Product Demand Prediction for Unmanned Shelf

IE 597 Independent Study Report

Student: Jiyao Li (jiyaoli2), Yin Zhang (yinz3)
Leader: Tiancheng Zhao (tz14) , Menglong Li (ml10)
Instructor: Professor Xin Chen

Chapter 1: Project Background

This independent study is rooted in a collaborated project with SF-Express called “Food Supply with Unmanned Shelf”. Our task aims at predicting product sales given historical data.

The project itself is basically to install shelves with fixed positions to store food in the office area, where there’s a QR code that staff of the company can scan it and buy the snacks/food/drinks all at a one shot. The idea emphasizes several points: (1) Bring convenience to the company staff with the in-building purchase and self-assist checkout. (2) Offer customized service based on customer characteristics and needs. (3) The project benefits from SF’s leading role in areas like inventory, warehouse, and logistics, lowering the cost of distribution by directly embedding in SF-Express Network standard express delivery.

Figure 1-1 Landing Page for the Company’s Project[1]

There are several challenges in our study: (1) Since the data is dumped from a database, there are different tables with different perspectives like orders, products, and shelf_companies.To form the most representative features for the prediction model, we should join the tables which involve complicated rules and results in a dataset with large dimensions. (2) The problem has 3 major dimensions we need to slice and dice to think over: date (time), shelf_id (location), and product_id (product type). (3) There is a huge amount of zero quantities in the dataset (no sales in a single day), meaning the dataset is really sparse, making it even harder to predict. (4) Not yet all the shelves are installed simultaneously, the amount of useful information differentiated among shelves, which may need to be distinguished during the process. (5) Given the time period of the data, it is still hard to observe any trend or seasonality.

Chapter 2: Data Exploration

2.1 Understanding the business through data

The geographical plot below is a glance of how the shelves have been deployed across the country until September 2018, mainly covering the east part of China from north to south. Alos, when we zoomed in to look at Shenzhen's data at right, where the color degree denotes the number of sales of products in the region in total and the size of radius denotes the count of distinct shelves in the region, we can see that Baoan, Longgang, Nanshan, Futian, and Longhua are several districts that have the most booming business. As the shelf grows, the total sales grow, meaning there are no dominant or extremely poor performing shelves.

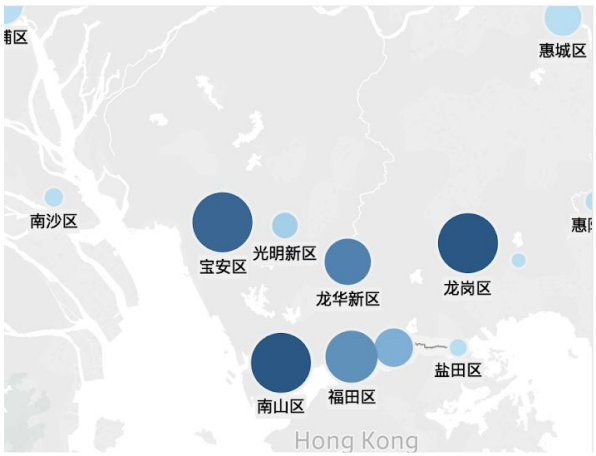


Figure 2-1 (a) Deployment & sales across China **Figure 2-1 (b)** Deployment & sales in Shenzhen

Below is a histogram of the total demand so far of all the products currently being sold. It's obvious that there are some products with significant high demand, with product id 3,4,5,19,26,28,29,32,33,103,105. Those are the products we will dig deeper into later on.

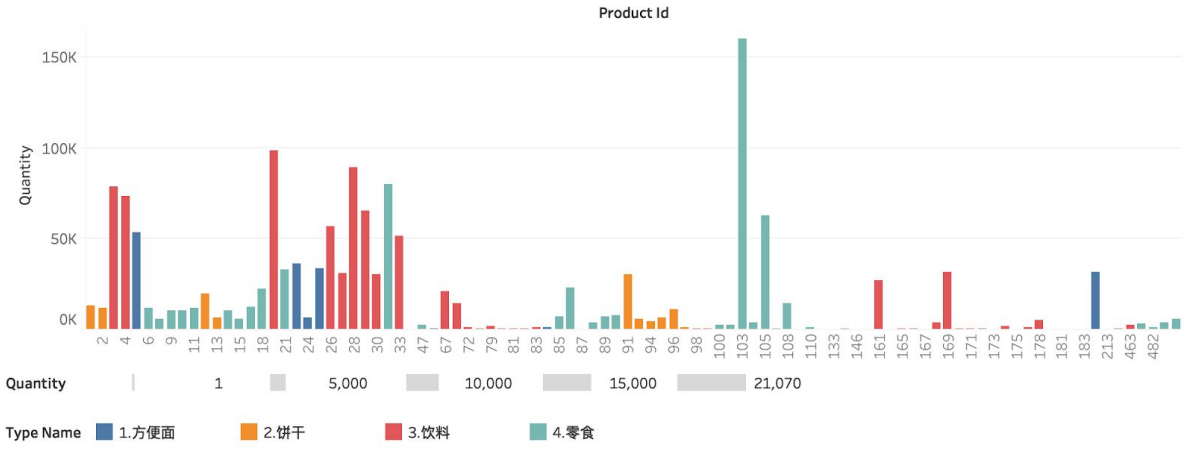
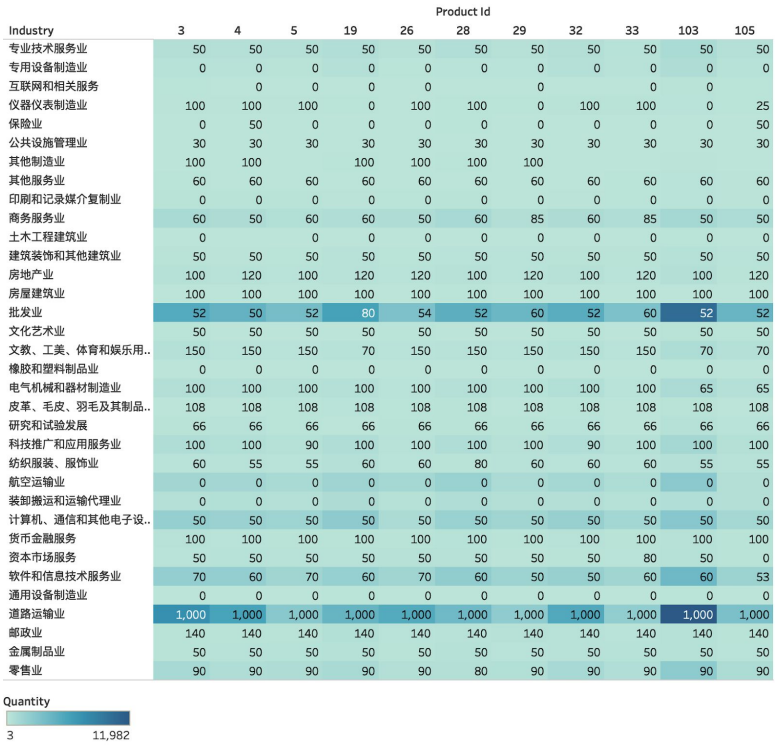


Figure 2-2 Categorical Products Sales in Total

Now we look at how those high-demand products are sold across different industries and companies. As we can see from the table below, the x-axis is the product id of high-demand products, the y-axis is

industry categories, the color of cells denotes the degree of demand (sales) and the number in the cell denotes the 75 percentile of the staff amount in that industry.

It shows that the wholesale industry and transportation industry have a significantly higher demand for purchasing the products compared to others. This happens probably because there are one or two targeted companies that this business is promoted on, which contribute to a huge amount of the sales data. The other thing we noticed is that the sales demand does not seem to be positively related to the amount of company staff, which raises to an assumption that the business should weigh the small companies as target customers as much as the giant ones.



Industry	Product Id 3	4	5	19	26	28	29	32	33	103	105
专业技术服务业	50	50	50	50	50	50	50	50	50	50	50
专用设备制造业	0	0	0	0	0	0	0	0	0	0	0
互联网和相关服务		0	0	0	0		0		0	0	
仪器仪表制造业	100	100	100	0	100	100	0	100	100	0	25
保险业	0	50	0	0	0	0	0	0	0	0	50
公共设施管理业	30	30	30	30	30	30	30	30	30	30	30
其他制造业	100	100		100	100	100	100				
其他服务业	60	60	60	60	60	60	60	60	60	60	60
印刷和记录媒介复制业	0	0	0	0	0	0	0	0	0	0	0
商务服务业	60	50	60	60	50	60	85	60	85	50	50
土木工程建筑业	0		0	0	0	0	0	0	0	0	0
建筑装饰和其他建筑业	50	50	50	50	50	50	50	50	50	50	50
房地产业	100	120	100	120	120	100	120	100	120	100	120
房屋建筑业	100	100	100	100	100	100	100	100	100	100	100
批发业	52	50	52	80	54	52	60	52	60	52	52
文化艺术业	50	50	50	50	50	50	50	50	50	50	50
文教、工美、体育和娱乐用..	150	150	150	70	150	150	150	150	150	70	70
橡胶和塑料制品业	0	0	0	0	0	0	0	0	0	0	0
电气机械和器材制造业	100	100	100	100	100	100	100	100	100	65	65
皮革、毛皮、羽毛及其制品..	108	108	108	108	108	108	108	108	108	108	108
研究和试验发展	66	66	66	66	66	66	66	66	66	66	66
科技推广和应用服务业	100	100	90	100	100	100	100	90	100	100	100
纺织服装、服饰业	60	55	55	60	60	80	60	60	60	55	55
航空运输业	0	0	0	0	0	0	0	0	0	0	0
装卸搬运和运输代理业	0	0	0	0	0	0	0	0	0	0	0
计算机、通信和其他电子设..	50	50	50	50	50	50	50	50	50	50	50
货币金融服务	100	100	100	100	100	100	100	100	100	100	100
资本市场服务	50	50	50	50	50	50	50	50	80	50	0
软件和信息技术服务业	70	60	70	60	70	60	50	50	60	60	53
通用设备制造业	0	0	0	0	0	0	0	0	0	0	0
道路运输业	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
邮政业	140	140	140	140	140	140	140	140	140	140	140
金属制品业	50	50	50	50	50	50	50	50	50	50	50
零售业	90	90	90	90	90	80	90	90	90	90	90

Figure 2-3 High Demand Product Sales (color) in Different Industries

The following table shows how the sales demand reacts to the change in prices (discount). Here the x-axis denotes the product id of the high demand products we selected previously, the y-axis is a time sequence, ranging from January 2018 to September 2018. The color of cells in the table denotes the average amount of discount for the specific product in that period of time, while the number in the cell denotes the total sales of the product in the same time. You can see the effect of discount on product demand by comparing the trend of numbers and colors with surrounding cells.

The dataset is actually composed of 2 data sources. One is from January to April and the other is from June to September. There's a clear break point in the sales and the discount is only available after June, so we simply look at the data from June to September. Then, it is obvious whenever the color gets darker, the sales grow higher compared to surrounding ones. For example, the sales of product 5 in August increases from 5 to 45 in the first week under a deep discount, and the sales keep at a level of higher than 10 for the following weeks until the discount cools down and the demand decreases again. So, we can roughly reach to a conclusion that any discount or coupon is worth trying based on careful calculation of demand change and profit margin.

		Product Id										
Y.. Month..	Week of ..	3	4	5	19	26	28	29	32	33	103	105
2.. January	Week 1	3,032	3,818	3,098	3,842	3,052	4,502	1,976	3,984	2,190	6,124	50
	Week 2	4,642	5,792	4,264	5,266	4,762	5,650	3,280	6,964	2,920	12,616	3,222
	Week 3	7,476	9,852	6,370	10,048	6,014	10,652	6,438	9,276	5,170	21,070	12,406
	Week 4	3,759	5,601	4,278	6,093	3,588	6,253	4,409	4,498	3,008	11,921	7,693
	Week 5	1,221	1,967	1,069	1,991	1,050	1,474	1,173	1,954	1,022	4,863	3,564
Februa..	Week 5	310	539	292	862	315	541	395	454	308	1,213	896
March	Week 9	1,177	381	1,095	1,732	262	1,673	1,386	1,023	845	3,441	1,704
	Week 10	3,132	944	2,522	4,357	1,141	3,892	3,380	2,503	2,374	10,191	4,342
	Week 11	3,379	1,422	2,491	4,563	1,097	3,830	3,068	2,060	2,621	9,473	3,429
	Week 12	3,362	1,495	2,441	4,371	1,717	4,115	2,605	2,938	2,499	9,426	2,185
	Week 13	3,598	924	2,321	4,698	1,498	4,717	2,945	2,523	2,630	9,117	1,912
April	Week 14	3,213	505	1,376	3,046	733	3,311	2,224	2,717	1,943	5,653	1,439
	Week 15	4,205	1,266	1,447	3,493	778	4,048	2,427	3,192	2,830	8,670	3,895
	Week 16	3,520	1,267	733	4,065	1,166	2,502	2,658	2,520	2,622	6,809	4,667
	Week 17	4,179	2,900	1,063	5,233	3,178	4,060	3,190	2,859	1,838	10,078	5,998
	Week 18	428	423	186	633	508	720	418	290	126	1,162	606
June	Week 22	64	209	20	136	128	172	78	114	20	180	101
	Week 23	270	547	53	389	421	430	289	228	80	1,209	548
	Week 24	235	515	31	504	514	540	324	224	70	1,141	463
	Week 25	169	426	14	297	345	442	268	193	47	821	428
	Week 26	237	517	20	467	357	551	270	167	49	911	579
July	Week 27	299	533	19	299	379	288	221	104	133	1,139	795
	Week 28	264	531	10	303	386	168	162	83	24	893	421
	Week 29	271	484	8	284	351	154	205	106	32	856	322
	Week 30	299	473	13	321	424	105	174	52	35	848	182
	Week 31	127	211	5	94	143	18	90	8	13	336	52
August	Week 31	163	270	45	203	234	70	142	16	29	534	64
	Week 32	284	505	13	299	305	78	204	13	29	966	121
	Week 33	331	596	12	435	473	49	265	15	32	1,071	100
	Week 34	405	651	10	567	588	35	291	9	36	1,139	70
	Week 35	372	673	23	626	565	40	319	1	58	1,412	129
Septe..	Week 35	59	92	7	98	76	4	91		5	185	14
	Week 36	434	801	10	940	768	37	499	1	41	1,497	115
	Week 37	613	704	9	1,146	819	40	557		27	1,430	68
	Week 38	391	748	15	761	852	56	572	1	35	1,449	62
	Week 39	368	636	9	744	542	37	404	4	32	994	49

Avg. AVG Discount

0.000 1.905

Figure 2-4 High Demand Product Sales (number) regarding to Discount (color)

Below is a truncated part of a multi-index plot showing how demand actually reacts to the change of prices for each product. In the y-axis, the first level index denotes the product id, the second level index denotes the price of a single piece of product, while the x-axis denotes the average sales of the product over the period with the specific price, which represents the demand or popularity of the product over the price range. For example, if 100 pieces of product A was sold over 10 days, then its value on x-axis here should be 10.

Here only the plot of 5 products is shown due to space limitations. There might be some variations, but the general trend for all products is the same: extreme high prices result in the lowest demand, while extremely low prices correspond to rather high demand. While in the middle of the price range, the demand always fluctuates, meaning that the customers are not as prices sensitive to the products as they are to the extreme prices, or there are other confounding factors affecting this. But generally speaking, it still follows the rules that the demand for products is negatively related to the sale price.

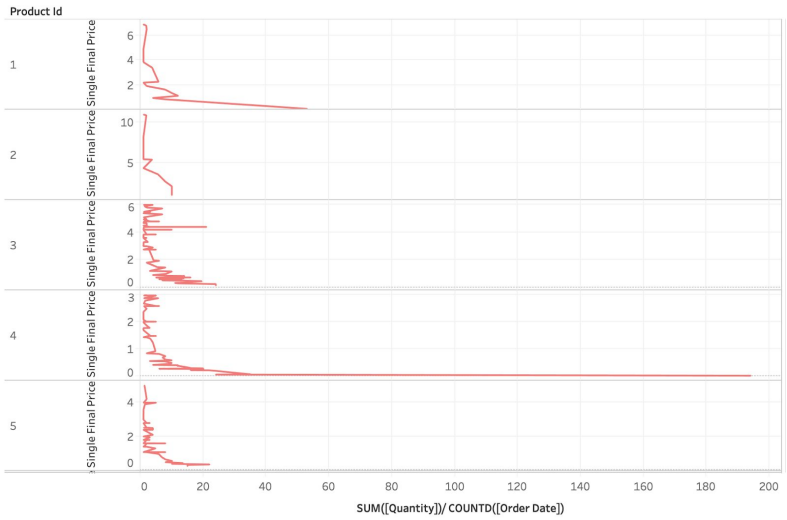


Figure 2-5 Sample Products Average Sales Relates to Discount

The chart below shows the comparison of relatively high demand products and price sensitive products. To begin with, it's natural to think that (1) high demand products are less price/discounts sensitive. (2) The company tends to give out more discounts on low demand products to stimulate sales.

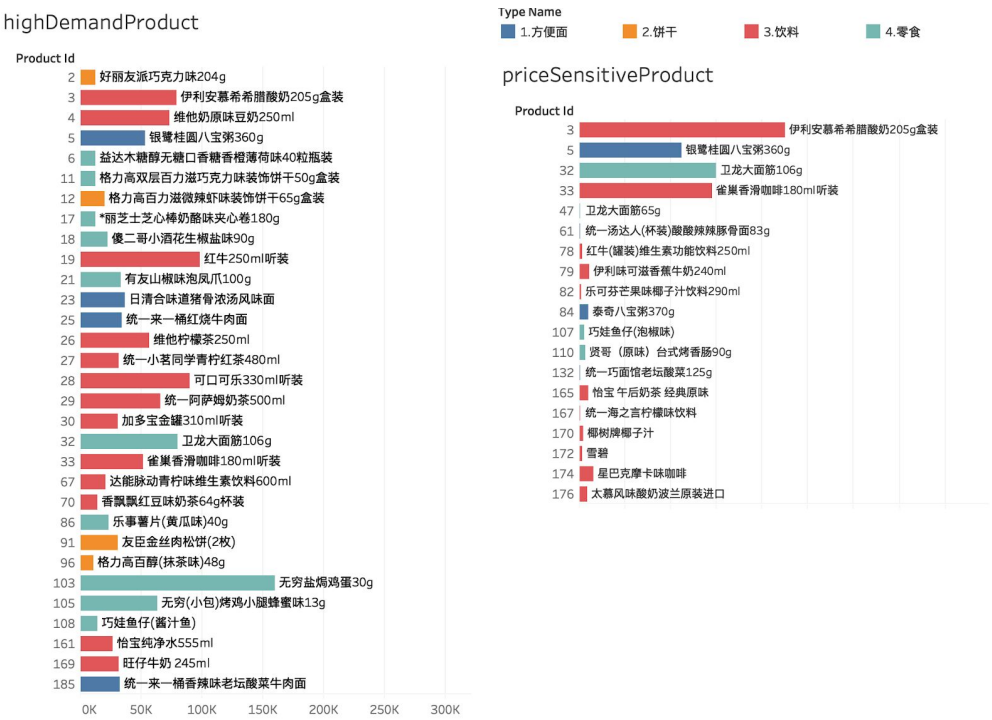


Figure 2-5 Comparison of High Demand Products & Price Sensitive Products in Category

It then can be seen that most of the products that are more price sensitive than others are either drinks or relatively low demand products, and most of the high demand products are those snack and drinks commonly seen. So, we form the assumption that even though drinks seem to be the most popular choice, the products that more price sensitive are somewhat special, not frequently seen, or are new flavors of some well-known brands and some with an even higher price. These products are more likely to be substituted by the high demand drinks (often well known, classic, lower price). People who buy products

of this kind are trying something new or better, or simply because the shelf was short of usual supply. Then money is not the major issue in these scenarios.

2.2 Data Exploration and Analysis

After the cleaning and preprocessing part, the dataset used for training the model initially has a total of 1308010 rows and 38 columns, meaning there are initially 38 features, which includes the shelf id, product id, the sequence of dates, price of product, which week it is, whether it is weekday or weekend, quantity of sales on the day, the average amount of sales in the past, the max sales in the past, the variation of sales in the past, the dummy variable of product category, type, second type and district, etc..

There are a total of 1308010 rows, which constitutes of a total of 188 shelves, 201 products. Each pair of these products and shelves has a total amount of 121 days which is exactly 4 months (18 weeks) of sales data, as is shown in the table below. The demand we are trying to predict also considers these dimensions, which is the sales demand of a specific product-shelf pair in a specific day.

Table 2-1 Basic Info: ID Range and Count for Shelf, Product and Date

		Shelf	**Product**	**Date**
Range of id	Min id	1	1	1
	Max id	1757	1379	121
Total Count (of distinct id)		188	201	121

Below is a glance at how the quantity-related features distribute. By looking at the last_weekday_sale, we can see that over 75% of the sale is actually 0, due to several reasons: low demand or late installation of shelves. And it can be seen that the standard deviation of quantities of all kinds is really large, nearly at the same magnitude of the mean.

Table 2-2 Distribution of Quantity-Related Attributes

	past_average	**past_max**	**past_var**	**last_weekday_sale**
count	1.308010e+06	1.308010e+06	1.308010e+06	1.308010e+06
mean	1.197606e-01	7.918380e-01	4.352738e-01	1.242972e-01
std	3.642771e-01	2.115514e+00	1.179175e+01	7.605347e-01
min	0	0	0	0
25%	0	0	0	0
50%	0	0	0	0
75%	7.142857e-02	1.000000e+00	7.142857e-02	0
max	2.500000e+01	1.510000e+02	1.606571e+03	1.510000e+02

Below is two histograms describing the sales distribution across different product categories(left) and type(right). The curve lines are estimated kernels (probability density function). Take the left chart as an example, it can be seen that the orange curve is more skewed towards the right, meaning the demand for fast food is higher than the other two, and the demand for the green curve which represents the drink is slightly higher than leisure as well. So, the priority would be fast food, drink, and leisure food. The right chart denotes the tendency of four product types, which has been hidden now. What we can see is that the orange curve, which is type 1 has the tendency towards higher sales compared to others.

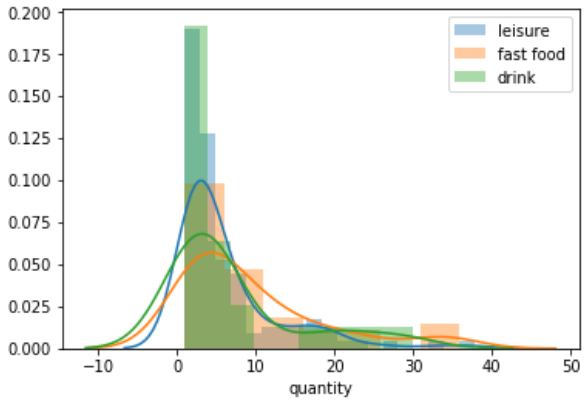


Figure 2-6 (a) Histogram of Sales across Category

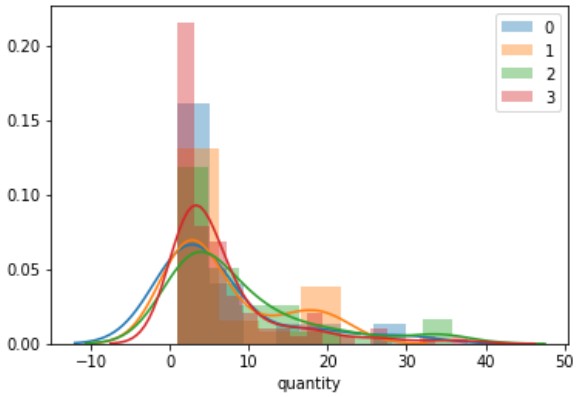


Figure 2-6 (b) Histogram of Sales across Type-1

The left plot below has an x-axis of product id and a y-axis of the count of shelves, denoting the total amount of shelves that each product is deployed on. It can be seen that the variation of deployment coverage is really large. However, nearly one-third of the products have been deployed on more than 100 shelves, which may probably contribute to the higher sales and a coincidence with those high demand products.

The right plot below has an x-axis of date, and y-axis of sales quantity, showing the trend of total sales overalls product and shelf over time. It can be observed that the total sales have a significant increase around the 90th day, and the demand level increases a little ever since. It is known that some of the shelves are not set until halfway through the four months, and some of the products are not deployed on specific shelves until some day, which makes the total demand jumps over time. But it should be noticed that the demand pattern after 90th day contains valuable information, which should be included in the training data for the prediction model. And it would be better if the time period could be expanded and more data could be fed into the model, so as to capture the patterns in different periods of time.

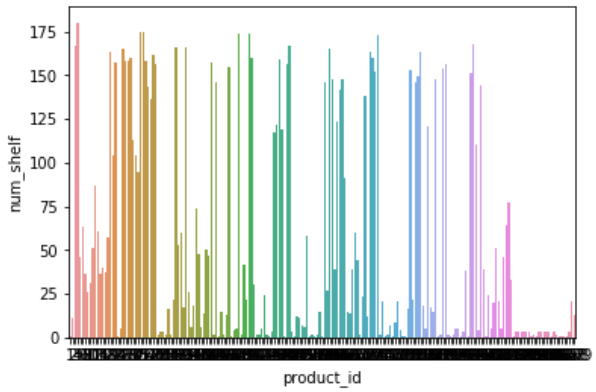


Figure 2-7 (a) Histogram of Shelf Counts across Products

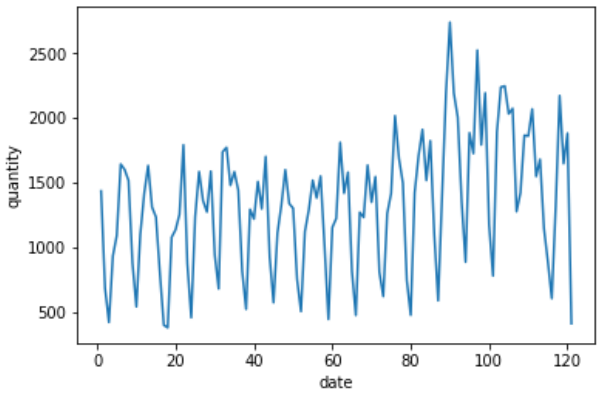


Figure 2-7 (b) Line Plot of Sales across Date

2.3 Data Filtering

We've already noticed that there are a large number of zero sales (over 75%), which makes the dataset really sparse. And It's really hard to predict the sales of some specific product-shelf pairs given their historical data, where there shows no sign of trend or seasonality. These pairs are barely predictable.

To address this situation, we compromise to only predict restricted pairs with conditions. Then the concept of coefficient of variation (CV) and Average Demand Interval (ADI) were brought to the front. ADI means the average amount of days that needed for a single sale to appear, which is the inverse of the mean. While CV is calculated as the standard deviation divided by the mean, so it is STD multiplies the ADI. Below are a chart and a table describing how the CV of all the pairs distribute.

The value of CV measures is a measure of relative variability. It is the ratio of the standard deviation to the mean (average), measuring the dispersion of data points in a data series around the mean. Since we are not only using the standard deviation, also considering the means, CV is a useful statistic for comparing the degree of variation among different data sources, even if the means are drastically different from one another. For example, if sample A has a CV of 26% and sample B has a CV of 15%, we can say that sample A has more variation than B, relative to their means.

Table 2-3 Description of Product-Shelf Pair CV

count	mean	std	min	25%	50%	75%	max
10810	6.085130	2.858573	1.012390	3.773951	5.430815	7.745698	11.000000

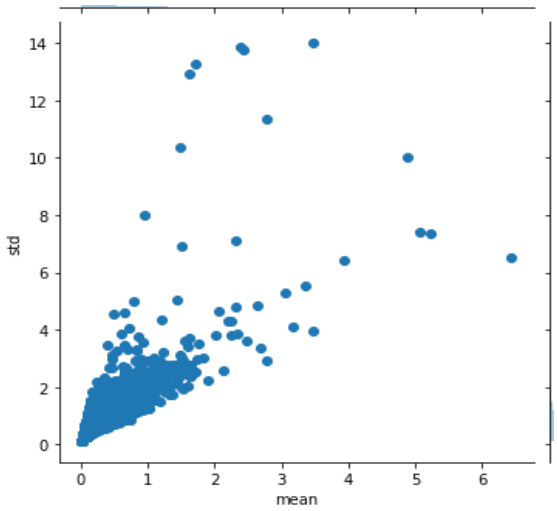


Figure 2-8 Scatter Plot on Mean & Std of Product-Shelf Pairs

Since the characteristics and performance of product-shelf pairs may differentiate from each other drastically, CV then can be used as an unified criteria to see how diverted the data is. So, it is used to filtering out the product-shelf pairs that have too much variation, and the pairs that have less variation are left, thus more solid pattern to track.

The distribution of pair's CV, and their std & mean are shown in the table and scatter plot above. A proper CV threshold for filtering is determined based on that. By applying this, the dataset will be further cleaned and better prepared for modeling.

Chapter 3: Feature Selection

For the datasets, there are over thirty feature columns, which contains both useful and useless information for the prediction. In the meanwhile, the instances are not enough given so many feature columns to consider. So, feature selection is very important for this prediction job.

The features we are using here include product id, shelf id, sale date, sale price, inventory, history sale quantity, shelf location, characteristic of the product, week number, weekend index, promotion index and so on. Some of these features are represented by real values and some are by dummies.

We searched through the different features and calculated the correlation between each of them and the quantity that is to be predicted. We found that the history sale quantity was the most correlated feature, then the price and product characteristics. We can conclude that this is a time series process which means that the future sale quantity is strongly correlated with historical data. Therefore, we came up with an idea that we needed to construct some artificial factors to better describe the quantity.

The new features related to the quantity that we added is Average Demand Interval (ADI) and Coefficient of Variation (CV), each represented by 3 dummies. Here ADI is generally used to address the intermittent problems. We also added 2 artificial features representing sale change trend. By this process, the dataset is expanded and this is a risky action indeed, since this may result in worse prediction accuracy if the features added are not useful. The ADI and CV dummies split the product-shelf pairs by their quantities' ADI and CV using 2 cutoff values. For the instances with the same product-shelf pair in different date, they share the same 6 dummies of ADI and CV.

Meanwhile, the useless features are eliminated from the datasets, like date index and other features which don't vary by different product-shelf pair. We dropped these useless features and this is good for improving prediction accuracy.

Finally, after selection, adding and elimination of features, we filtered the instances by choosing product-shelf pairs with relatively steady sale quantity distribution. Then, we check the prediction accuracy represented by R square (R^2), Mean Square Error (MSE) and Mean Absolute Error (MAE) by the LGBM algorithm in different settings. Since each product-shelf pair has a total of 121 days of data points, we splitted the training data as the first 95 days and the remaining 26 days for testing.

Chapter 4: Model Selection(LGBM)

4.1 Light Gradient Boosting Model

Light gradient boosting machine (LGBM) is a gradient boosting framework that uses a tree-based learning algorithm. The framework is fast and is designed for distributed training. It supports large-scale datasets and training on the GPU. In the case of gradient boosted decision trees, successive models are found by applying gradient descent in the direction of the average gradient, calculated with respect to the error residuals of the loss function. In this way, the loss is reduced in every iteration where a new tree is generated for yielding the average gradient.

The algorithm can be summarised as follows:

1. Set the initial prediction F_0 as the mean of real labels
2. Calculate the gradient r_{im} of loss function L(y_i, F_{m-1}) respecting to the predictions F_{m-1} of this step
3. Use a regression tree $h_m(x)$ to fit the gradient in this step
4. Use the predicted gradient to calculate the next step's prediction $F_m = F_{m-1} + \lambda_m h_m(x)$
5. Repeat the process until reach the stopping criteria

4.2 LGBM Fused with Trend Feature

Even though the model is supervised learning, it is still involved with some sort of time series. To take the effect of time sequence into account, there are different basis we should consider, for example, average sales for the past month, average sales for the past week, sales of same day last week..etc. Those sampled data points provided valuable information from historical data in a hierarchical manner.

So, in order to provide more accurate prediction, we'd better provide a baseline first - the trend of future sales compared to recent data. That is, we first train two logistic regression models to predict two things for the targeted day: (1) whether there will be sales (yes/no). (2) How the the sales will change (decrease/same/increase). These two models reached an accuracy of 99%. Also, by transferring these two information into dummy variables and combining them to the feature vector, we trained a fused LGBM model.

4.3 Fused LGBM with Stacking

Stacking is one of the ensemble method, the others includes but not limited to bagging and voting. Stacking is short for stacked generalization, it is based on the idea that, instead of using trivial functions (such as voting) to aggregate the results of all predictors in an ensemble, we can just train a model to perform this aggregation process. Figure 4-1 shows such an ensemble performing a regression task on a new instance. Each of the bottom three predictors predicts a different value (3.1, 2.7, and 2.9), and then the final predictor (called a blender, or a meta learner) takes these predictions as inputs and makes the final prediction (3.0).[2]

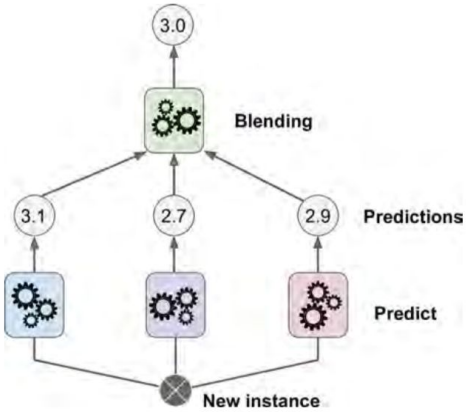


Figure 4-1 Aggregating predictions using a blending predictor [2]

Here instead of simply aggregating predictions from different predictors, we combine the predictions with the other features, and use that as the new feature vector to train the LGBM model.

4.4 Model Result and Comparison

Then we show the result of prediction by the LGBM algorithm in the different scenario and in different datasets of cities:

Table 4-1 Modeling Results of Different Cities

	Shenzhen	**Shanghai**	**Chengdu**	**Taiyuan**
filter criteria	cv(quantity<2)	cv(quantity<2)	cv(quantity<3)	cv(quantity<2)
input	1308010 obs of 38 features	2603664 obs of 38 features	2891976 obs of 36 features	3495660 obs of 30 features
output	The predicted quantity of sale on the same day			
LGBM only				
R2	0.48461	0.56650	0.41251	0.44928
MSE	8.90197	2.33365	1.15861	0.80452
MAE	0.71864	0.35490	0.21469	0.16358
LGBM with two artificial (Trend) features				
R2	0.84109	0.81354	0.72399	0.79154
MSE	2.74478	1.00375	0.54432	0.30453
MAE	0.30807	0.16078	0.08627	0.06287
LGBM + two artificial features (Trend) + stacking				
R2	0.81574	0.71238	0.69907	0.69896
MSE	3.18264	1.54834	0.59347	0.43977
MAE	0.30538	0.18156	0.08278	0.06250

The metrics we use here are R square (R^2), Mean Square Error (MSE) and Mean Absolute Error (MAE), where MSE and MAE measures the error, and R^2 measures how good the model is. It turns out that for each city, the LGBM fused with trending features performs the best, with an R^2 score of 0.84109 (Shenzhen), 0.81354 (Shanghai), 0.72399 (Chendu), 0.79154 (Taiyuan) respectively. And for each model, the performance of Shenzhen and Shanghai stand out compared to the other two cities.

One thing to note is that, in order to get the models with fair performance, the value of CV filtering should be carefully chosen. As we can see, the filter threshold for Chengdu is 3, larger than the other three that are 2. It is reasonable that the quality of data sources from different city varies depending on the size and status of business in that city. Thus different filter criteria should be applied to properly clean and preprocess the various datasets.

One more thing is that the stacking model is not as good as expected. Its performance is worse than the fused LGBM. One assumption is that the newly added prediction column has largely diluted the effect of other features, which in turn may ignore some useful information contained in those features. However, it is worth trying with 3 or more predicts and aggregate them with blenders, which will need larger data size.

Chapter 5: Evaluation and Analysis

Firstly, we need to mention that the data is pretty messy and the filtering process is of much importance because of the large coefficient of variation of some product-shelf pair. After filtering, the good instances remain while others dropped. In this way, we can see that only the product-shelf pairs with a relatively steady distribution of sale quantity are kept in the input, which improves the running speed and prediction accuracy.

Given the results of R2 score, mean squared error (MSE) and mean absolute error (MAE), we can conclude that the two artificial trend features ("sale change" and "has sale") are playing a very important role in the prediction accuracy. With the "has sale" dummy (1 if there is a sale on that instance, 0 if not) and "sale change" dummy (1 if sale increase, 0 not change, -1 if decrease), the R2 score of the LGBM algorithm prediction is improved a lot for all the cities. From this phenomenon, we inferred that the factor of sale change/trend was extremely significant for the prediction of future sale quantity. With these two features, the algorithm can get a better understanding of the future trend and grab a reasonable baseline for prediction, which will increase the model accuracy a lot.

Besides the artificial trend features, we also tried the stacking method - add predicted labels back to the dataset as a new feature column and retrain the new algorithm to do prediction. The result is kind out of expectation. This new feature impairs LGBM prediction accuracy. We conjecture that this is due to the new stacking which has weakened the algorithm to extract useful information from the other feature columns.

Chapter 6: Conclusion

Finally, by the thorough exploration of data, building and analyzing the LGBM model, our suggestions for improving the unmanned-shelf business are as the following:

1. For most of the shelves, products under the category of drinks play a major part in the high demand product set, especially those with common names.
2. Applying coupons/promotions is an effective way to stimulate the sale, especially useful for those high demand category (drink) products with new flavors or new designs however.
3. The business performance shows no significant difference between companies that has different amount of staff, either large or small. It means that we should place equal emphasize on promoting the business on companies with different sizes.

And for further improving the prediction model, we concluded that:

1. The modeling procedure includes the feature selection, data filtering and algorithm selection. The criterias for these operations are based on the characteristics of the four cities' datasets, and the results are robust.
2. LGBM is the best algorithm for prediction comparing to several other machine learning algorithms attempted. For example, the the performance of neural network is really bad, which may due to improper initialization.
3. There are some drawback and uncertainty for this project due to very limited data. The split point of training and testing data can affect the prediction accuracy, due to the unstable trend around some dates.
4. The two predicted features representing the future sale quantity trend are very important. With the help of them, the prediction accuracy by LGBM can be significantly improved.
5. Implementing the algorithms on the original data is rather time consuming and inefficient, since the dataset is huge and some instances are useless; after we filter the data and drop all the unstable product-shelf pairs, the running time is significantly reduced and the prediction accuracy is improved. Even though we abandoned some pairs, but the overall prediction performance increased significantly. This might be improved if larger dataset can be acquired and those unstable pair will finally turn to have stable patterns and become predictable.
6. Stacking method which is sometimes used for improving accuracy does not work here. With new stacking added there, the prediction accuracy decreases. This phenomenon is found over all cities and several other algorithms in this project. Stacking model aggregating more predicts may worth trying if given more data in the future.

Reference

[1] https://www.feng1.com/

[2] Géron A. Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems[M]. " O'Reilly Media, Inc.", 2017.

Appendix

1. Core code for constructing models and training the algorithms:
This is run in the environment of Python 3.6 constructed by the Anaconda release.

```
# import necessary libraries
import numpy as np
import pandas as pd
import sklearn.metrics as metrics
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from matplotlib import pyplot
from lightgbm import LGBMRegressor

# import data and add is_sale, is_change features
dfs = pd.read_table('daily_sale_310100.txt',sep=',')
is_sale=dfs['quantity'].map(lambda x: 1 if x>0 else 0)
dfs=dfs.drop(['Unnamed: 0'], axis=1)
dfs.loc[:,'is_sale']=is_sale

dfs.sort_values(['shelf_id','product_id','date'])
a=dfs.loc[0:2603662,'quantity'].reset_index(drop=True)
b=dfs.loc[1:2603663,'quantity'].reset_index(drop=True)
c=pd.DataFrame({'f1':a,'f2':b,'is_change':np.zeros(len(a))})
c['is_change']=np.where(c['f1']<c['f2'],1,c['is_change'])
c['is_change']=np.where(c['f1']>c['f2'],-1,c['is_change'])
d=list(c['is_change'])
d.append(0)
dfs.loc[:,'is_change']=d
df_sh=dfs[dfs['date']<=121]

# split the training and testing sets
day=95
df_train = df_sh[df_sh['date']<day]
df_test = df_sh[df_sh['date']>=day]

# add the two features (ADI, CV) for train
df_train['pairADIclass']=df_train['quantity']
df_train['pairCVclass']=df_train['quantity']
cvcutoff1=1.5
cvcutoff2=2
meancutoff1=0.5
meancutoff2=0.15
df_ADIclass=df_train[['shelf_id','product_id','pairADIclass']].groupby(['shelf_
```

```
id','product_id']).transform(lambda x: x-x+1 if\
    x.mean()>meancutoff1 else(x-x+2 if x.mean()>meancutoff2 else x-x+3))
df_CVclass=df_train[['shelf_id','product_id','pairCVclass']].groupby(['shelf_id
','product_id']).transform(lambda x: x-x+1 if\
    x.std()/(x.mean()+0.000001)<cvcutoff1 else(x-x+2 if
x.std()/(x.mean()+0.000001)<cvcutoff2 else x-x+3))
df_train['pairADIclass']=df_ADIclass
df_train['pairCVclass']=df_CVclass
df_train=pd.get_dummies(df_train,prefix=['ADIclass','CVclass'],columns=['pairAD
Iclass','pairCVclass'])
df_train=df_train.groupby(['shelf_id','product_id']).filter(lambda
x:x['quantity'].std()/(x['quantity'].mean()+0.000001)<2)

# add the two features (ADI, CV) for test
df_test['pairADIclass']=df_test['quantity']
df_test['pairCVclass']=df_test['quantity']
cvcutoff1=1.5
cvcutoff2=2
meancutoff1=0.5
meancutoff2=0.15
df_ADIclass=df_test[['shelf_id','product_id','pairADIclass']].groupby(['shelf_i
d','product_id']).transform(lambda x: x-x+1 if\
    x.mean()>meancutoff1 else(x-x+2 if x.mean()>meancutoff2 else x-x+3))
df_CVclass=df_test[['shelf_id','product_id','pairCVclass']].groupby(['shelf_id'
,'product_id']).transform(lambda x: x-x+1 if\
    x.std()/(x.mean()+0.000001)<cvcutoff1 else(x-x+2 if
x.std()/(x.mean()+0.000001)<cvcutoff2 else x-x+3))
df_test['pairADIclass']=df_ADIclass
df_test['pairCVclass']=df_CVclass
df_test=pd.get_dummies(df_test,prefix=['ADIclass','CVclass'],columns=['pairADIc
lass','pairCVclass'])
df_test=df_test.groupby(['shelf_id','product_id']).filter(lambda
x:x['quantity'].std()/(x['quantity'].mean()+0.000001)<2)

xtrain, xtest = df_train.drop(['quantity'], axis=1), df_test.drop(['quantity'],
axis=1)
ytrain, ytest = df_train['quantity'].values, df_test['quantity'].values

rfreg = LGBMRegressor(n_estimators=1000, n_jobs=3)
rfreg.fit(xtrain, ytrain)

p = rfreg.predict(xtest)
df_test_extended=df_test.copy()
df_test_extended['forecast']=p.copy()
p_week=df_test_extended.groupby(['shelf_id','product_id','weeknum']).forecast.s
um()
ytest_week=df_test_extended.groupby(['shelf_id','product_id','weeknum']).quanti
```

```
ty.sum()
mse = metrics.mean_squared_error(ytest_week,p_week)
mae = metrics.mean_absolute_error(ytest_week,p_week)
r2 = metrics.r2_score(ytest_week,p_week)
mape = mae/np.mean(ytest_week)
print('day %d - mse %.5f' % (day, mse))
print('day %d - mae %.5f' % (day, mae))
print('day %d - r2 %.5f' % (day, r2))
print('day %d - mape %.5f' % (day, mape))

q1=rfreg.predict(xtrain)
q2=rfreg.predict(xtest)
df_train.loc[:,'p_q1']=q1
df_test.loc[:,'p_q1']=q2
xtrain, xtest = df_train.drop(['quantity'], axis=1), df_test.drop(['quantity'],
axis=1)
ytrain, ytest = df_train['quantity'].values, df_test['quantity'].values

rfreg1 = LGBMRegressor(n_estimators=1000, n_jobs=3)
rfreg1.fit(xtrain, ytrain)

p = rfreg1.predict(xtest)
df_test_extended=df_test.copy()
df_test_extended['forecast']=p.copy()
p_week=df_test_extended.groupby(['shelf_id','product_id','weeknum']).forecast.s
um()
ytest_week=df_test_extended.groupby(['shelf_id','product_id','weeknum']).quanti
ty.sum()
mse = metrics.mean_squared_error(ytest_week,p_week)
mae = metrics.mean_absolute_error(ytest_week,p_week)
r2 = metrics.r2_score(ytest_week,p_week)
mape = mae/np.mean(ytest_week)
print('day %d - mse %.5f' % (day, mse))
print('day %d - mae %.5f' % (day, mae))
print('day %d - r2 %.5f' % (day, r2))
print('day %d - mape %.5f' % (day, mape))
```

2. More source files on cloud:
https://drive.google.com/drive/folders/1Ee9YvFKQlzCZ53eY3vNfLQtgAXmQUmHe?usp=sharing