GB656 Final Project Report

Team Submission (Ming Chen, Dan He, Minqi Liao)

1 Introduction

1.1 Business framing

The purpose of this model is to predict future sales given historical sales data. With the predicted sales for each product in each store, companies, especially manufacture and retail companies knows in advance how many products they need to produce and send to their store branches so that the store branches have lower chance of out-of-stock situation, which helps ensure a continuous revenue from each products while keeping the inventory cost at low as possible.

1.2 Problem statement

We are provided with a train dataset containing daily historical sales data from January 2013 to October 2015, along with the descriptive dataset about the products and shops. The task is to forecast the total sales for every product sold in every shop in November 2015. Our model is required to learn from the data in train dataset, to see patterns of how features affect monthly sales and use that knowledge to predict the future month's sales in test dataset.

2 Data Cleaning

2.1 Data descriptions

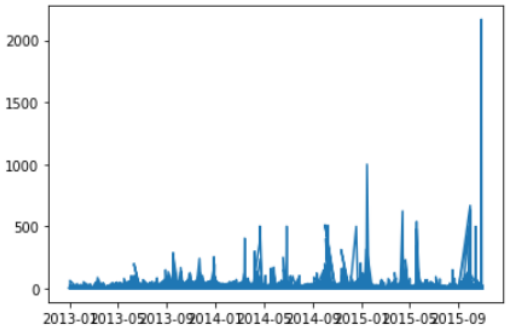
- sales_train.csv – (date, date_block_num, shop_id, item_id, item_price, item_cnt_day)
- test.csv – (id, shop_id, item_id)
- shops.csv – (shop_id, shop_name)
- items.csv – (item_id, item_name, category_id)
- item_categories – (category_id, category_name)

2.2 Delete negative values

We first generate a descriptive statistics table for all columns in the train data. We find that both the price and sales have negative numbers. Negative values may represent returned items. In that case, it is better to set them to zero since no item was sold on that date and that returned item is included in previous sales. In addition, There are 2935849 rows in train data, among them only 7536 rows (0.00256%) have negative value, so we decide to delete all negative values.

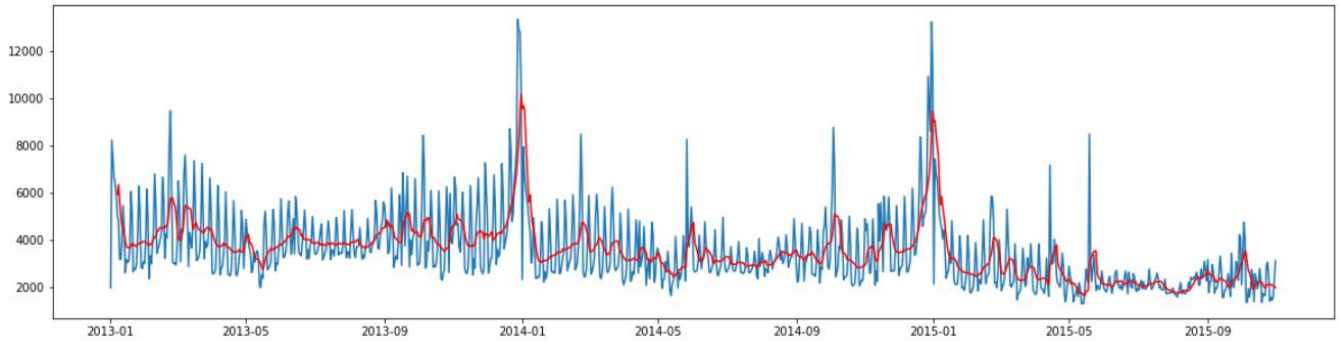
2.3 Delete outliers

The graph on the right side shows the distribution of daily sales for all products from 2013 to 2015. We can see that there are rarely values above 1000, so we decide remove records with sales higher than 1000. Similarly, we drop the records with a price higher than $300000.

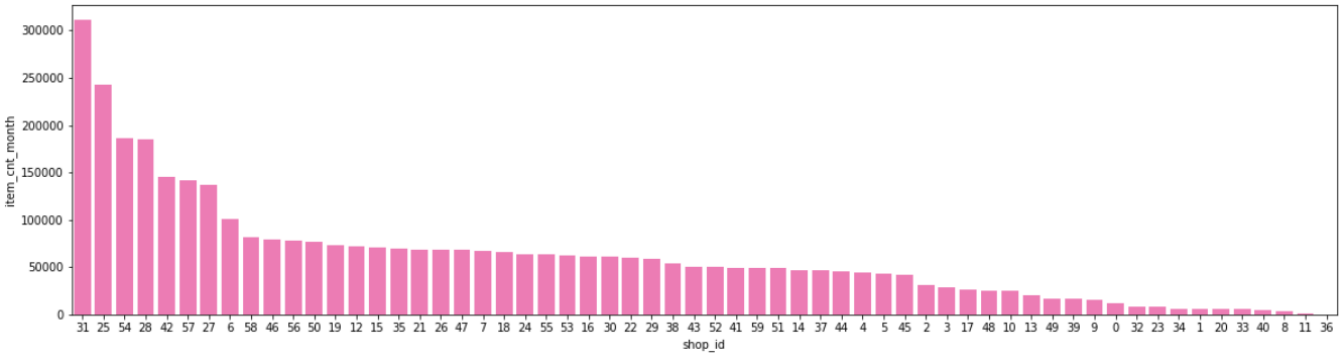


3 Exploratory data analysis

a) The line graph below depicts daily sales of all products of the company. We can see that there is obvious seasonality in the sales data, with the highest sales occurring in January. This means we may need to include year and month in our features.



b) The bar graph below shows the total sales of all products at each shop. As we can see, certain stores have relatively higher monthly sales, so we'll also use shop_id as features.



4 Feature Engineering

We are only provided with raw time series data, in which there is no concept of input and output features. Hence, in order to use machine learning algorithms, it must be re-framed as a supervised learning dataset by choosing the variable to be predicted and using feature engineering to construct all of the inputs that will be used to make predictions for future time steps. The following steps are executed on both train data and test data so as to make sure we can predict the sales for products in test data using the same input features.

4.1 Split date to year and month

We want to test whether year and month works better than date_block_number (unique identifier for the 33 months), so instead of keeping date, month, and year as a single entity, it was broken into two different inputs: year, month.

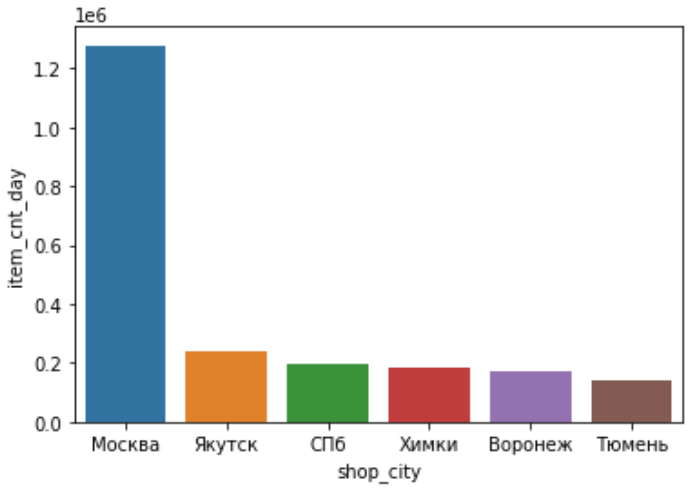
4.2 Aggregating daily sales to monthly sales

Since our goal is to predict the total sales for each product in the next month, we would need to add a column aggregating daily sales to monthly sales . By grouping data by month, we generate two additional columns – monthly average price and monthly median price

4.3 Merge potential useful features from supplemental files

- Extract city name and convert them to dummy variables

  The shop.csv includes shop_id and shop_name. We found that each shop name is mainly consisted of two parts: city name and the shop center name.
  The graph on the right side depicts daily sales by cities. We can see that sales may be highly correlated with cities, so we would extract the city name from shop name and convert them to dummy variables (30 dummy variables in total) as part of the input features.



- Convert items categories to dummy variables

  Categories is also one of the input variables that might affect the sales prediction, so we also convert the 80 item categories to dummy variables.

After adding these features to the original training dataset, we end up with a new training dataset called "train_by_month_dummies" consisting of 124 columns. The table below shows the first 11 columns:

	shop_id	item_id	date	date_block_num	item_price_mean	item_price_median	item_cnt_month	month	year	item_category_id_0	item_category_id_1
8113	2	31	2015-10-01	33	399.000000	399.000000	1.000000	10	2015	0	0
8205	2	486	2015-10-01	33	300.000000	300.000000	3.000000	10	2015	0	0
8254	2	787	2015-10-01	33	420.000000	420.000000	1.000000	10	2015	0	0
8284	2	794	2015-10-01	33	3300.000000	3300.000000	1.000000	10	2015	0	0
8367	2	968	2015-10-01	33	58.000000	58.000000	1.000000	10	2015	0	0

5 Modeling

5.1 Split data for training and validation

We first separate the "train_by_month_dummies" into a training set and testing set. Since our goal is to predict sales for Nov. 2015, using the last month (Oct. 2015) data for testing the out-of-sample RMSE of each fitted model would be appropriate.

We have 2 versions of splitting data, one of which has each category and city converted to dummy variables, another has category and city as categorical variables. The reason for that is that we found using dummy variables actually gives us worse RMSE than using categorical variables when fitting a random forest.

Also, when fitting models, we found that there is no difference in using date_block_num and the combination of year and month as input features, so the splitted data does not include month, year and date columns.

5.2 Fit a linear Regression

We first fit a most simple and commonly used model- Ordinary Least Squares (OLS) regression. Using all 121 features for prediction, the OLS gives us a good RMSE, which is **2.31799**.

5.3 Fit a Lasso Regression

The reason why we use all features to fit the OLS regression is that we don't know which features may be irrelevant and what's the relative importance of each feature. That's when Lasso regression comes in handy. The advantage of the LASSO is that it does both feature selection and shrinkage.

When fitting the model, we use stepwise model selection to choose the tuning parameter with the lowest RMSE. We run 5 lasso regressions, with varying levels of alpha. It turns out that the best Lasso Regression has a parameter of 0.0001, which is pretty small, meaning that we do shrinkage on features' coefficients weigh more than selection. This best Lasso Regression achieves a RMSE of **2.3163**, which is slightly better than OLS regression.

The table shows the coefficients for part of features in ascending order. There are 45 features with a coefficient of 0, meaning they are excluded from the model.

	features	coef
63	item_category_id_58	-0.805435
16	item_category_id_11	-0.767382
48	item_category_id_43	-0.694549
64	item_category_id_59	-0.613194
62	item_category_id_57	-0.544511
...	...	...
94	city_code_5	3.931638
87	item_category_id_82	6.581857
84	item_category_id_79	7.042619
14	item_category_id_9	7.167078
76	item_category_id_71	16.247974

120 rows × 2 columns

5.4 Fit a Random Forest

While Lasso is a still a modification of linear regression, we want to fit a nonlinear regression, random forest specifically, to see whether it can improve the accuracy. Random forests arrive at their predictions by fitting many trees from bootstrap replications and aggregate over their predictions, which helps reduce variance. Also, they additionally sample from the set of features to reduce the correlation of the predictors.

However, the random forest fails to improve RMSE since we get a weigh higher RMSE – **6.6639**.

5.5 Final Model

Based on the RMSE of the 3 models we trained, we decide to use Lasso Regression as our final model. The model is a linear combination of 76 features, each of which has a coefficient reflecting their relative importance.

6 Advantages

6.1 Interpretability

Unlike advanced learners, Lasso regression allows us to explore the relative importance of the features. This is particular useful in business settings. For example, if we know that the sales are highly correlated with shops, then we should think about expansion of those shops with higher sales .

6.2 Simplicity

Although Lasso is within the scope of linear regression, it gives us decent prediction for the next month's sales. The algorithm for lasso not only gives us the best linear model (through shrinkage and selection) but also takes much shorter time to train a model than those advanced learners such as random forest and neural networks.

7 Further Improvement

7.1 Use modified versions of random forest

Since Random Forest is a fully nonparametric predictive algorithm, it may not efficiently incorporate known relationships between the response and the predictors. However, if we run Lasso before Random Forest and train a Random Forest on the residuals from Lasso, the Random Forest is able to incorporate known relationships between the responses and the predictors.