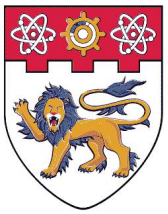


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

**AY2021/2022**

# MH8111 Analytics Software I

Group Project Report

*Predicting Future Sales*

# Table of Contents

## 1. Introduction

In this project, we will be exploring the dataset consisting daily sales data of various items from a list of shops. From the data provided, we will attempt to predict the total amount of every item sold in every shop for the next month.

Being able to predict the future sales figures is useful as it allows for planning of strategies to increase the sales of a company. For example, if a product is identified as more popular than others, decisions can be made, such as to increase the price of the product to generate more revenue or place popular items at shops with higher sales.

## 2. Dataset Description

The sales dataset consists of 4 tables as listed below. There are 59 shops, 83 item categories and 22169 items in the dataset.

| Table Name | Description of table | Number of attributes | Attribute description |
|---|---|---|---|
| ***Sale*** | Daily historical data from January 2013 to October 2015 | 6 | Date, Month number from 0 to 33, Shop ID, Item ID, Item price, Number of item sold |
| ***Items*** | Supplemental information about the items/products | 3 | Item name, Item ID, Category ID |
| ***Items_categories*** | Supplemental information about the item categories | 2 | Category name, Category ID |
| ***Shops*** | Supplemental information about the shops | 2 | Shop name, Shop ID |

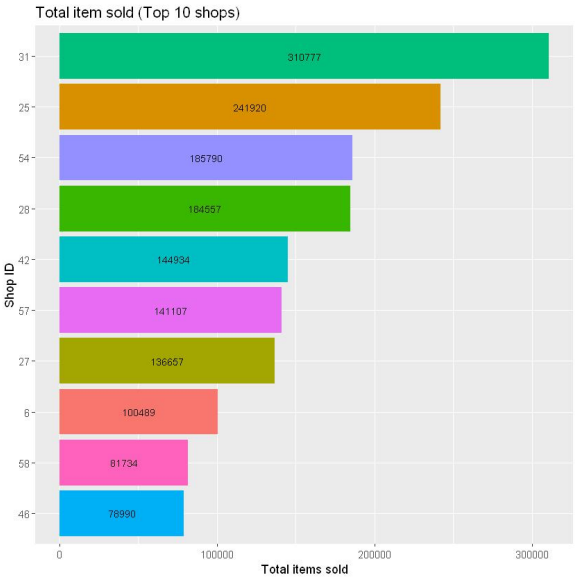
*Table 1: Overview of dataset*

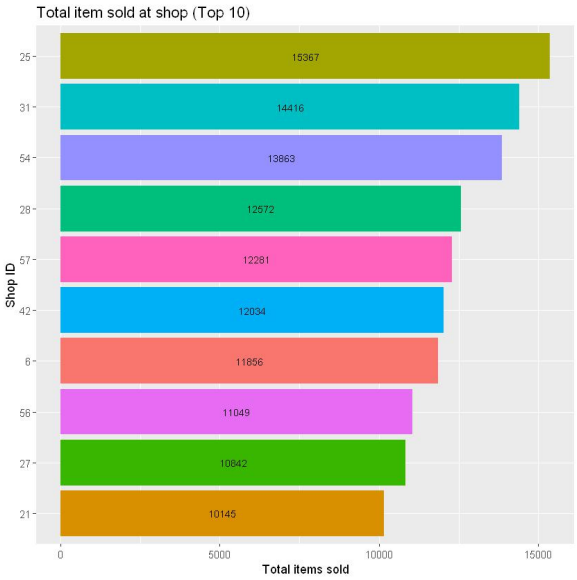
## 3. Exploratory Data Analysis

Using R code on Jupyter Notebook, the exploratory data analysis was conducted

### 3.1 Total items sold at shops

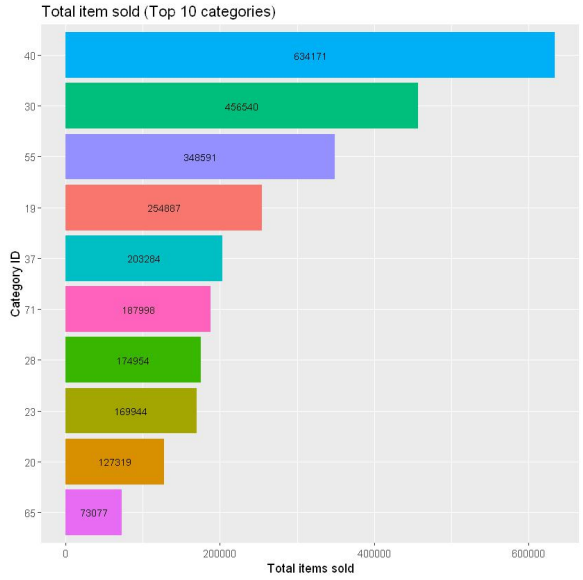
Since we are interested in the total amount of items sold per shop, we can look at the top 10 shops with the highest number of items sold. We also look at the 10 shops that sell the most items. We can see a strong correlation between them, with the 6 shops (25, 31, 54, 28, 57, 42) that sell the most items also having the top 6 total number of items sold.

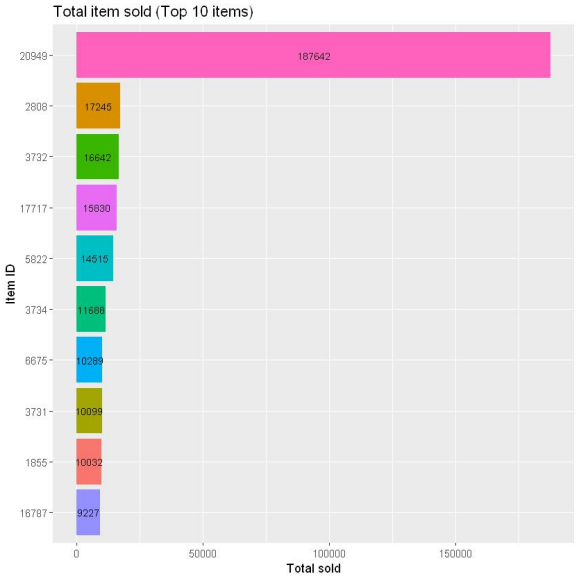


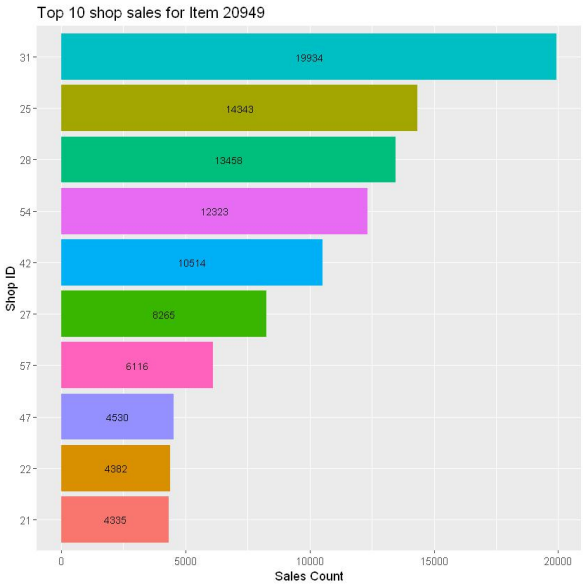

## 3.2 Most popular categories and items

The top 10 categories with the most items sold, and top 10 items sold are as seen below



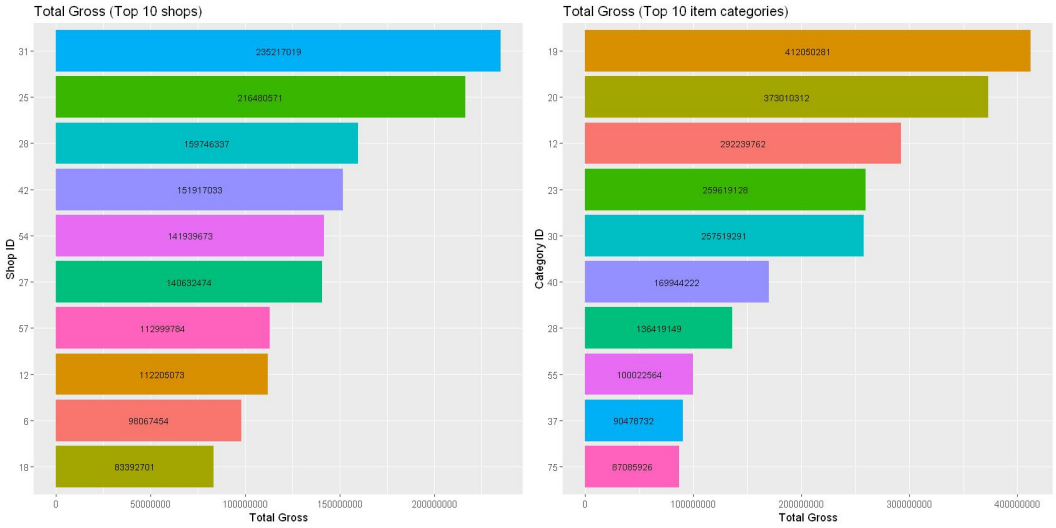



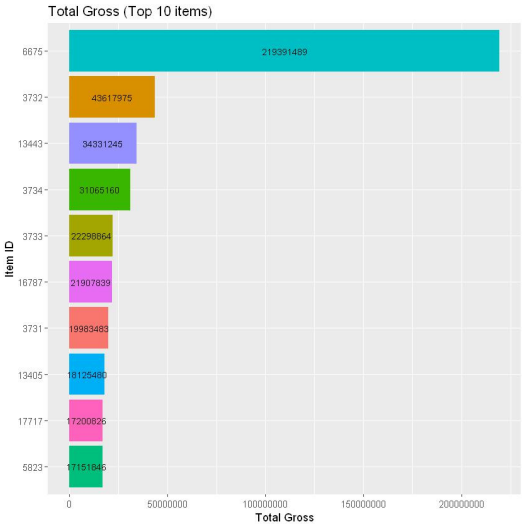

We observe that item 20949 is very popular, with over 10 times the number sold compared to the next most popular item. On further examination, a total of 53 out of the 59 shops sells item 20949. We explore the top 10 shops with the highest sales of item 20949. The top 10 shops with highest sales of item 20949 is very similar to the top 10 shops with highest number of items sold.




## 3.3 Top grossing category and item

Next, we explore the top grossing shops, items and categories.



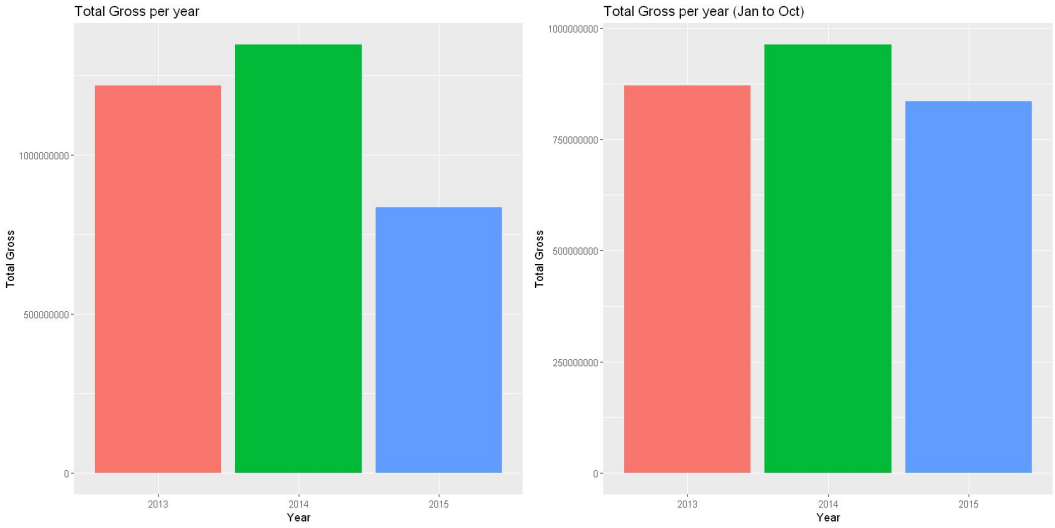
Like the top selling item, the top grossing item is much larger than the next top grossing item, with over 5 times the amount grossed. Item 6675 is sold at 52 of the 59 shops, with the price ranging from 14992 to 28990, which is more than the majority of all items prices (outlier in boxplot of item prices).

The top 10 grossing items are listed in the table below. It is worth noting that the top 7 grossing items belong in the top 5 grossing categories.

| Item_ID | Item | Item_category_id (Category gross ranking) |
|---|---|---|
| 6675 | Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1108A/B01) | 12 (3) |
| 3732 | Grand Theft Auto V [PS3] | 19 (1) |
| 13443 | Sony PS4 (500 Gb) Black (CUH-1108A/B01) + Grand Theft Auto V | 12 (3) |
| 3734 | Grand Theft Auto V [Xbox 360] | 23 (4) |
| 3733 | Grand Theft Auto V [PS4] | 20 (2) |
| 16787 | | 19 (1) |
| 3731 | Grand Theft Auto V [PC] | 30 (5) |
| 13405 | Microsoft Xbox One 500GB (Day One Edition Green Box)" + "FIFA 15" + "Forza Motorsport 5" | 16 |
| 17717 | | 79 |
| 5823 | Playstation Store | 35 |

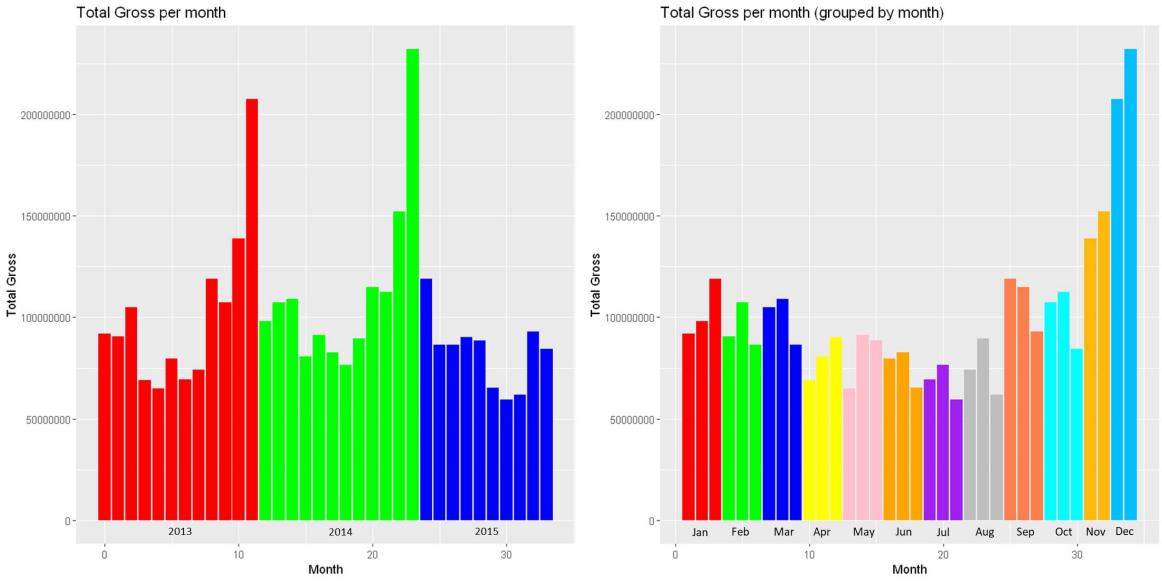
### 3.4 Total gross by Month and year

Finally, we can look at the total amount grossed by year. As the dataset is from Jan 2013 to Oct 2015, we also look at the amount gross from Jan to Oct for 2013 to 2015 for a better comparison.




In general, there is an increase in amount gross from 2013 to 2014 followed by a drop in 2015.

Comparing the month on month figures, we observe the months close to the end of the year generated more total gross as compared to the months in the middle of the year.



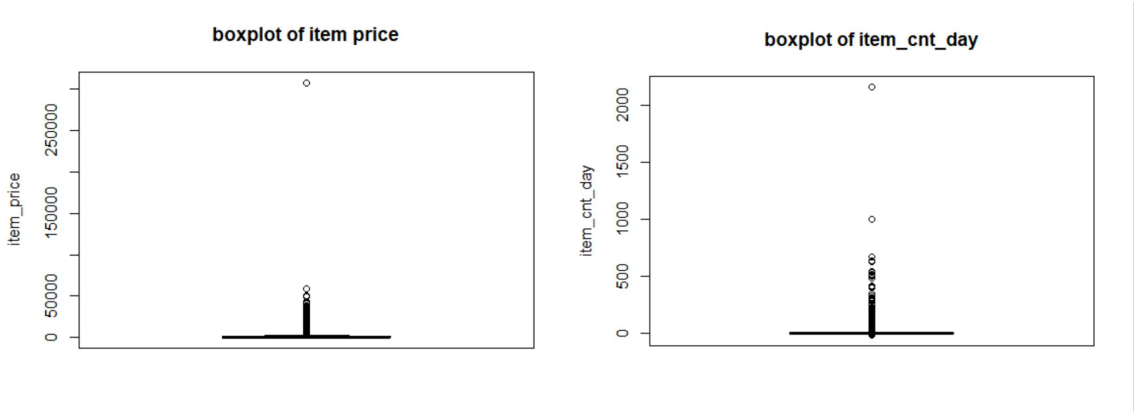

## 4. Data Pre-processing and Feature exploration

4.1 Train data set

*4.1.1 dealing with missing value*

After summarizing the training data set, no missing value was observed.

*4.1.2 dealing with outliers*




Thus, we set the limits to exclude outliers:

item_price>=0 and item_price<=5000 and item_cnt_day>=0 and item_cnt_day<=1000

*4.1.3 Calculate monthly sales for each shop-item group*

According to the submission sample, we need to predict monthly sales rather than daily sales, so we need to calculate the monthly sales of every shop-item group.

Moreover, in a certain month, there will be several different prices for the same item in the same shop, so we use the mean of item price to replace the item price, ensuring each item in each shop in one month will only show once.

Besides, in order to construct features for prediction, we need to merge the useful variables from other data sets, like item category.

Processed Training Data:

| | item_id | shop_id | date_block_num | month | item_cnt_month | item_price | item_category_id |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 54 | 20 | 09 | 1 | 58 | 40 |
| 2 | 1 | 55 | 15 | 04 | 2 | 4490 | 76 |
| 3 | 1 | 55 | 20 | 09 | 1 | 4490 | 76 |
| 4 | 1 | 55 | 21 | 10 | 1 | 4490 | 76 |
| 5 | 1 | 55 | 18 | 07 | 1 | 4490 | 76 |
| 6 | 1 | 55 | 19 | 08 | 1 | 4490 | 76 |

4.2 Test data set

*4.2.1 Analysis test data set*

Compared with training data set, we observe that:

363 kinds of items did not appear in the training set, so 363*42=15,246 kinds of items-shop group have no data in training data, accounting for about 7%.

87,550 kinds of item-shop combinations are only items that have appeared, and no combination has ever appeared, which is about 42% of the test data set.

111,404 kinds of item-shop combinations are present completely in the training set, accounting for about 51%.

*4.2.2 fill the columns like train data set for test data set*

The given test data set only includes shop _ id, item _ id, so we need to fill the rest of variable in train data set for future prediction.

Date_block_num in test data set is 34, and month is November (11), set the item_cnt_month as 0 for future process.

Item price:

The first method: using 34th item price of shop-item group

However, only 28672 shop-item groups have sales in 34th month, so this method is unavailable.

The second method: using historical shop-item price as item price of 35th month, and the price of the rest items are replaced by item average price, and the rest will be replaced by average category price.

Processed test data:

| | item_category_id | item_id | shop_id | date_block_num | month | item_cnt_month | item_price |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 5441 | 12 | 34 | 11 | 0 | 87.8 |
| 2 | 0 | 5441 | 3 | 34 | 11 | 0 | 87.8 |
| 3 | 0 | 5441 | 7 | 34 | 11 | 0 | 87.8 |
| 4 | 0 | 5441 | 47 | 34 | 11 | 0 | 87.8 |
| 5 | 0 | 5441 | 26 | 34 | 11 | 0 | 87.8 |
| 6 | 0 | 5441 | 42 | 34 | 11 | 0 | 87.8 |

4.3 feature exploration

The provided features: shop _id, item _id, item_category_id, date_block_num, month, item_price.

The above features are obviously not enough to support a reliable prediction, so we need to create some new features for prediction.

The main idea of creating new features is to use self-defined lag function to transform historical data as the new feature of next month. For example, we can use the data of last month as the feature of this month.

According to shop, item and item category, we have created some features:

- Item_cnt_month _lag_[i]: The sales of the combination of shop and item in the past i month.
- Item_single_sale_lag_[i]: The average sales of all items themselves, without considering shop, in the past i month.
- Shop_sales_lag[i]: The average sales of all shops themselves, without considering items, in the past i month.
- Category_sale_lag[i]: The average sales of all item categories in the past i month.
- Shop_category_sale_lag[i]: The sales of the combination of shop and item category in the past i month.
- Cum_sales_3/6_month: The cumulative sales of past 3 and 6 months of the combination of shop and item.
- Avg_sales_3/6_month: The average sales of past 3 and 6 months of the combination of shop and item.
- Shop _item _price: The average item price for the combination of shops and items.
- Category _price: The average item price for item categories.
- Item _single _price: The average item price for items themselves.

At first, we determined that i = 1,2,3,6,12, but the result was not good because of too much features. So according to the feature importance of Xgboost model, we determined that i = 1,3,6 and also conducted feature reduction. Finally, we obtain the below features to do prediction.

- Item_cnt_month _lag_[i]: The sales of the combination of shop and item in the past i month.
- Item_single_sale_lag_[i]: The average sales of all items themselves, without considering shop, in the past i month.
- Shop_sales_lag[i]: The average sales of all shops themselves, without considering items, in the past i month.
- Category_sale_lag[i]: The average sales of all item categories in the past i month.

- Shop_category_sale_lag[i]: The sales of the combination of shop and item category in the past i month.
- Shop _item _price: The average item price for the combination of shops and items.
- Category _price:  The average item price for item categories.
- Item _single _price: The average item price for items themselves.

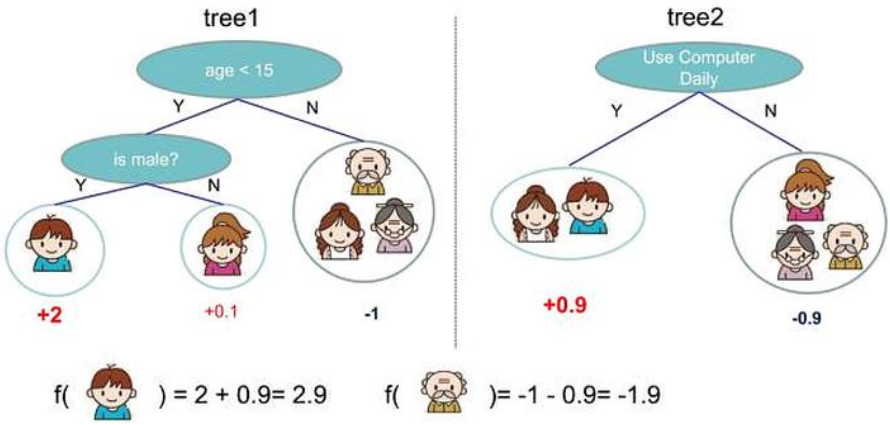
## 5. XGBoost

5.1 Model description

XGBoost is a universal Tree Boosting algorithm, one of which is Gradient Boosting Decision Tree (GBDT). Also known as MART(Multiple Additive Regression Tree).

The principle of XGBoost is that a tree is first trained with the training set and the truth value of the sample (i.e. the standard answer), and then the tree is used to predict the training set to get the predicted value of each sample. Since the predicted value and the truth value have deviation, the "residual" can be obtained by subtracting. Then train the second tree to use the residuals as the standard answer instead of the truth value. After the two trees are trained, the residuals of each sample can be obtained again, and then the third tree can be further trained, and so on. The total number of trees can be manually specified, or training can be stopped by monitoring certain metrics (such as errors on the validation set).

A diagram using two trees to predict whether a person likes a computer game is shown below, with the predicted values of the boy and the old man being the sum of the predicted values of the two trees.




5.2 Parameter introduction and adjustment

*5.2.1 Parameter introduction*

The parameters used in our model are listed below.

**Objective:** Define learning tasks and corresponding learning objectives[ default=reg:linear ].

**n-rounds:** This refers to the number of iterations of promotion, that is, how many base models are generated.

**eta:** By reducing the weight of each step, the robustness of the model can be improved [default 0.3].

**max_depth:** Control the maximum depth of the tree. The greater the max_depth, the more specific and local samples the model learns. [default 6]

**subsample :** Control the rate of random sampling for each tree. By decreasing the value of this **parameter, the algorithm becomes more conservative and avoids overfitting. [default 1]**

**colsample_bytree:** Control the proportion of samples taken from the number of columns at each split of each level of the tree. [default 1]

**alpha/lambda:** The L1 regularization term of the weight./The L2 regularized term of the weight.

*5.2.2 Parameter Adjustment*

We notice that in this Kaggle competition grade is based on RMSE of our prediction. So we

$$\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

regard RMSE of our test set as Benchmark to adjust parameter. RMSE is calculated as follows

In the adjustment procedure, we take grid search method to find the best parameter. As we do not find such function in R, we use a loop function to realize it.

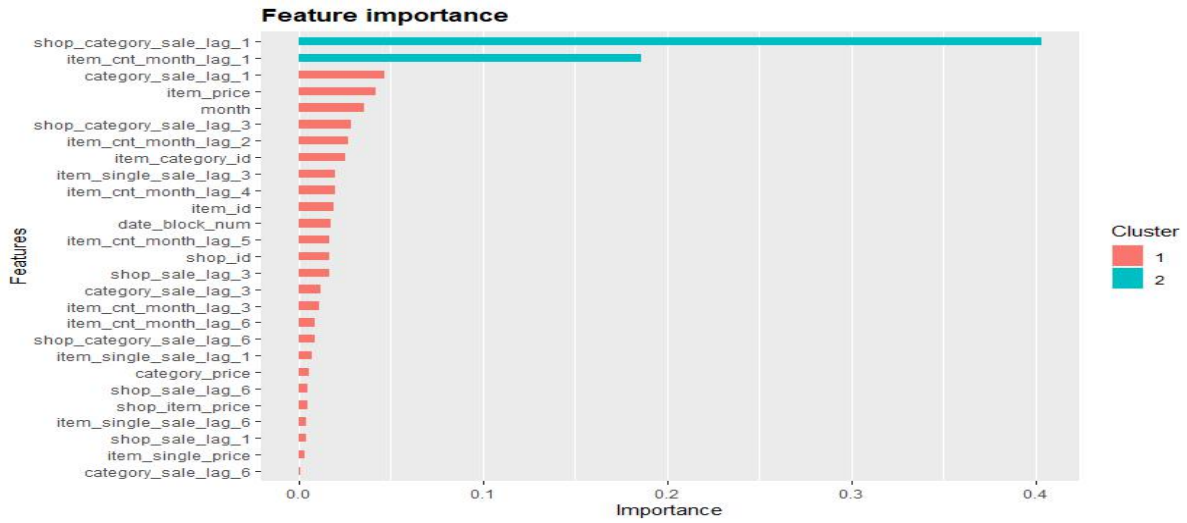
**Code(for eta):**

```
# parameter selection
para = seq(0.1,0.9,0.01)
rmse_save = 100
para_save = 0
for(i in para)
{
  xgb = xgboost( data = train,
```

```
            nrounds = 18,
            nthread = -1,
            eta =i,
            max_depth= 9,
            subsample=0.67,
            colsample_bytree=0.78,
            alpha =0.68,
            lambda=0.39,
            set.seed(123),
            eval_metric = 'rmse')
  pre_xgb = predict(xgb,newdata = test)
  rmse = sqrt(sum((testdata$item_cnt_month-pre_xgb)^2))
         /nrow(testdata)
  if(rmse<rmse_save){
    rmse_save = rmse
    para_save = i
  }
}
rmse_save
para_save
```

By parameter adjustment we reduce the RMSE of test data from 0.0432(default setting) to 0.0266.

5.3 Evaluation




After training the model, we obtain the feature importance from the model as follow.

The 1st and 2nd most important features are shop last month category sales and the item last month sale. Item price and month can also be very important features. This is reasonable since we are predicting different item sales in next month.

## 6. LightGBM

### 6.1 Model description

Commonly used machine learning algorithms, such as neural networks and other algorithms, can be trained in mini-batch mode, and the size of training data will not be limited by memory.

In each iteration, GBDT needs to traverse the entire training data multiple times. If the entire training data is loaded into the memory, the size of the training data will be limited; if it is not loaded into the memory, it will consume a lot of time to read and write the training data repeatedly. Especially in the face of industrial-grade massive data, the ordinary GBDT algorithm cannot meet its needs.

The main reason LightGBM put forward is to solve the problems encountered by GBDT in massive data, so that GBDT can be used in industrial practice better and faster.

LightGBM is a gradient boosting framework that makes use of tree-based learning algorithms, which is a very powerful algorithm when it comes to computation. It is also a fast processing algorithm. While other algorithms trees grow horizontally, LightGBM algorithm grows vertically meaning it grows leaf-wise and other algorithms grow level-wise. LightGBM

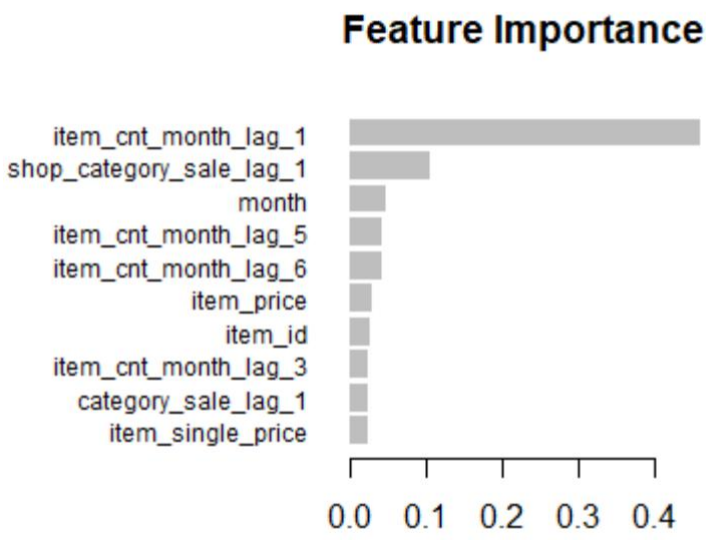
chooses the leaf with large loss to grow. It can lower down more loss than a level wise algorithm when growing the same leaf.

6.2 Parameter Adjustment and Model Performance

```
 #Set parameters for model training
train_params <- list(
  max_depth = 200,
  num_leaves = 100, num_iterations = 200,
  learning_rate = 0.4  , nthread = -1
)
#--------------------Basic Training using lightgbm----------------
# This is the basic usage of lightgbm you can put matrix in data field
print("Training lightgbm with sparseMatrix")
bst <- lightgbm(
  data = traindata2,
  label = traindata3
  , params = train_params,
  obj = "regression"
  , nrounds = 10L,
  verbose = 2L,
  boosting = 'gbdt',
  metric = 'rmse'
)
?lightgbm
```

The parameters set are shown in the screenshot above. Initially, a higher learning rate, about 0.1, was set to increase the convergence speed. This is necessary for tuning and adjusting the basic parameters of the decision tree and regularization parameter tuning. Finally, the learning rate is reduced to improve the accuracy.

We could see the performance through generating and plotting the importance of the features.




6.3 Evaluation

```
#importance of features
importance<- lgb.importance(bst)
importance
lgb.plot.importance(importance)
#test predict
pred <- predict(bst, testdata2)
summary(pred)
rmse = sqrt(sum((test$item_cnt_month-pred)^2))/nrow(test)
rmse*100
answer = cbind( item_id=test$item_id,shop_id=test$shop_id,item_cnt_montl
cor(pred,test$item_cnt_month)
```

To evaluate the performance, the summary of the prediction is generated and the metric of rmse is used. Overall, the correlation between prediction and item_cnt_month is about 0.77, which means the features chosen is correct.for the Kaggle score. we obtained a score of 3.25 in the competition using LightGBM model (rmse*100=4.78), which is generally slower than the XGboost model(rmse*100=2.66) performance.

| Name | Submitted | Wait time | Execution time | Score |
|---|---|---|---|---|
| answer.csv | 20 hours ago | 1 seconds | 1 seconds | 3.25150 |

Complete

The main reason may be parameter setting and data extraction. However, the task is successfully accomplished, and we would work step by step in the long run.