Capstone Project Rossman Sales Prediction with LSTM

Yiran Tian

2020-May-26

Content

Definition

Project Overview

Currently, Rossman store managers are tasked with predicting their daily sales for up to six weeks in advance. More precisely, the target outputs are sales prediction for each day, and can up to six weeks long. As state before, these sales can be influenced by many factors, and those data can work together, to help identify the future sales figure.

To solve this issue, a machine learning model was built to fulfill these tasks. Candidate model including general supervised learning model such as SGD Regressor, Lasso, EnsembleRegressors, XGBoost, as well as RNN. To better understand the time-series prediction mechanism, the decision made was to use a deep-learning approach, with LSTM (Long-Short Term Model).

The data was explored at first, the hidden pattern was extracted during the exploration. Then, data were standardized and labeled in a meaningful way for further computations. The training and validation datasets were further constructed in a way that the model can handle. Finally, the prediction was made.

Problem Statement

Rossman store provides around 3 years of historical data including sales, promotion status, and others. Through this project, Rossman store hope competitors create model and training with this data, to predict their sales figure in the coming weeks.

The problem itself is a typical time-series supervised learning issue. The data Rossman provided was time labeled sales and store status data. Store status including Open Status, Promotion Status, State Holiday Status and School Holiday status, etc. That status indicates the condition of the given date in the store. For prediction, the store status is also provided. The model should take those statuses as input, to generate a prediction for sales.

Metrics

Metrics was assigned to measure the model prediction's deviation between the real sales figure and predicted value. According to the Kaggle competition page, the chosen Metrics was RMSPE (Root Mean Square Percentage Error), shown as below:

$$RMSPE = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2$$

This RMSPE will be used as the final metric to evaluate a model's performance.

Besides, during training, BCELoss (Binary Cross-Entropy Loss) was also introduced as the loss function. The BCELoss will be used to evaluate the model's performance during the training stage, and the best performance model will be used to predict a result, and eventually evaluate by RMSPE approach. BCELoss function is as:

$$l(x, y) = L = \{(1, \cdots)^{l_N}\}^T, \qquad l_n = -\omega_n[y_n \times log\, x_n + (1 - y_n) \times log(1 - x_n)]$$

Analysis

Data Exploration

Rossman store provides the following data files for this project:

- **train.csv**: historical data including Sales
- **test.csv**: historical data excluding Sales
- **sample_submission.csv**: a sample submission file in the correct format
- **store.csv**: supplemental information about the stores

For train.csv and test.csv, they all contain the following data fields:

- **Id**: an Id that represents a (Store, Date) duple within the test set
- **Store**: a unique Id for each store
- **Sales**: the turnover for any given day (this is what you are predicting)
- **Customers**: the number of customers on a given day
- **Open**: an indicator for whether the store was open: 0 = closed, 1 = open
- **StateHoliday**: indicates a state holiday. Normally all stores, with few exceptions, are closed on state holidays. Note that all schools are closed on public holidays and weekends. a = public holiday, b = Easter holiday, c = Christmas, 0 = None
- **SchoolHoliday**: indicates if the (Store, Date) was affected by the closure of public schools
- **StoreType**: differentiates between 4 different store models: a, b, c, d
- **Assortment**: describes an assortment level: a = basic, b = extra, c = extended
- **CompetitionDistance**: distance in meters to the nearest competitor store
- **CompetitionOpenSince**[Month/Year]: gives the approximate year and month of the time the nearest competitor was opened
- **Promo**: indicates whether a store is running a promo on that day
- **Promo2**: Promo2 is a continuing and consecutive promotion for some stores: 0 = store is not participating, 1 = store is participating
- **Promo2Since**[Year/Week]: describes the year and calendar week when the store started participating in Promo2
- **PromoInterval**: describes the consecutive intervals Promo2 is started, naming the months the promotion is started anew. E.g. "Feb, May, Aug, Nov" means each round starts in February, May, August, November of any given year for that store

According to provided data, there are 1115 stores with around 2 years of data provided. Most of the data is complete. However, there still some NA value in some columns that need handle before further analysis.

The sales figure was analyzed to better capture the pattern within. The yearly pattern and weekly pattern were specially studied. Those patterns will further influence our model design decision.

Exploratory Visualization

Weekly Pattern

To study the weekly pattern, four stores were randomly chosen and 4 weeks of sales data were printed. Illustrate as below:

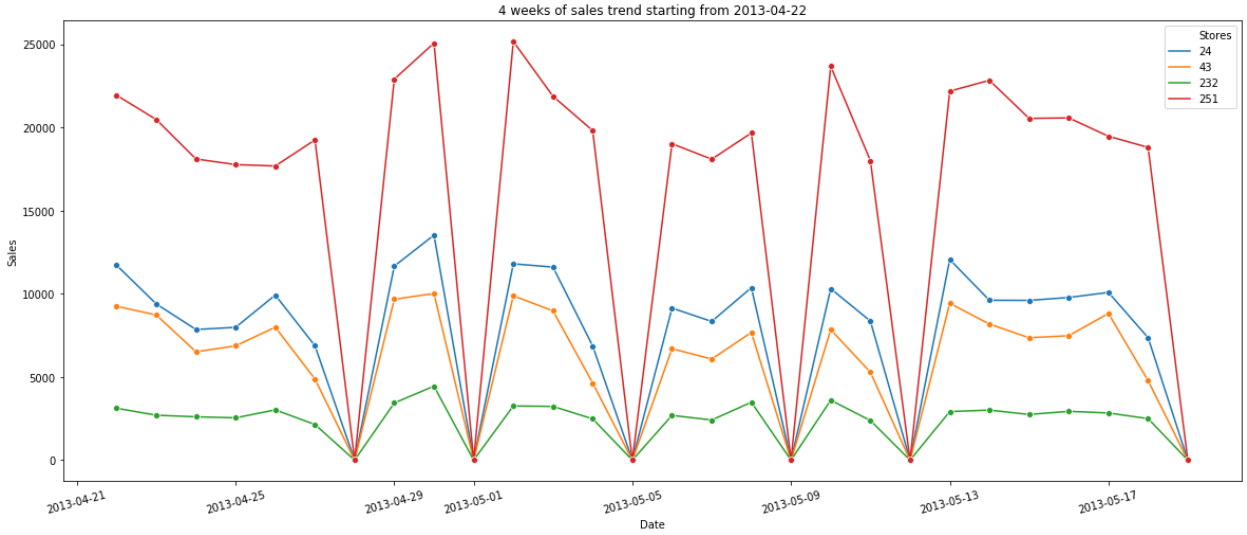

Figure 1 four weeks of randomly chosen store's sales

There are few most important patterns shown in the figure above:

- When the store is closed, there will be zero sales
- The day after the store closed, the sales are most likely above average
- There are no obvious weekly repeating pattern

Monthly Pattern

Then, 9 weeks of sales data from four randomly chosen stores were printed. This is to investigate the monthly pattern within sales figures. As below:

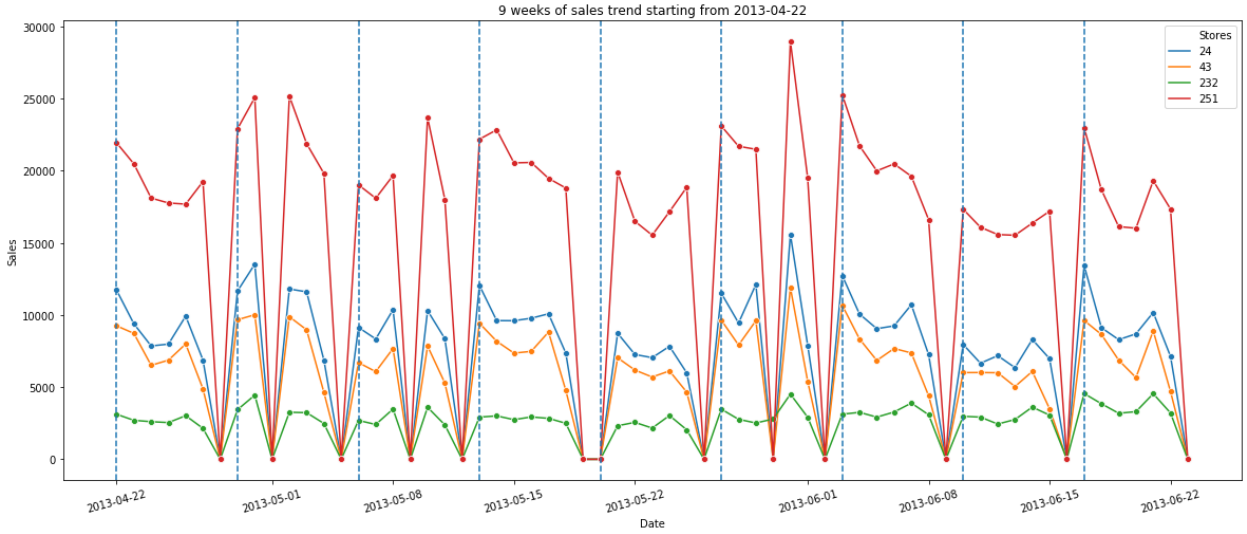

Figure 2 nine weeks of sales number from four randomly chosen stores

From the monthly sales data, no significant pattern was recognized other than those already mentioned in the weekly pattern.

Sessional Pattern

Finally, all the sales from four randomly chosen stores cross the years were printed to capture the yearly pattern.

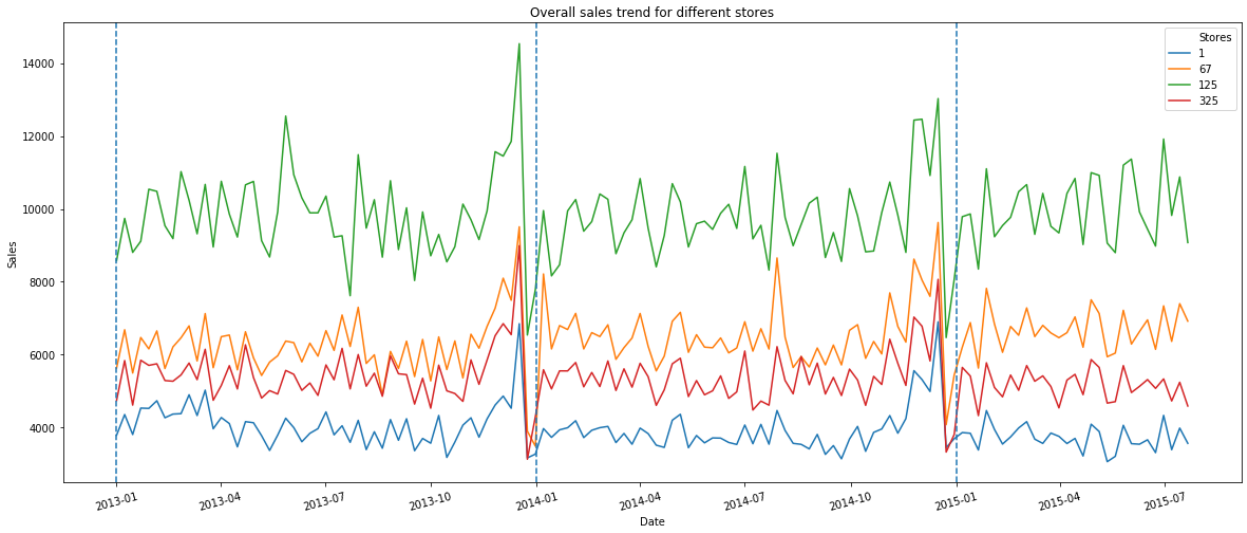

Figure 3 overall trend of sales data from four randomly chosen stores

With this yearly sales illustration, following critical pattern was recognized:

- The sessional pattern exists, especially around charismas holidays. There will be a selling peak, followed by a trough.
- There is a certain degree of similarity to the sales number. This can be easily recognized from store 67 and store 325.

Correlation Analyze

Correlation between different columns is also analyzed to better understand the data. Firstly, the column's correlation graph analyzed as below:

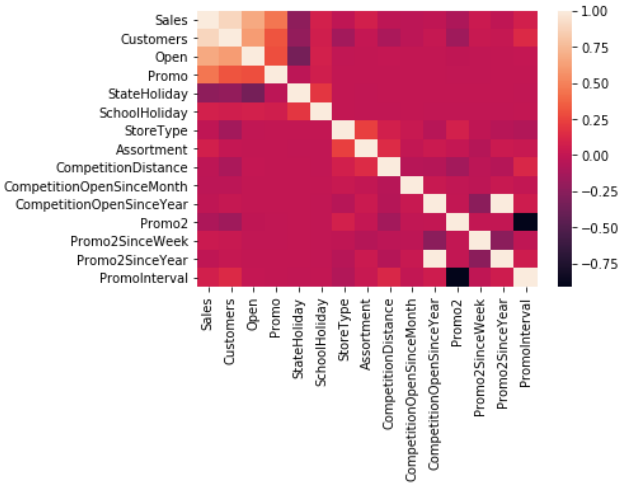

Figure 4 correlation analyze between different columns

The most concerning columns is Sales, which is the predicted value eventually. According to the result, Sales is heavily related to columns Customers, Open. That also reflects the common sense that only if a store is open it will have sales; and the more customers a store has, the more sales it should happen.

Another correlation matrix analyzed is sales number's relation cross different stores. Shown below:

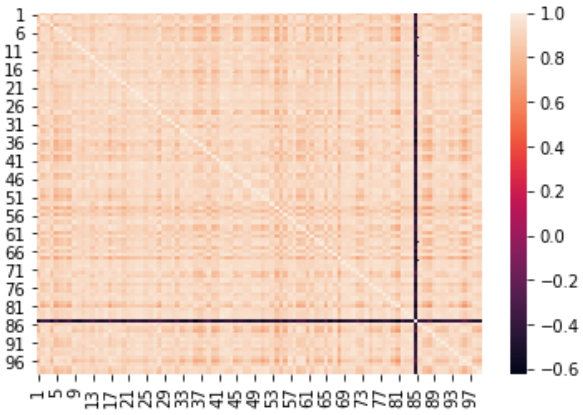

Figure 5 sales correlation analyzes between stores

According to this result, most of the stores have correlation, except for some special ones. The assumption is that most of the stores are in the same region/country, so they are all in a similar environment. Therefore, it will show a strong correlation with each other.

Algorithms and Techniques

As state before, to better practice time-series prediction and deep-learning algorithm, the model decided is mainly LSTM. LSTM is a variance of RNN (Recurrent Neural Network), which can better handle long sequences, providing better performance with long sequence prediction.

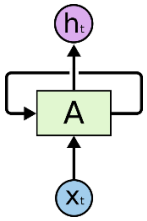

Figure 6 RNN in principle

The most basic RNN has loops. While unfolding the loops, the RNN can be seen as a chain network that can preserve state information from previous inputs. That is the principle why RNN is better for time-series prediction. However, RNN has a general problem for remembering long term inputs. Previous inputs will fade along with time and this cause problem for large sequence prediction. Then LSTM can be used to better solve this issue.

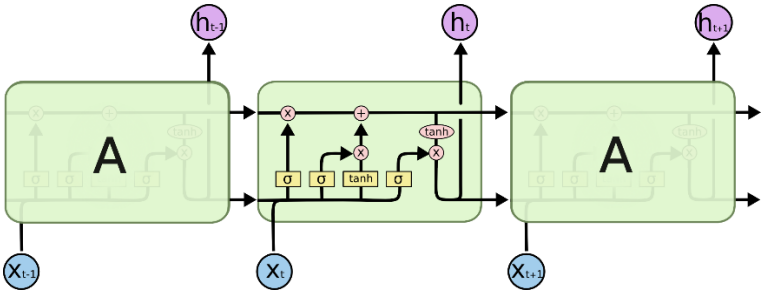

Figure 7 LSTM in principle

Compare with RNN, LSTM has two internal states, namely hidden state and cell state. The hidden state is designed to pass short-term memory, and the cell state is especially for long-term memory. During each stage within the LSTM chain, information is calculated to decide which parts will contribute to long-term memory, and which part will pass through short-term memory. By doing so, LSTM provides excellent results in long series input compared with RNN.

Pytorch with Cuda was chosen to implement the LSTM model. scikit-learn was also used for data pre-processing. As for plotting, Seaborn package was selected.

Benchmark

As it is a Kaggle competition for sales prediction, a proper benchmark is hard to define. On the other hand, the result should at least on the top 50% of the leader board on Kaggle.

The intention of this project is never to copy what is already proved to work well, but to explore possibilities and come up with a special solution that can complete the task. Therefore, the target for the Kaggle competition leader board is not set relatively high, such 10%.

Methodology

Data Preprocessing

NaN Handling

Before any numerical preprocessing, the NaN value distribution was analyzed.

Store	0
DayOfWeek	0
Date	0
Sales	0
Customers	0
Open	0
Promo	0
StateHoliday	0
SchoolHoliday	0
StoreType	0
Assortment	0
CompetitionDistance	2642
CompetitionOpenSinceMonth	323348
CompetitionOpenSinceYear	323348
Promo2	0
Promo2SinceWeek	508031
Promo2SinceYear	508031
PromoInterval	508031

Figure 8 NaN distribution in columns

According to result, all the NaN values are within columns **CompetitionDistance**, **CompetitionOpenSinceMonth**, **CompetitionOpenSinceYear**, **Promo2SinceWeek**, **Promo2SinceYear** and **PromoInterval**. Considering the stability of the model, all the NaN were replaced by the mean value of the corresponding column.

Label Encoding

Within the origin dataset, column **Assortment**, **StoreType**, **PromoInterval**, and **StateHoliday** are provided as a string. Therefore, for following numeric calculation, all the columns above are label encoded with the help of sci-kit learn LabelEncoder class.

For the rest of the columns, standardization scaler will be applied as those columns are all valued information, which should be processed by the scaler.

Standardization

Standardization is further applied to all rest of the columns. The reason performing standardization is that model is “distance-based” value calculation, it’s better to have all the input value within a similar range. By doing so, the model’s weight could have better distribution and provides a more stable result.

To achieve this, MinMaxScaler was used for the most critical column, sales. MinMaxScaler will scale the origin input within the range from 0 to 1. This range is critical as the output of the model will also generate a number from 0 to 1, and inverse transfer back to the real sales number.

The scaler result for this column shown as below:

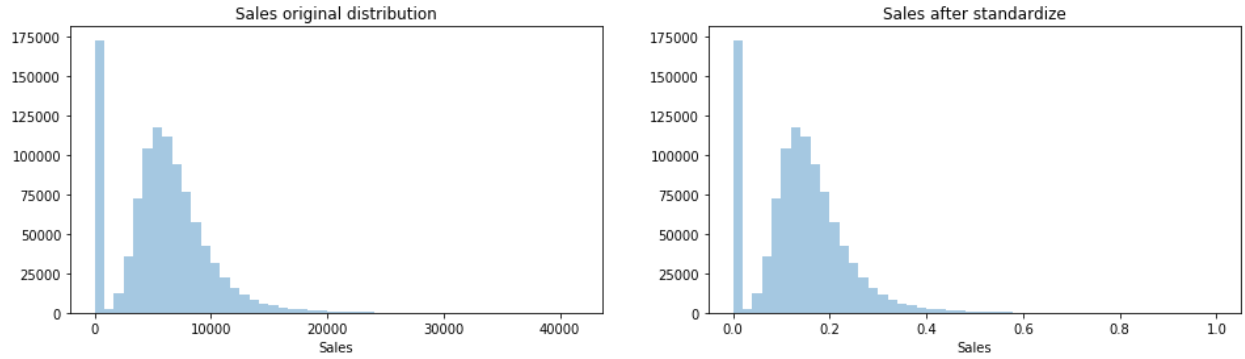

Figure 9 Sales original data (left) with standardized data (right)

Data Used for Model

When performing LSTM prediction, data will feed into the model to train the weights. Obviously, not all the data are necessary to feed into the model. Especially those constant value, for example, StoreType. Those kinds of input will be a constant value during the time-series inputs. From the view of model training, those constants will eventually become a pre-set constant bias and will be learned by the model itself.

Therefore, the decision was made to not input those data to reduce the training dataset. The reason is that the model itself should learn from the dynamic data itself to re-construct the constants. The input data used for model decided is shown as below:

- Sales
- Promo status
- Open status

Implementation

Training and Inference Approach

As the final task is to predict sales for around 800 stores for up to 3 weeks each. The training and inference approach is either to train one model for one store and predict for that individual store or train a master model that can handle any store's prediction task.

Considering the complexity, the master model is preferred. This can heavily reduce the training time for an individual store. Besides, this single model approach is much easier to maintain, compare with the other approach.

Therefore, the decision made was as below:

- A single model will be used to receive inputs from any store to predict for any store
- This model should be trained with data from all the store

Time independent Design

Generally speaking, time-series prediction should train the model with sequenced data and predict with the sequenced result. However, as discussed above, the model designed should be capable of any store inputs. Therefore, the model should be able to receive time-independent inputs and generate a prediction result.

To do so, the inputs should not be the value of a single sale, but a sequence of historical data. And the decided input of the model is as below:

Date	Sales_T0	Sales_T1	Sales_T2	Sales_T3	...	Sales_TN	OPEN_TN-3	...	OPEN_TN+1	PROMO_TN-3	...	PROMO_TN+1
2015-07-31	0.581	0.25	0.555	0.568	...	0.846	1	1	1	0	...	1

where:

- **Date**: not an input data, indicate this input is used for which date's prediction, also notate as T-0D in below
- **SalesDataPart1**: T-30D ~ T-1D sales data, short term sales information
- **SalesDataPart2**: T-370D ~ T-350D historical sales data, long term sales information, used to catch the yearly pattern
- **OpenStatus**: T-6D ~ T-1D store open status
- **PromoStatus**: T-6D ~ T-1D store promotion status
- **CurrentOpenStatus**: T-0D store open status

By doing so, the model can predict any date's sales with a time-independent input sequence.

Batches

As discussed above, the training will use all the data from all the stores. Besides, each input is constructed by multiple values. To have better training performance, the batch input was used.

Moreover, the batch number decided was a relatively large value, 1000. This was considering the memory copy between the Cuda device and CPU. By increasing the batch number, the memory copy overhead between the Cuda device and CPU can minimize.

With the help of time-independent design, the batches were created in a shuffling way, which includes data from all the stores.

Model

Inside the model, at the first stage, 3 LSTM was deployed. All **SalesData** feed into one LSTM, **OpenStatus** and **PromoStatus** feed into the reset of two LSTMs. All three LSTM outputs feed into 2 full connected layers and then generate a single output activates with a sigmoid function. Finally, this sigmoid output is multiplied with the predicted day **CurrentOpenStatus**, and generate the final output.

Besides, to achieve the time-independent design, the hidden state and cell state is ignored. For each inputs, all hidden state and cell state will be initialized as zero, no information is preserved for next inputs.

Detail as below:

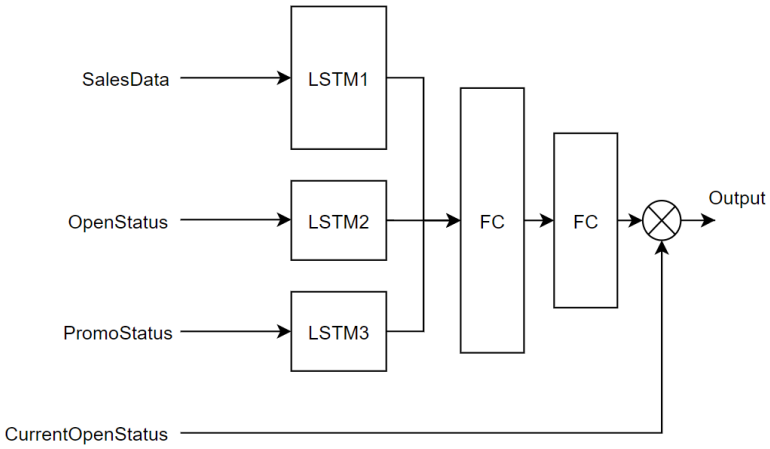

Figure 10 model detail

The reason builds this rather complex model is because this single model will be used to generate predictions for all stores. The data should be enough to avoid over-fitting, and this model will not be recognized as complex also.

Besides, the **CurrentOpenStatus**'s multiplying with full connected network outputs should eliminate the non-linear introduced by store open status, even though this should also learn by the full connected network itself with a large weight value.

The parameters decided for this model as below:

```
LSTMRegressor(
  (lstm): LSTM(51, 30)
  (lstmOpen): LSTM(6, 6)
  (lstmPromo): LSTM(7, 7)
  (dropout): Dropout(p=0.2, inplace=False)
  (fc1): Linear(in_features=43, out_features=1024, bias=True)
  (fc2): Linear(in_features=1024, out_features=1, bias=True)
  (sigmoid): Sigmoid()
)
```

Figure 11 model parameters

Training Model

The validation dataset and training dataset is also generated from original training data with a ratio of 0.2. Due to the benefit of time-independent design, the validation dataset and training dataset is randomly selected from all stores' data.

While training model, the BCELoss function was used to evaluate the training loss and validation loss.

The training was designed to have 20 epochs, each epoch will run through all the training data. The validation is performed every 20 steps, each step contains a batch input, which including 1000 individual training set.

The model with the best validation loss is preserved as a checkpoint after validation. This model will eventually be used for the prediction task.

Results

Model Evaluation and Validation

During the training, the learning curve illustrates as below:

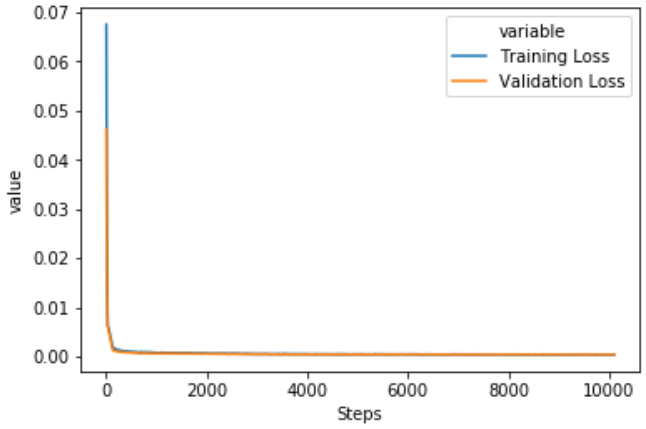

Figure 12 learning curve

According to the learning curve, validation loss is always dropping with training loss, over-fitting has never occurred. However, the dropping rate of the loss is largely decreased after the first few epochs. The loss result from different epochs was as:

Epoch	Training Loss	Validation Loss
1	0.000945	0.000749
2	0.000774	0.000621
3	0.000731	0.000568
4	0.000622	0.000527
5	0.000579	0.000461
…	…	…
10	0.000457	0.000395
…	…	…
15	0.000409	0.000375
…	…	…
20	0.000387	0.000367

Figure 13 training loss and validation loss during training

And the best validation result is checked at Epoch 15, step 7400, with a training loss of 0.000403 and a validation loss of 0.000352.

Prediction Visualization

To better understand the model's performance, tests were taken to verify the prediction accuracy. There were two prediction tests conducted.

- Test1 (day-by-day prediction): Predict T-0D sales, then use the real sales value as the T-0D input, together with the rest of the data to predict T+1D sales. Continue till complete target days prediction.

- Test2 (long-term prediction): Predict T-0D sales, then use the prediction T-0D as input, together with the rest of the data to predict T+1D sales. Continue till complete T+30D sales.

The goal of these two tests was to verify the performance of day-by-day prediction and 30 days long term prediction.

Result for test1 was as below:

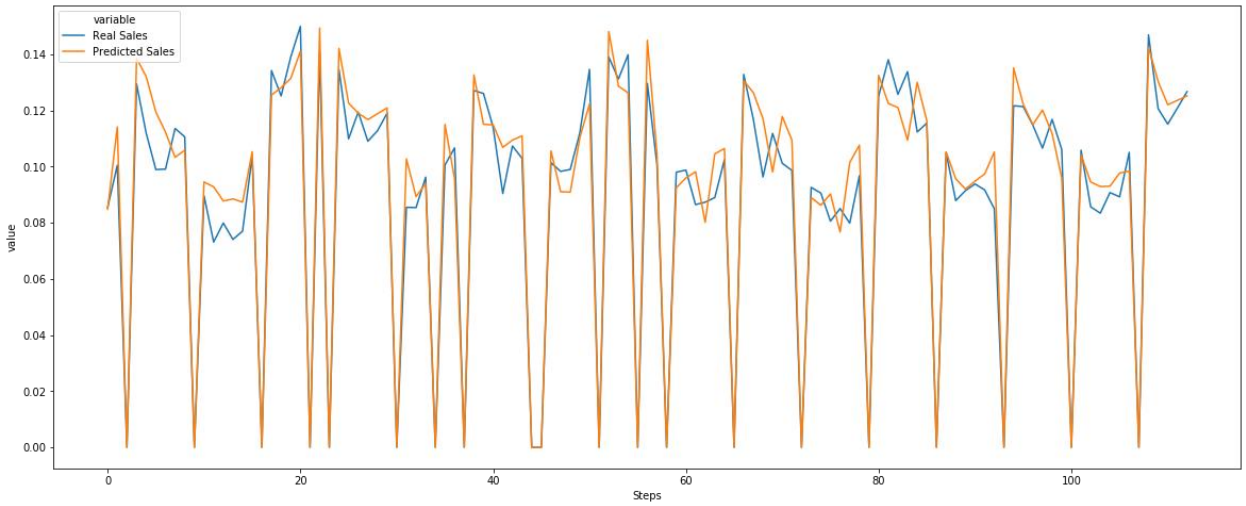

Figure 14 day-by-day prediction result

According to the result, the model can capture most of the trend. Including the peak day after store close, and the trend within a week. More importantly, the average sales is always around the correct sales label.

Result for test2 also shown as below:

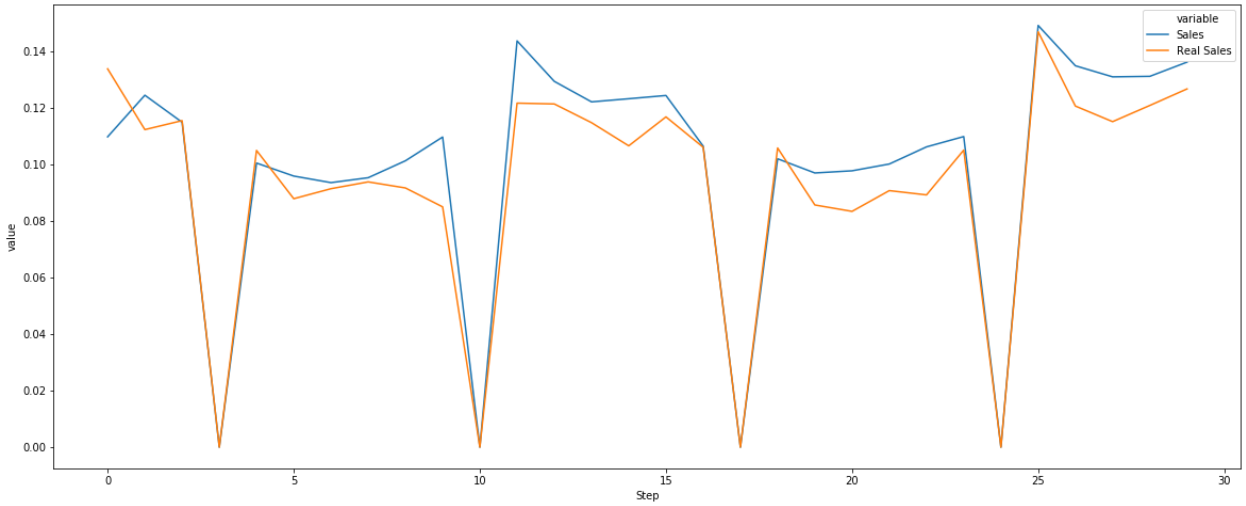

Figure 15 long-term prediction result

Like day-by-day prediction, those critical pattern is captured. However, compare with day-by-day prediction, the long-term prediction result shows some drifting on the far future prediction result. As the drifting amount is under a tolerable range, this is considered acceptable.

Besides, as Kaggle will eventually use RMSPE to evaluate performance, RMSPE was also calculated for the monthly prediction. The calculate result shown as below:

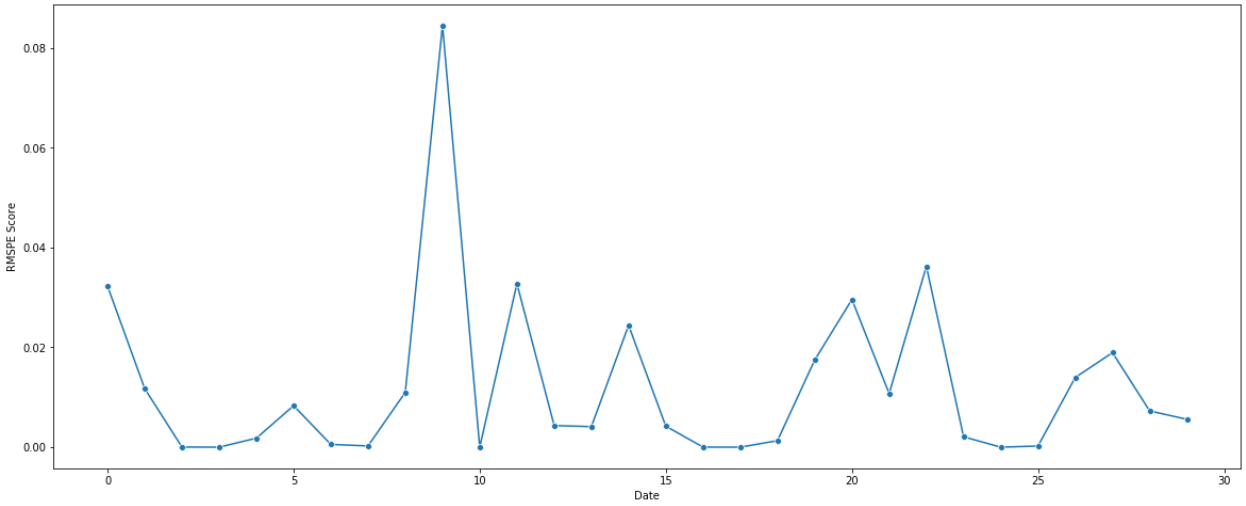

Figure 16 RMSPE result for monthly prediction

According to the result, the RMSPE result is mostly below 0.02, and this result is considered acceptable.

Finally, the model and prediction also submit to Kaggle competition for performance review, the final performance for the submitted prediction result are 0.158. Recorded as below:

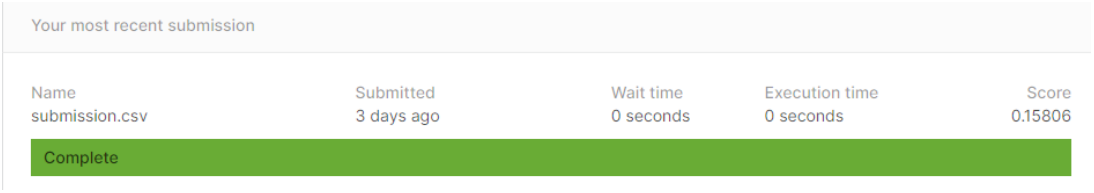

Figure 17 Kaggle submission record

Conclusion

According to the test result, the model trained to capture the general patterns identified in the data exploration section. The test result also shows that the prediction result is good enough.

After submitting the result to Kaggle, the result score is 0.168, which is around 50% of the competition leader board. Therefore, the model is a good practice but still a lot of room to improve.

Improvement

Restricted by the time-independent design, time sequences are format as inputs sequence. Therefore, even the LSTM model was used, the preserved information is still limited, only less than 60 history sales data were used for a single prediction.

To overcome this issue, one model one store approach can further adapt to use time-dependence sequence input. The hidden state and cell state can also be used to preserve the information.

Currently, most of the store's information is ignored. However, that information might provide information that does not include in the training data. The future model may further test the effect of adding that information.

Another limitation current model has is it used a sigmoid activation function as output. As a result, the output range is limited to (0, 1). To align with this activation function, sales data was scaled with MinMaxScaler, which will scale data into the range of (0, 1). This, however, will restrict the prediction result. Meaning the prediction can not generate any data exceeding historical minimum value and maximum value. To overcome this, a linear activation function should be used, and the scaler should further update to scaler without range limitation, such as StandardScaler.

Finally, for each prediction, only the relevant store's status inputs to the model. However, other store's information may also influence the sales figures. The future model may also try to add other store's status to monitor such correlation.