# TIME SERIES PREDICTION ON SALES OF WALMART

*“In this paper, we implemented ARIMA, Single- and Multi- layer LSTM, and Light GBM on sales data from Walmart.”*

## 1. ARIMA

For this part, we simply use an ARIMA model to establish a baseline for forecasting Walmart sales. We firstly differentiate the series. By Dickey-Fuller test, we make sure that the sequence has been stationary. After dividing the data into training and test sets, we use *pm.auto_arima* function to train the model, and select ARIMA(3,1,2) model which has the lowest AIC.

Then doing the Ljung-Box statistical test, we can find the residual is very closed to the white noise; thus the model is adequate. So we use ARIMA(3,1,2) model to make predictions on the entire dataset. As can be seen from the figures below, the basic trend of the products' sales volume can basically be predicted using the ARIMA model. For the test set, the ability to predict lower sales volume is stronger.

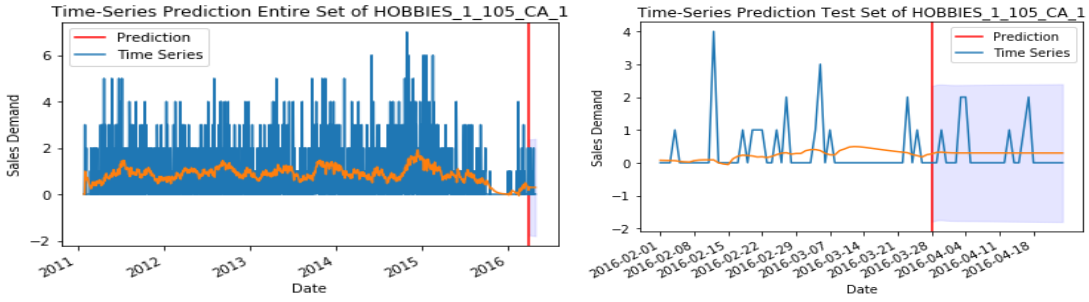


*Figure 1 : ARIMA ——Entire Set and Test Set Prediction*

Meanwhile, we use RSSME to measure the accuracy of the point forecasts. This indicator is often used as a standard for measuring the prediction results of machine learning models.

By defining $\boldsymbol{RMSSE} = \sqrt{\frac{1}{h}\frac{\sum_{t=n+1}^{n+h}(Y_t-\hat{Y}_t)^2}{\frac{1}{n-1}\sum_{t=2}^{n}(Y_t-Y_{t-1})^2}}$ , we can calculate that RMSSE equals to 0.4358. In addition, we can also know that the insample RMSE is 1.0546 and outsample RMSE is 0.6571.

For ARIMA models, we generate 1000 observations, and calculated the MSE and RMSSE of in-and- out sample.

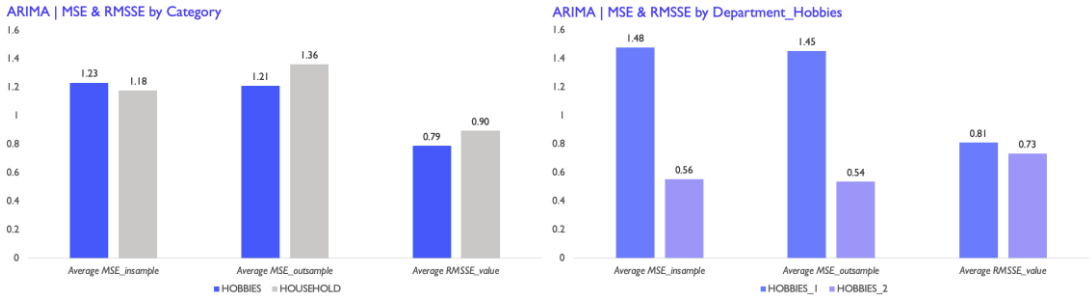


*Figure 2 : ARIMA ——Comparison of 1000 Observations Prediction*

Shown as figures above, in terms of categories, the average MSE and average RMSSE of outsample and value of Household is higher than those of Hobbies. But for insample, Hobbies has lower MSE than Household does.

Looking into the Hobbies category, it is clearly that, compare with Hobbies 2, Hobbies 1 has lower MSE in- and-out sample as well as RMSSE.

# 2. LSTM Network

## 2.1 Theoretical Basis

Long Short Term Memory network – usually just called "LSTM" – is a special kind of RNN, which is explicitly designed to avoid the long-term dependency problem. Based on the ordinary RNN, LSTM adds memory units to each neural unit in the hidden layer, so that the memory information in the time series can be controlled. Although LSTM has the chain like RNN above, the repeating module has a different structure. Instead of having a single neural network layer, there are three, interacting in a very special way. So each time passing between the units in the hidden layer, information need to pass through three kinds of controllable gates (forget gate, input gate, output gate), which can control the memory and forgetting of the previous and current information. Therefore, the network has a long-term memory function. And the structure of LSTM is shown below:

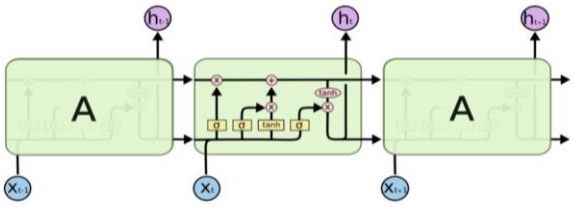


*Figure 3 : The Structure of LSTM*

As can be seen from the figure above, the core of LSTMs is cell state, which is represented by a horizontal line through the cell at the top of the memory block. The cell state is like a conveyor belt. It penetrates the entire cell but has only a few branches, which ensures that the information flows through the entire RNNs unchanged. Then, the LSTM network can carefully delete or add information to the cell state through a structure called a gate. The gate is a combination of a sigmoid layer and a dot product operation. LSTM designs two such gates to control the amount of information in the memory unit state c: one is the forget gate. It is the "memory defect", which determines how much "memory" of the unit state at the previous moment can be retained until the current moment; the other is the input gate, which determines how much of the input at the current moment is saved to the unit state. In fact, for the convenience of expression, many documents have added a gate, called the candidate gate, which controls the proportion of the fusion of "historical" information and "present" stimuli. Finally, LSTM designed an output gate to control how much information is output from the cell state.

## 2.2 Train, Evaluate and Predict

In this part we will construct the single-layer and double-layer LSTM, and here is the flow chart of LSTM nets:

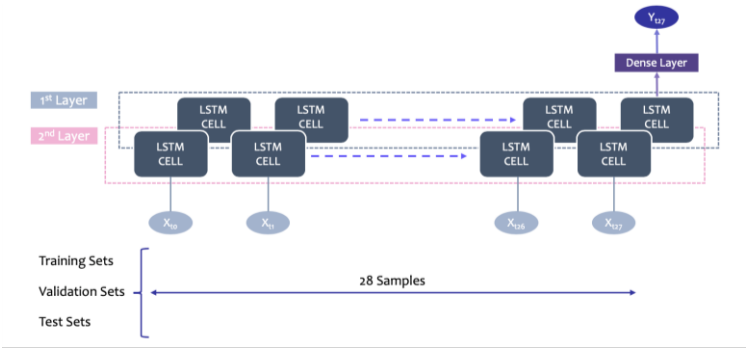


*Figure 4 : The Flow Chart of LSTM Nets*

At first, we create the simple Pytorch LSTM model which just has one LST layer and one dense input layer. After spliting the normalized data into training validation, and testing set , we define a class LSTM, which inherits from nn.Module class of the PyTorch library and create an object of the LSTM() class. Then we set criterion, optimizer and scheduler that can improve network performance by reducing network learning rate. Next, we train the model on training data for 100 epochs, evaluate on validation data (print the loss after every 25 epochs), and make prediction on the entire data set. So we can draw the predicted time series of this product below. The predicted value of this product sales fluctuate basically between 1 and 2. The trend of forecasted sales increase or decrease is also approximately the same as the actual value, although some extreme values cannot be predicted accurately by this model. On the test set, the predicted value is basically stable between 1 and 1.5. In addition, the insample RMSE is 1.2076 and outsample RMSE is 1.1645. we can get RMSSE equal to 0.7755.

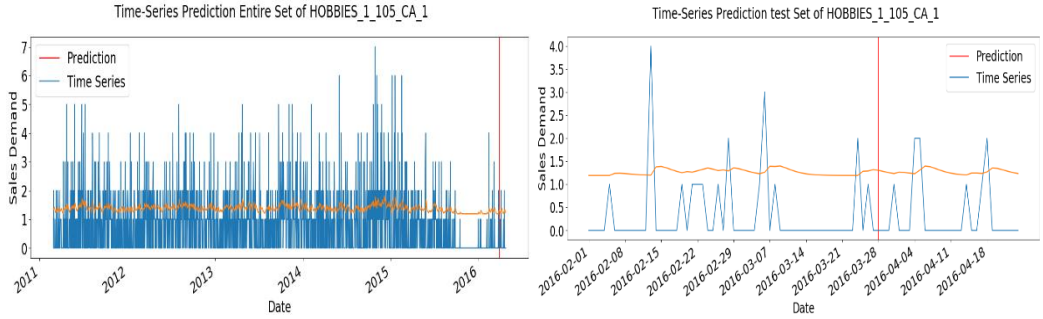


*Figure 5 : Single-layer LSTM——Entire Set and Test Set Prediction*

For single-layer LSTM models, we also generate 1000 observations, calculated the MSE and RMSSE of in-and-out sample, and the result are shown above.

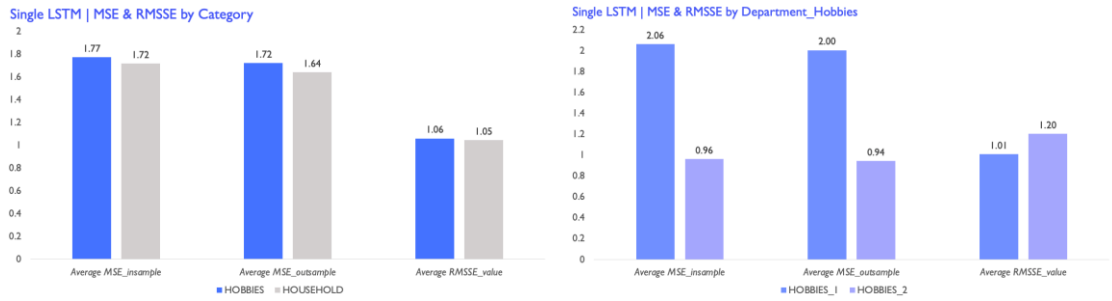


*Figure 6 : Single-layer LSTM——Comparison of 1000 Observations Prediction*

In terms of categories, the average MSE and average RMSSE of Hobbies are all higher than those of Household.

Looking into the Hobbies category by department, for in- and- out sample, Hobbies 1 has higher MSE compare to Hobbies 2. As for the RMSSE, Hobbies 2 has higher RMSSE than Hobbies 1 does.

However, when we use a single RNN Cell's call function to do calculations, we are only one step forward in sequence time. In view of the limited ability of single-layer RNN, we build a two layers LSTM model. We can get another predicted time series plot again by repeating the previous steps. It can be seen from the figure that the range of the predicted value on entire set basically fluctuates between 0.5 and 1.5. On the test set, the predicted value is basically fluctaute between 0.5 and 1. The model's ability to predict the minimum value is enhanced, and the fluctuation of the predicted value is closer to that of the true value. Again, we can get the insample RMSE is 1.0809 and outsample RMSE is 0.7895. we can get RMSSE equal to 0.5249, which is much lower than the value in one layer LSMT model.

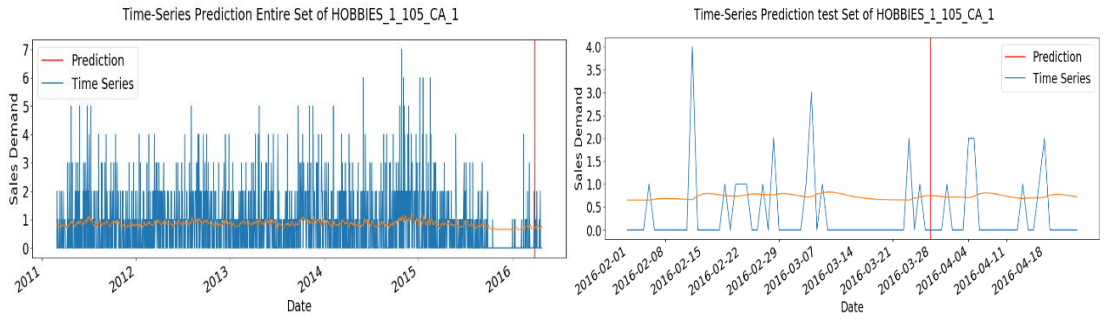


Figure 7: Multi-layer LSTM——Entire Set and Test Set Prediction

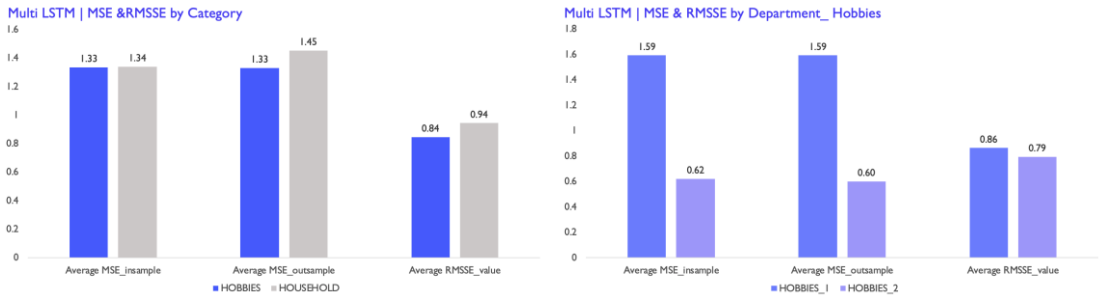


*Figure 8: Multi-layer LSTM ——Comparison of 1000 Observations Prediction*

For multi-layer LSTM models, the MSE and RMSSE of in-and- out sample of the 1000 observations are shown above.

In terms of categories, the average MSE and average RMSSE of Household are all higher than those of Hobbies.

Looking into the Hobbies category by department, for in- and- out sample, Hobbies 1 has higher MSE as well as RMSSE compare to Hobbies 2.

# 3. Light GBM

## 3.1 Theory and algorithm

Light GBM, proposed by Microsoft, is mainly used to solve the problems encountered by GBDT in massive data, so that it can be better and faster used in practice. From the name of Light GBM, we can see that it is a light gradient elevator (GBM), which has the characteristics of fast training speed and low memory occupation

compared with XGBoost. Considering the large amount of data in this paper, we choose LGBM model instead of XGBoost. Compared with XGBoost, LGBM algorithm has three aspects improvement: Gradient-based One-Side Sampling, histogram algorithm and Exclusive Feature Bundling algorithm.

**Gradient-based One-Side Sampling**

**Algorithm 2:** Gradient-based One-Side Sampling

**Input**: $I$: training data, $d$: iterations
**Input**: $a$: sampling ratio of large gradient data
**Input**: $b$: sampling ratio of small gradient data
**Input**: $loss$: loss function, $L$: weak learner
models ← {}, fact ← $\frac{1-a}{b}$
topN ← a × len($I$) , randN ← b × len($I$)
**for** $i = 1$ ***to*** $d$ **do**
  preds ← models.predict($I$)
  g ← $loss$($I$, preds), w ← {1,1,...}
  sorted ← GetSortedIndices(abs(g))
  topSet ← sorted[1:topN]
  randSet ← RandomPick(sorted[topN:len(I)], randN)
  usedSet ← topSet + randSet
  w[randSet] × = fact ▷ Assign weight $fact$ to the small gradient data.
  newModel ← L($I$[usedSet], − g[usedSet], w[usedSet])
  models.append(newModel)

The gradient size of GBDT algorithm can reflect the weight of samples. The smaller the gradient, the better the model fitting. The gradient based one side sampling algorithm (GOSS) uses this information to sample, reducing a large number of samples with small gradient. In the next calculation, only the samples with high gradient need to be paid attention to, greatly reducing the calculation amount. The specific algorithm is as follows:

We can see that GOSS sorts the samples based on the absolute value of the gradient in advance (without saving the result after sorting), and then gets the samples with a large gradient in the first a% and b% of the total samples. When calculating the gain, we can multiply (1-a)/b to enlarge the weight of samples with small gradient. On the one hand, the algorithm pays more attention to the samples with insufficient training. On the other hand, it multiplies the weight to prevent the sampling from causing too much influence on the distribution of original data.

**Histogram algorithm**

The basic idea of histogram algorithm is to discretize continuous features into k discrete features and construct a histogram with width of K for statistical information. Using histogram algorithm, we can find the best splitting point by traversing K bin instead of traversing data.

When constructing the histogram of leaf node, we can also reduce the computation by half by subtracting the histogram of parent node from that of adjacent leaf node. In the practical operation, we can calculate the leaf nodes with small histogram first, and then use the histogram as the difference to get the leaf nodes with large histogram.

**Exclusive Feature Bundling**

**Algorithm 3:** Greedy Bundling

**Input**: $F$: features, $K$: max conflict count
Construct graph $G$
searchOrder ← $G$.sortByDegree()
bundles ← {}, bundlesConflict ← {}
**for** $i$ ***in*** *searchOrder* **do**
  needNew ← True
  **for** $j = 1$ ***to*** *len(bundles)* **do**
    cnt ← ConflictCnt(bundles[j],$F$[i])
    **if** *cnt + bundlesConflict[i]* $\leq K$ **then**
      bundles[j].add($F$[i]), needNew ← False
      break
  **if** *needNew* **then**
    Add $F[i]$ as a new bundle to $bundles$
**Output**: $bundles$

High dimensional features are often sparse, and features may be mutually exclusive. For example, two features are not taken as non-zero values at the same time. If two features are not completely mutually exclusive (for example, if only a part of the case is not taken as non-zero values at the same time), the mutual exclusion rate can be used to express the degree of mutual exclusion. The exclusive feature bundling (EFB) algorithm points out that if some features are fused and bound, the number of features can be reduced. So here comes a question: Which features can be bound together?

EFB algorithm constructs a weighted undirected graph based on the relationship between features and transforms it into graph coloring algorithm. We know that graph

coloring is a NP-Hard problem, so we use greedy algorithm to get the approximate solution.

If the feature dimension reaches million level, the calculation amount will be very large. In order to improve efficiency, we propose a faster solution. The strategy of sorting according to node degree is changed to sorting according to non-zero value technology, because the more non-zero value, the greater the probability of mutual exclusion.

## 3.2 Light GBM implementation and results

During this part, we'll introduce the establishment of Light GBM, show the algorithm result and analyze the contribution of different features. We first merge the different datasets and also ignore some columns because Light GBM is a quite fast model and we don't want to waste the algorithm performance on the useless data columns. In order to find more useful data features in our Light GBM algorithm, we create some features in demand, price and time datasets. For example, we make different time interval like 7 days, 14 days and 28 days to calculate the demand mean value and price change percentage over these time lags. These features we create are evolved from the existing important features which may have great contribution in Light GBM algorithm. We'll analyze the feature contribution and see whether the created features have valuable influence in Light GBM.

In the algorithm training and test section, we use the date "2016-03-01" as the node to split the data as the training set and test set. For timely purpose of our algorithm, we set the number of folds in the Light GBM algorithm to be 3. Besides, there are many different parameters in the Light GBM and some of them are very important to the performance of algorithm. We have tried many values in these parameters. In the final version, we set the number of leaves (num_leaves) to be 400 because the large num_leaves can make the algorithm search is more refined. The learning rate we set equals to 0.05 because Light GBM is a fast algorithm and we don't need a very high learning rate. And the rest of parameters, most of them we set them around their default parameter values.

We use the selected features and parameters to run the Light GBM algorithm prediction section and calculate the "training's rmse", "valid_1's rmse" and "val rmse score" which are the performance evaluation standard in this competition. The following chart is the result of "val rmse score" in all the 3 folds. It shows that the rmse score is quite small and also shows the great efficient of our Light GBM algorithm.

| **Fold No.** | **1** | **2** | **3** | **mean** |
|---|---|---|---|---|
| **val rmse score** | 2.4492 | 2.3064 | 2.237 | 2.3309 |

*Table 1 : Light GBM —— rmse score over 3 folds*

As we have created many new features in the dataset, we need to evaluate the feature importance and analyze these features. We draw the following plot to show all feature importance over 3 folds in average. The top 4 contribution features are the basic features in the origin datasets. It shows that the features of demand rolling mean value in different time lag (rolling_mean_t7, rolling_mean_t14, rolling_mean_t30, rolling_mean_t90, rolling_mean_t180) have great contribution in our Light GBM algorithm. Also, the rolling standard deviation of demand has much influence in the performance of algorithm. So, it seems like the created features from demand dataset are valuable in Light GBM algorithm prediction and for the further research, we may focus more on the demand dataset and dig in more useful feature information.

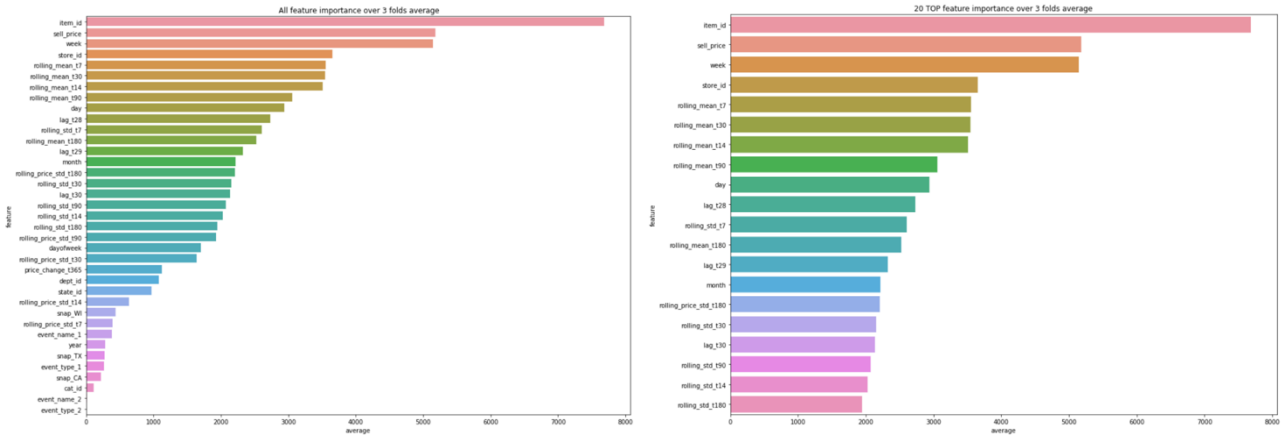


*Figure 9 : All feature importance & 20 TOP feature importance over 3 folds average*

We also present the plot of the top 20 feature contribution in the 3 folds. This allows us to select useful features more clearly, and we can eliminate features that contribute less, in order to improve the efficiency and performance of the algorithm.

# 4. Conclusion

In this paper, we introduced three models to estimate and predict the sales volumn data of Walmart- ARIMA, Single- and Multi- layer LSTM, and Light GBM. The result shows that LSTM and Light GBM algorithm can well predict the trend of sales data.

For each model, we calculate RMSE in the prediction section; and for all models we built, RMSE are all quite small, which indicates these algorithms all have good performance in predicting sales data of Walmart.

In Light GBM section, we illustrated the significance of different features, and we focus more on those features that show great contributions, which could have greater impact on our prediction models.

The Members in Our group & Workload Distribution

| Section | Workload | Member |
|---|---|---|
| **ARIMA** | • Model Construction and prediction<br>• Sample analysis<br>• Report writing and slides making | ZHONG, Jing<br>LIU, Ruiyan<br>LI, Han |
| **Single- Layer LSTM** | • Model Construction and prediction<br>• Sample analysis<br>• Report writing and slides making | ZHONG, Jing<br>LIU, Ruiyan<br>LI, Han |
| **Multi- Layer LSTM** | • Model Construction and prediction<br>• Sample analysis<br>• Report writing and slides making | ZHONG, Jing<br>LIU, Ruiyan<br>LI, Han |
| **Light GBM** | • Model Construction and prediction<br>• Sample analysis<br>• Report writing and slides making | LI, Yijin<br>ZHANG, Xuanyu<br>ZHANG, Tianyu |
| **Conclusion** | • Analysis<br>• Report writing and slides making | LI, Yijin<br>LI, Han |
| **Presentation** | • Part I<br>• Part II | LIU, Ruiyan<br>ZHANG, Tianyu |

Links to Files:

Code: https://github.com/Ryann729/MAFS-6010U/tree/master/code_ppt

Presentation: https://youtu.be/Y9a8yLi39w4