to illustrate a single block.

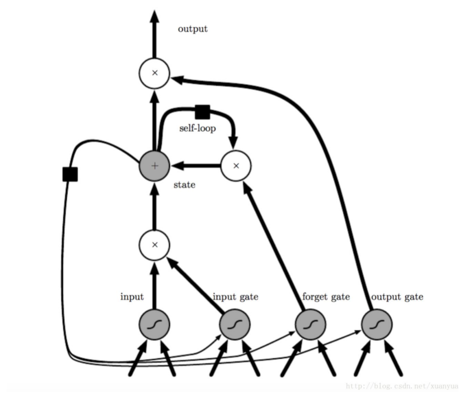


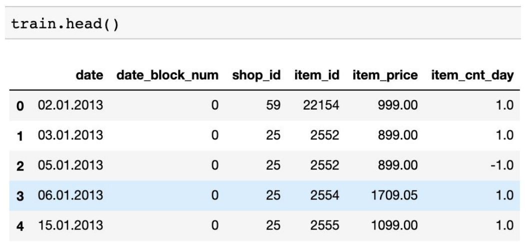
Figure 2 (Xuanyuan)

3.2 Dataset description

Our dataset is from a kaggle competition. This kaggle competition is mainly for education purposes. So it does need to do real future prediction. The dataset provided by 1C company, which is a software company in Russia.

The dataset contains six data files. The first file is the “sales_train.csv”, which is the sales information from January 2013 to October 2015. The second file is the “test.csv”, which is the specific shops and specific items need to forecast for November 2015. The third file is the “sample_submission.csv”, which is the format sample for the submission of the competition. The fourth file is the “items.csv”, which is the information about products for sale. The fifth file is the “item_categories.csv”, which contains the product categories information. The last file is the “shops.csv”, which contains the shops information.

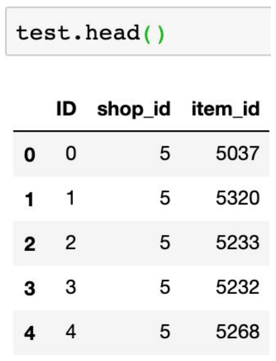
Here are the samples of the dataset:



```
train.head()
```

	date	date_block_num	shop_id	item_id	item_price	item_cnt_day
0	02.01.2013	0	59	22154	999.00	1.0
1	03.01.2013	0	25	2552	899.00	1.0
2	05.01.2013	0	25	2552	899.00	-1.0
3	06.01.2013	0	25	2554	1709.05	1.0
4	15.01.2013	0	25	2555	1099.00	1.0

Table 1: the top five lines of “sales_train.csv”



```
test.head()
```

	ID	shop_id	item_id
0	0	5	5037
1	1	5	5320
2	2	5	5233
3	3	5	5232
4	4	5	5268

Table 2: the top five lines of “test.csv”

Since we only need to predict the future sales of specific shops and specific items, so we decided to use “sales_train.csv”, “test.csv” in order to do the prediction. The “sales_train.csv” contains the “date” which is the date of sales happen, “date_block_num” which is the specific month from zero to thirty-three of sales happen, “shop_id” which is the specific shop ID, “item_id” which is the specific product ID, “item_price” which is the specific price for specific item sold in specific shop, item_cnt_day which is the number of specific item sold at that date. The “test.csv” contains “ID” which is the specific combination of “shop_id” and “item_id”, the “shop_id” and the “item_id” are the same as “sales_train.csv”. We will called “sales_train.csv” train file and “test.csv” test file at the rest of our report.

3.3 Reason of doing LSTM model

One of the big reasons we decide to use LSTM to do the prediction model is that the dataset is a time series data. It contains date and specific month block in the dataset. From the above explanation, LSTM is basically used for time-based prediction. The model runs by time steps. We use the “date_block_num” as steps to do the model.

Other reasons we decided to use LSTM is the data contains a long period of time. There need the gate to prevent information loss. Also, LSTM can handle multiple features. For sale prediction, the result does not only based on the previous number of sales. It also based on price. Thus, we decided to use LSTM to create our model.

We also think of using ARIMA to do the model. ARIMA is also a good method to do time series model. However, there are two problems on ARMIA. The first problem is that ARIMA can only handle on one series of data. It need more complex model combine together to do our dataset modeling. The second problem is that ARIMA need the data stationary. These figures are the price and “item_cnt_day” with “date_block_num”.

```
g1 = sns.lineplot(x="date_block_num", y="item_cnt_day", data = train)
```

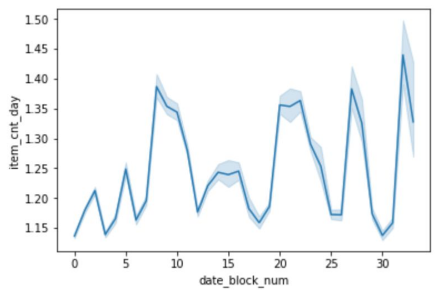


Figure 3: relationship between “date_block_num” and “item_cnt_day”

```
g2 = sns.lineplot(x="date_block_num", y="item_price", data = train)
```

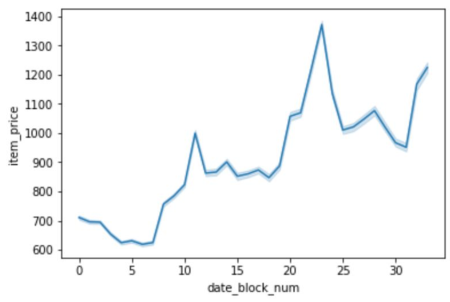
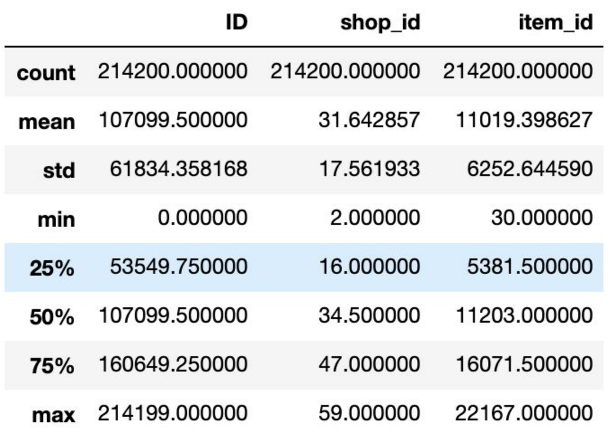


Figure 4: relationship between “date_block_num” and “item_price”

All figures present that price and number of sales are not stationary with “date_block_num”. We tried to do difference between two data points in order to make the data stationary, but the graphs still looks not stationary. Thus, it is better to use LSTM instead of ARIMA.

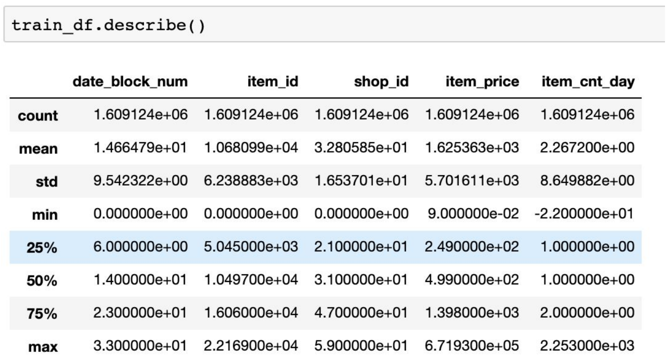
3.4 exploratory data analysis

Here is the description of the “test.csv” and “sales_train.csv”:



	ID	shop_id	item_id
count	214200.000000	214200.000000	214200.000000
mean	107099.500000	31.642857	11019.398627
std	61834.358168	17.561933	6252.644590
min	0.000000	2.000000	30.000000
25%	53549.750000	16.000000	5381.500000
50%	107099.500000	34.500000	11203.000000
75%	160649.250000	47.000000	16071.500000
max	214199.000000	59.000000	22167.000000

Table 3: description of “test.csv”



```
train_df.describe()
```

	date_block_num	item_id	shop_id	item_price	item_cnt_day
count	1.609124e+06	1.609124e+06	1.609124e+06	1.609124e+06	1.609124e+06
mean	1.466479e+01	1.068099e+04	3.280585e+01	1.625363e+03	2.267200e+00
std	9.542322e+00	6.238883e+03	1.653701e+01	5.701611e+03	8.649882e+00
min	0.000000e+00	0.000000e+00	0.000000e+00	9.000000e-02	-2.200000e+01
25%	6.000000e+00	5.045000e+03	2.100000e+01	2.490000e+02	1.000000e+00
50%	1.400000e+01	1.049700e+04	3.100000e+01	4.990000e+02	1.000000e+00
75%	2.300000e+01	1.606000e+04	4.700000e+01	1.398000e+03	2.000000e+00
max	3.300000e+01	2.216900e+04	5.900000e+01	6.719300e+05	2.253000e+03

Table 4: description of “sales_train.csv”

From the description of “test.csv”, there are 214200 samples in the dataset. From the description of “sale_train.csv”, The are more samples than “test.csv”. Since we only need to predict 214200 samples, so we need to merge the train file and test file. But first, we need to clean the data first in order to make the data fit the LSTM model.

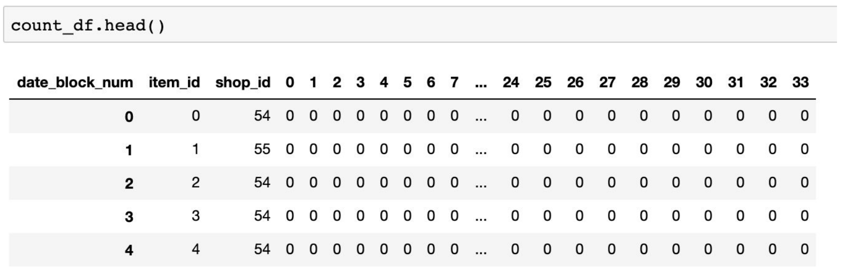
3.4.1 Data cleaning

We clean the data to make it fit the LSTM model. We want to create two data frames. One uses the “item_cnt_day” and “date_block_num”, the other uses “item_price” and “date_block_num”.

For the first dataframe, we followed these steps to do the data cleaning.

1. Drop “date” and “item price” columns
2. Use pandas “piovt_table” method to make the dataset into matrix form.

The rows are the “date_block_num”, “shop_id”, and “item_id” and the specific number of sales for specific shop and specific product, the columns is 34 month blocks. All the loss information we put zero in it. The dataset look like this:



```
count_df.head()
```

date_block_num	item_id	shop_id	0	1	2	3	4	5	6	7	...	24	25	26	27	28	29	30	31	32	33
0	0	54	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	1	55	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	2	54	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	3	54	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	4	54	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

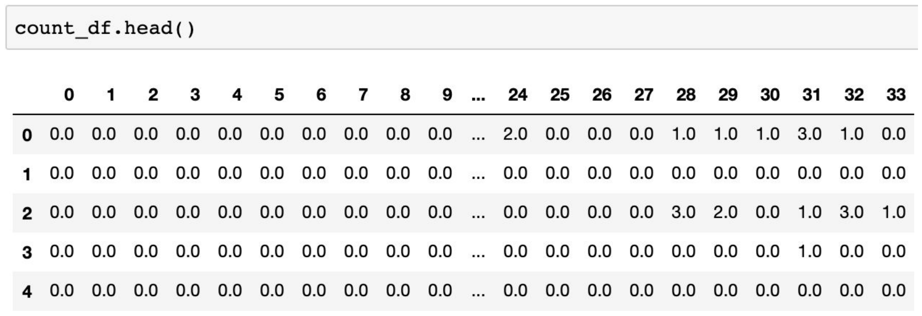
Table 5: matrix form count and date dataset

3. Merge the “test.csv” file by “item_id” and “shop_id”
4. Drop useless columns: “ID”, “date_block_num”, “item_id”, “shop_id”

For the second dataframe, we basically do the same steps except we drop “item_cnt_day” column, and in each cell contains price information.

We merge these two dataframe when we did the modeling. The explanations are in the Experiment Details section.

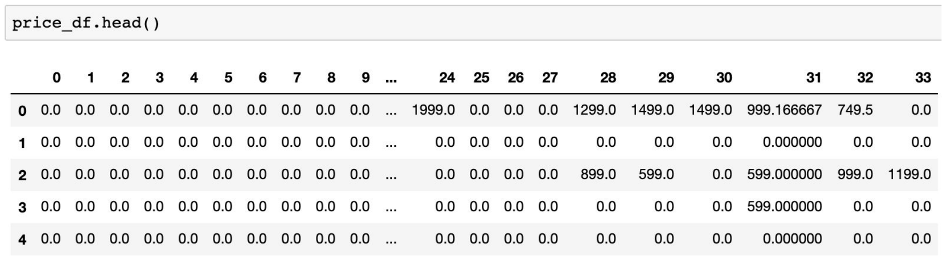
The final dataset look like this:



```
count_df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	24	25	26	27	28	29	30	31	32	33
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	2.0	0.0	0.0	0.0	1.0	1.0	1.0	3.0	1.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	3.0	2.0	0.0	1.0	3.0	1.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 6: count dataset



```
price_df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	24	25	26	27	28	29	30	31	32	33
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1999.0	0.0	0.0	0.0	1299.0	1499.0	1499.0	999.166667	749.5	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	899.0	599.0	0.0	599.000000	999.0	1199.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	599.000000	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0

Table 7: price dataset

From these two tables, it basically a time steps data with no gaps in it. These tables contain all specific “ID” with 0 to 33 “date_block_num”. These two datasets can directly used in LSTM model.

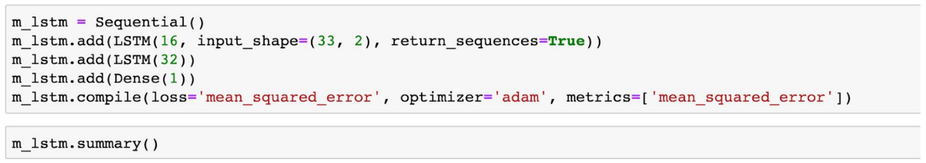
4. Experiment Details

Here are the explanations of our experiment. Since our experiment basically focused on learning time series data and LSTM, we show the procedures on doing a regular LSTM model.

First of all, we create the training set on count dataset, “x_count”, for the experiment. “x_count” is an array type. “x_count” contains 214200 rows and 33 columns. The last column we used as “y” to check the accuracy of our model. The shape of “x_count” is (214200, 33), and the shape of “y” is (214200,). Then we need to reshape “x_count” into (214200, 33, 1). The reason for that is LSTM need parameters for total number of samples, time steps, and features number to do the modeling. We did the same procedure on price dataset. Then we combine these two together by np.append to create a new “X” which the shape is (214200, 33, 2). “X” contains two features. We used “X” to do the model. For “y”, we reshaped it to (214200, 1) because we only need the number of sales.

Next, we create the test set. We drop month block one data and used the rest of the data to create the test set. We reshaped the data and combine price and count as the same form of training set, which is (214200, 33, 2). The name of the test set is “T”.

After that we create our LSTM model. Here is the code of our model:



```
m_lstm = Sequential()
m_lstm.add(LSTM(16, input_shape=(33, 2), return_sequences=True))
m_lstm.add(LSTM(32))
m_lstm.add(Dense(1))
m_lstm.compile(loss='mean_squared_error', optimizer='adam', metrics=['mean_squared_error'])
```

```
m_lstm.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_7 (LSTM)                (None, 33, 16)            1216
_________________________________________________________________
lstm_8 (LSTM)                (None, 32)                6272
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 33
=================================================================
Total params: 7,521
Trainable params: 7,521
Non-trainable params: 0
_________________________________________________________________
```

We create two layers for our LSTM model.

Then, we create the validation set, “X_val” and “y_val”, which change “X” and “y” into (19278, 33, 2) and (19278, 1) shape. The shape of “X_val” is (21420, 33 ,2), and “y_val” is (21420, 1). We create a call back list by keras Early Stopping for our model because each epoch run a really long time. “Early stopping is a method that allows you to specify an arbitrary large number of training epochs and stop training once the model performance stops improving on a hold out validation dataset. (Jason 2015)” This increase the speed of running the model. We fit our training and validation set to the model.

The last steps is that we use the test set to do the prediction. Here is the plot of the MSE:

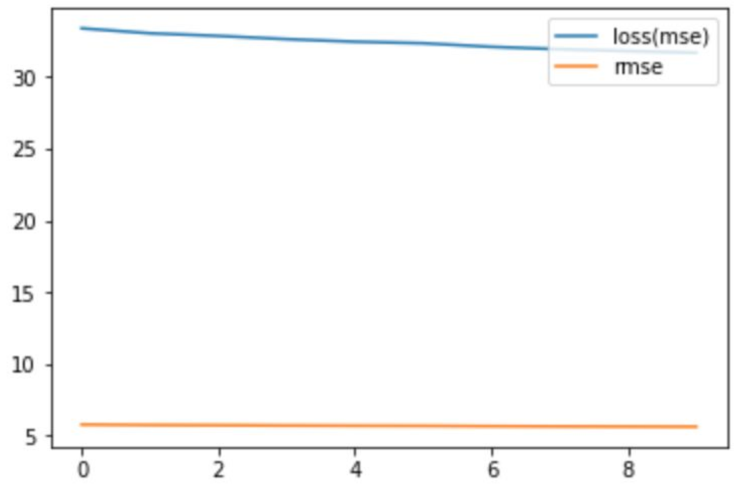


Figure 5: plot of epoches and MSE

We submit our submission to the kaggle competition, the score is 1.03461 which is at 1880 position. The score calculated by the RMSE. The result shows that our model can use to do the sales prediction.

5. Error Analysis

Our model is working properly in a whole scale, however, some details needs to be improved.

The model is LSTM and it is time series data, so there might not be missing values or the model cannot be built, the way we implemented the missing values are implemented them with all 0s, though I don't think it is a good way to do this, it is the easiest way to do this. Filling all 'NaN's with 0s can cause bias on the prediction values, but if a series have 33 values and we just got 1 value out of the 33 values, interpolation might not be a good way to do this, the interpolated values can lead the model to a wrong prediction, it is the thing confused us a bit.

Also, after the prediction, the loss-mse plot shows the loss is really high, is this the thing may happen in a correct model? This should be made sure about if we really want to use this model to predict in real world, a biased prediction can cause tons of money loss, that merchants cannot afford.

6. Conclusion

After all, the model is well performed, the accuracy is okay after several times of testing, the predictions are saved in a csv file and we submitted the file to competition, the score where is RMSE is about 1.2 which is our best score, the position is around 2000 out of 4000 competitors. LSTM is the method we used, however we can take more aspects into consideration such as holiday effects and some economic domain knowledge.

Work Cited

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. “Deep Learning”. ISBN-13: 978-0262035613. Nov 18, 2016.

Colah’s Blog. “Understanding LSTM Network”. August 27, 2015. http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Xuanyuan Sen. “Introduction to LSTM Algorithm Principle and Tutorial”. March 13, 2017. Follow the CC 4.0 by-sa copyright agreement. https://blog.csdn.net/xuanyuansen/article/details/61913886

Jason Brownlee. “Use Early Stopping to Halt the Training of Neural Networks At the Right Time”. December 10, 2018, https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/

Some codes cited from: https://www.kaggle.com/nicapotato/multivar-lstm-ts-regression-keras and https://www.kaggle.com/john850512/predict-future-sales-lstm