Statistical Methods for Data Science Final Group Project

Team 8
Member: Wenhan Ji, Yu Ruan, Huanzhen Zhang, Hao Fu

1.Introduction

Our project is based on the competition, Store Item Demand Forecasting Challenge, on Kaggle (https://www.kaggle.com/c/demand-forecasting-kernels-only). We are given 5 years of store-item sales data and asked to predict 3 months of sales for 50 different items at 10 different stores.

In the project, we have tried time series methods Arima to explore the data. Also, we have trained machine learning model with Linear Regression, Ridge Regression, Lasso Regression, KNN, Decision Tree, Random Forest and Gradient Boosting.

2.Description of the data set

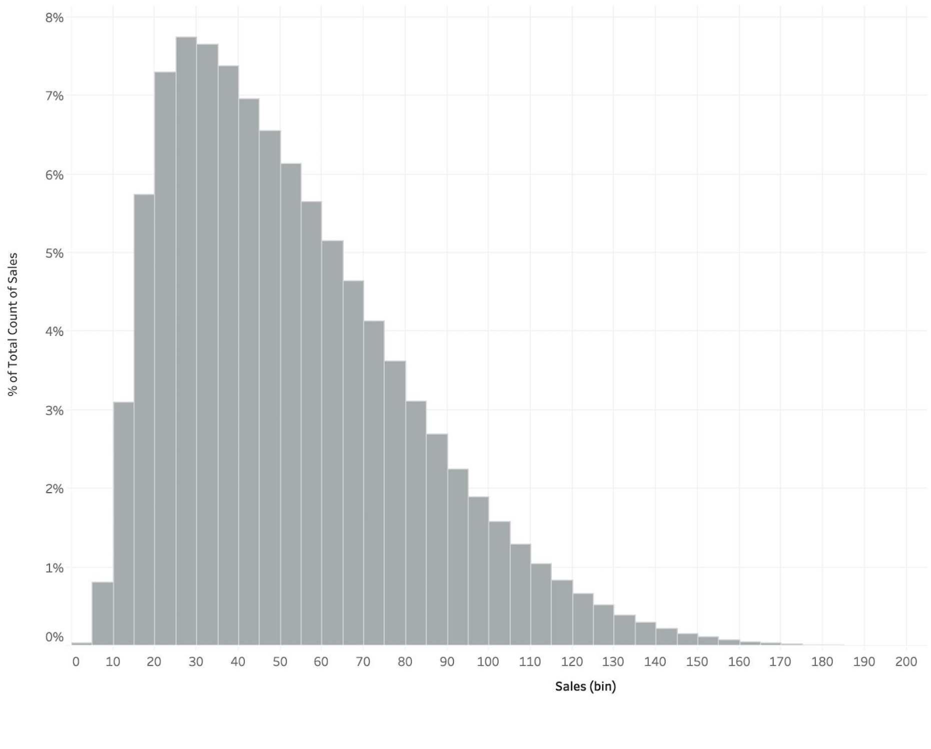
Names of variables with their description:

- date - Date of the sale data. There are no holiday effects or store closures.
- store - Store ID
- item - Item ID
- sales - Number of items sold at a particular store on a particular date.

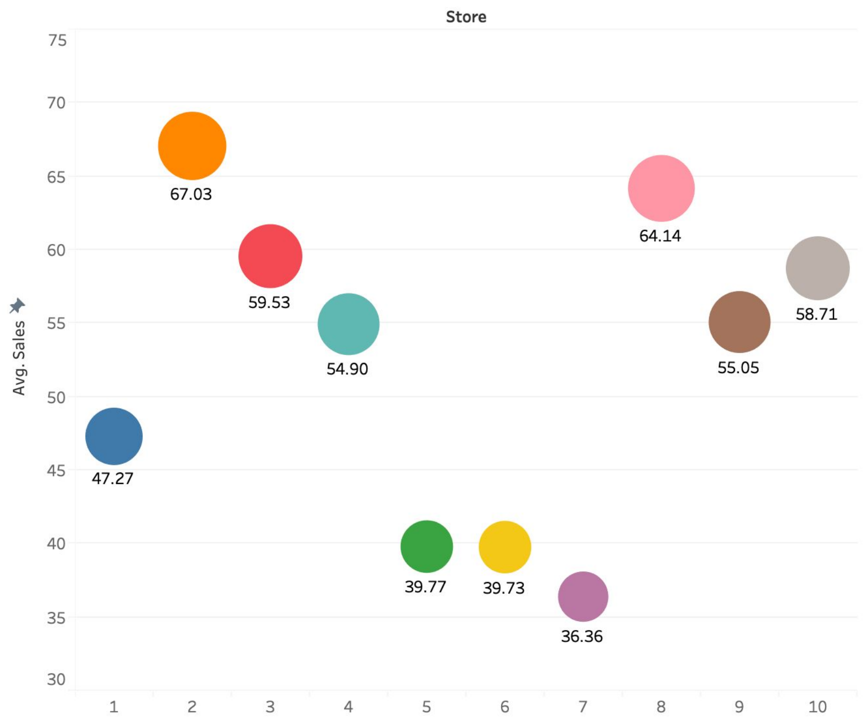
Entries (Train / Test) : 913000 / 45000
Stores (Train / Test) : 1 - 10 / 1 - 10
Items (Train / Test) : 1 - 50 / 1 - 50
Dates (Train / Test) : 2013-01-01 - 2017-12-31 / 2018-01-01 - 2018-03-31

3.Exploratory Data Analysis

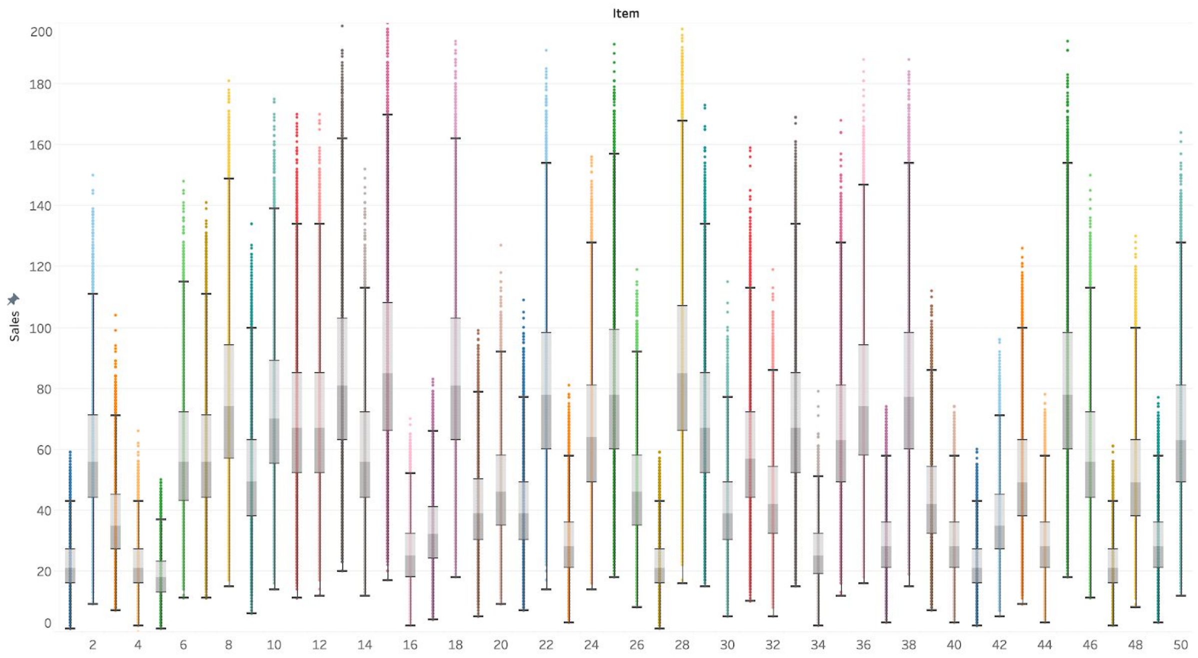
We perform data visualization methods to explore and discover the patterns of original data. First, we plot the sales distribution graph to see which interval that most sales' values locate. From the bell-looking graph, we can conclude that the range of sales is within the interval [0,180], and most sales values are in [15,100].



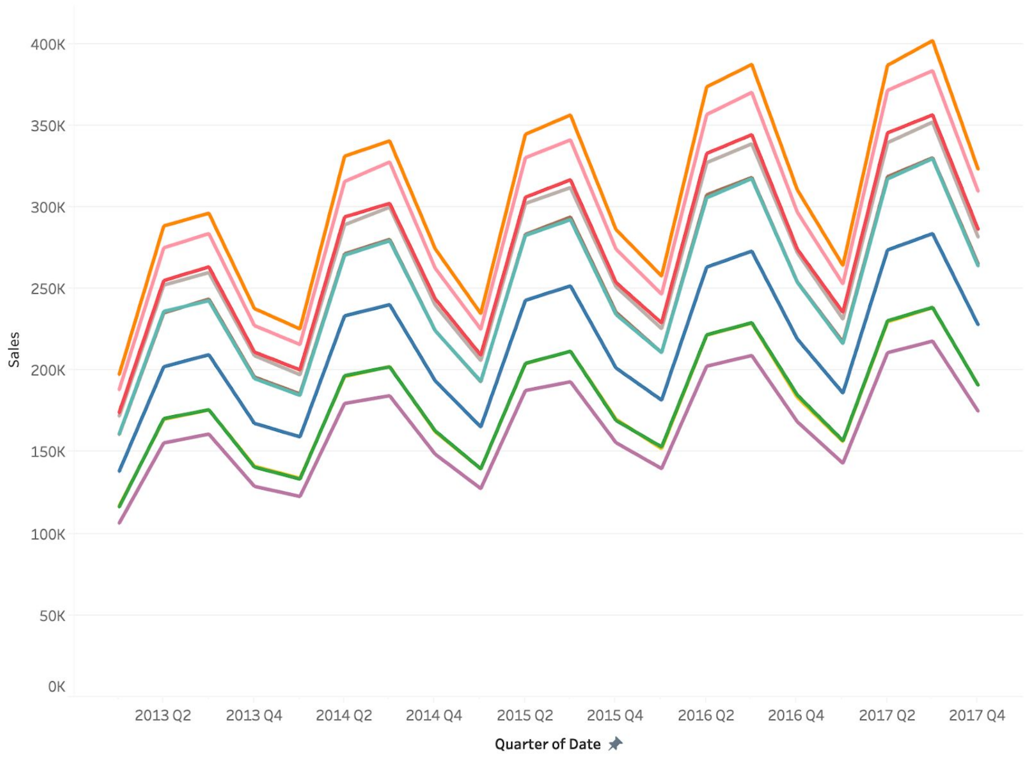
The original data is further analysis based on items and stores respectively. The scatterplot illustrated below shows information about each store and their sales. The graph shows that store 2 and store 8 achieve the largest sales, while store 7 has the least sales revenue.



We then delve into the item-based analysis. The boxplot shows every items' sales distribution. It is clear that items' sales distribution varies a lot. For example, item 28 sales distribution range from 5 to 200, while for the item 5, the sales vary from 0 to 50.



Then, we plot the data in time basis, the graph below indicates that the sales trends of each stores are same. Although, the sales vary a lot because some seasonality reasons, the sales avenues increased by each year.



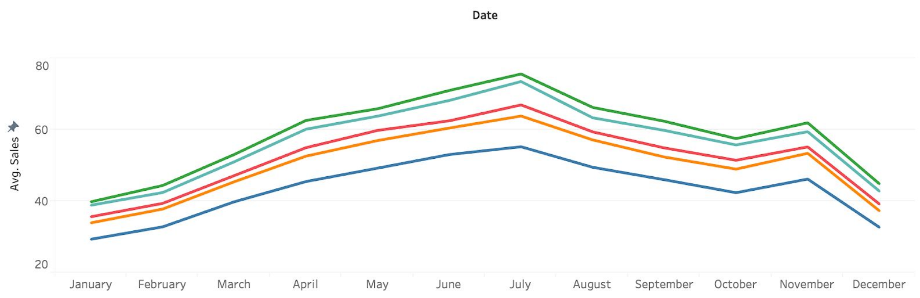
Finally, we explore the seasonality on three facts (i.e. week, month and quarter) within each year, and then plot and combine them into the graph illustrated below.

From the chart, we can draw conclusion that on yearly basis, the largest sales data achieved in July and the sales value is low in December and January. There is a sales rebound occur in November.

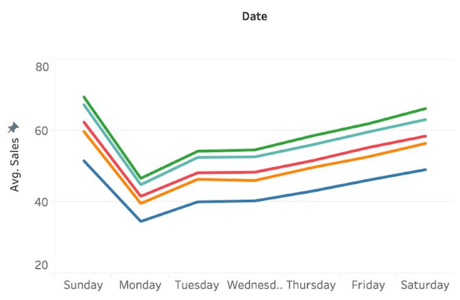
From the graph (average sales by weekday), it is clear that Monday witness lowest sales revenue, and the climax of sales value occurs on weekends.

From the chart (average sales by quarter), we can see that the quarter 2 and quarter 3 have the largest sales value but it is a stagnation periods for sales.

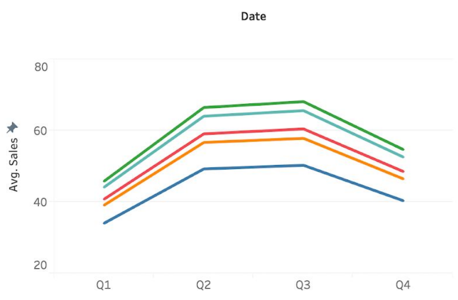
Average Sales By Month



Average Sales By Weekday

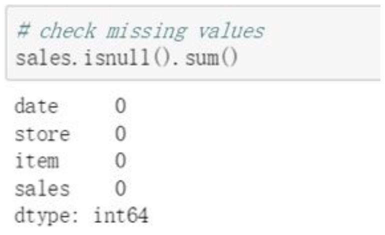


Average Sales By Quarter



4.Time Series Analysis with Arima

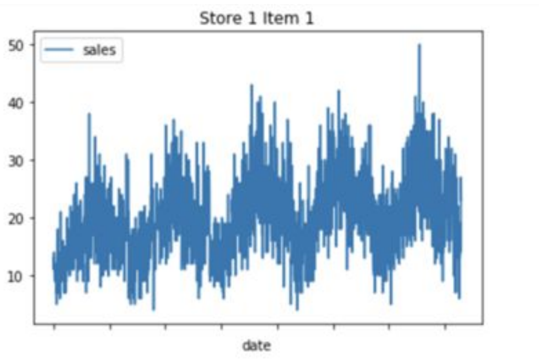
In terms of whole data, we find this dataset is perfect without missing value:

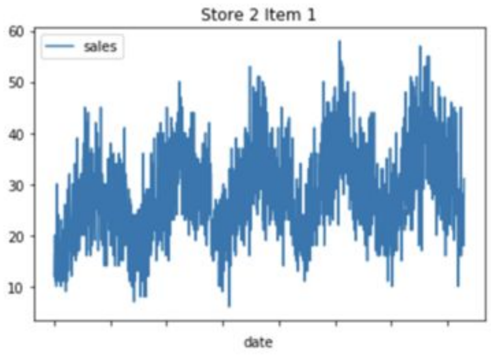


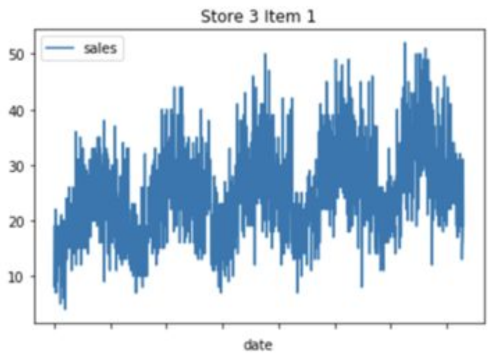
```
# check missing values
sales.isnull().sum()
```

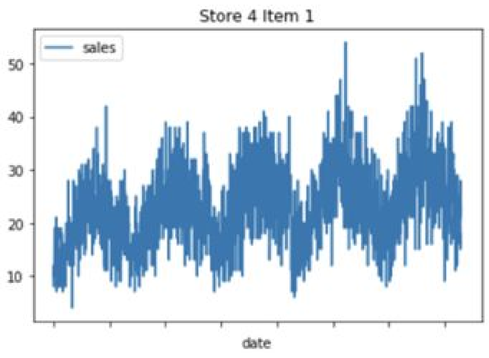
```
date     0
store    0
item     0
sales    0
dtype: int64
```

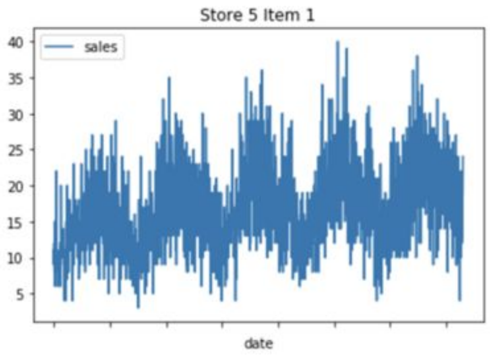
After finishing data check, we want to make a visualization of the changement of sales for particular item in some stores, for example, here we plot item one's sale in some different stores:







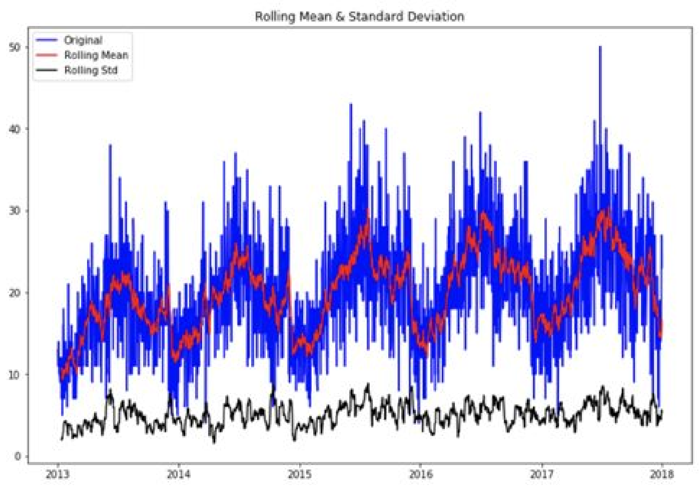




It's obviously that with the increasement of date, the trend of sales of item 1 increase in different stores.

Secondly, we are going into time series part:

For a more accurate assessment there is the Dickey-Fuller test for testing stationary.



```
Test Statistic                  -3.157671
p-value                          0.022569
#Lags Used                      23.000000
Number of Observations Used   1802.000000
Critical Value (1%)             -3.433984
Critical Value (5%)             -2.863145
Critical Value (10%)            -2.567625
dtype: float64

0.022569380626571 81
```

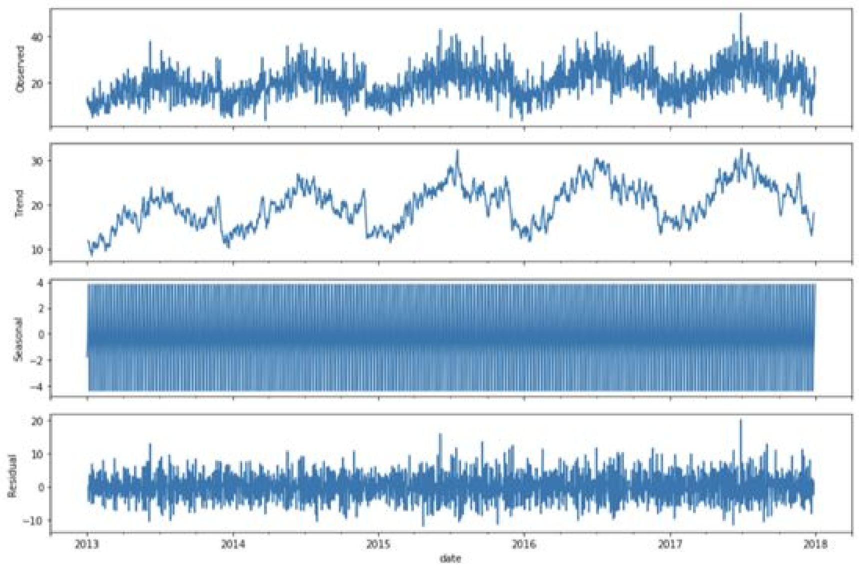
Here we can see the value of p-value is about 0.022. Actually it's not bad and as we know, the smaller p-value, the more likely it's stationary.

P-value is 0.022 which is less than 0.05, so we reject the null hypothesis.The original time series is stationary.

And then, as we have proved this time series has already stationary, the next step is to choose a difference to make stationary, there is no doubt that the value of d will be 1

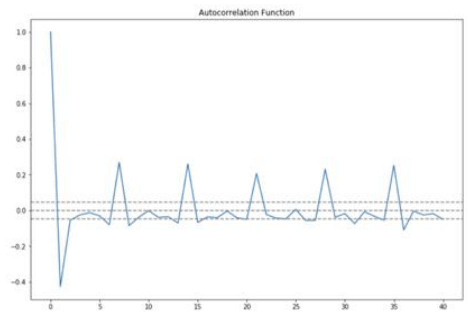
```
The order of difference is:  1
```

Decompose the example time series into trend, seasonal, and residual components.



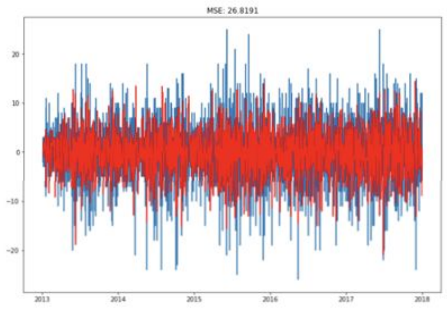
We can see here has a obvious pattern yearly seasonality. The first plot is true data, the second is the trend, we can see trend is increasing by years.

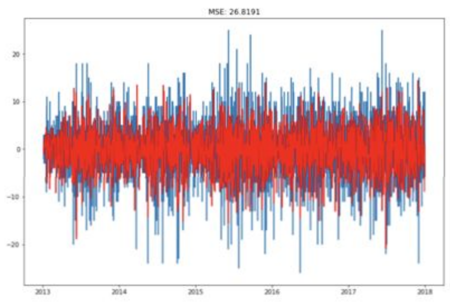
Next step is to determine the p and q, using ACF and PACF.

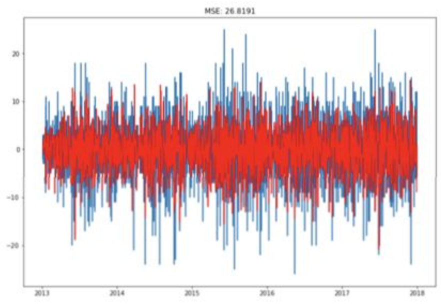


Now the parameters are determined with (3,1,8).

Here we can get the plot of ARIMA:



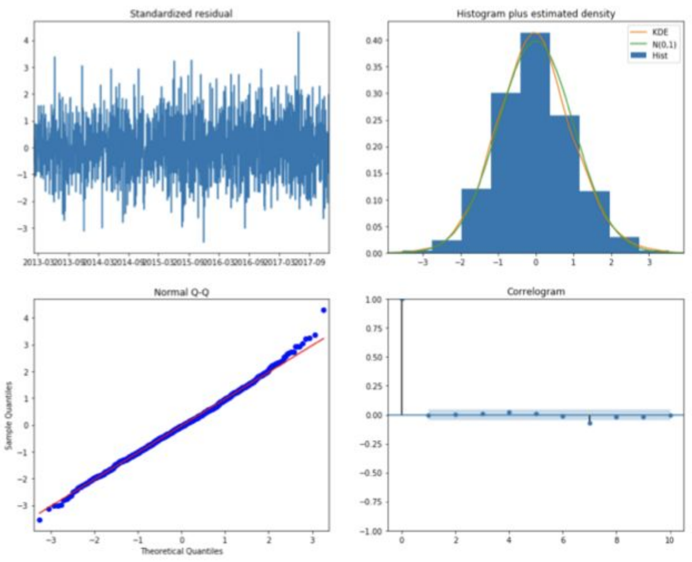




By observing these three plots, the red part is estimated by arima model and the blue part is the true observation, it's obviously that red part fit really well in blue part, it means the model is not bad.

Finally,To see how our model performs we can plot the residual distribution. if the residual distribution is normal distribution, it can prove this is a good model.

Also we can see the normal Q-Q plot here, points are almost distribute in the diagonal, it means this model is not bad as well.

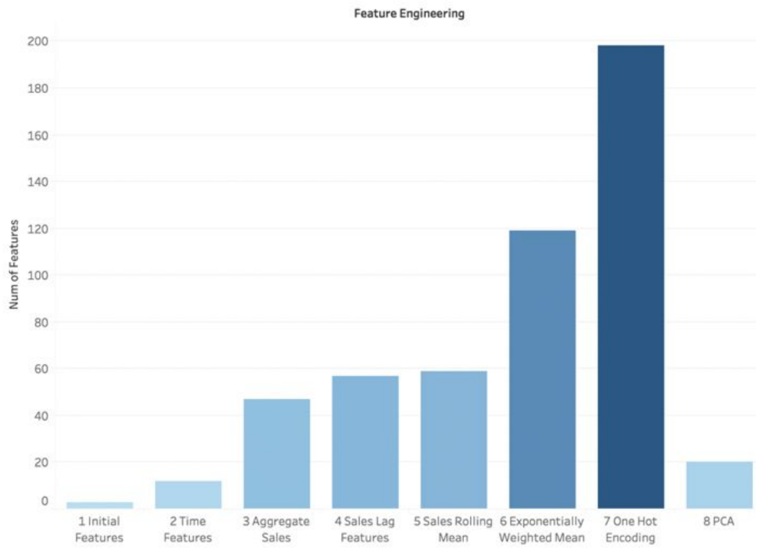


But in our project, we have 10 stores and each store has 50 items. So there are 500 items and each item has a time series sales data from 2013 to 2017. But the kaggle test set contains all these 500 items, and it's hard to tune parameters of Arima model 500 times to predict the test set. So then we tried machine learning model to solve this problem.

5.Feature Engineering

	date	store	item	sales
0	2013-01-01	1	1	13.0
1	2013-01-02	1	1	11.0
2	2013-01-03	1	1	14.0
3	2013-01-04	1	1	13.0
4	2013-01-05	1	1	10.0

The table above shows the original train dataset, which have 3 predictors in X and 1 sales target y. In the project, we computed different features based on it and then get the dataset for model training.



The bar plot above shows how the number of predictors increases after each step of feature engineering. We start from 3 initial predictors and increases it to 197 by different kinds of computing, then apply PCA to decrease the dimension to 20.

Here we will describe detailed description of each step of feature engineering computation.

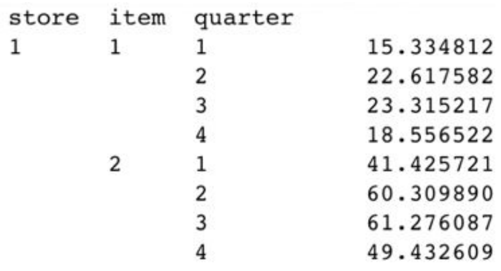
5.1 Time Series Features

	date
0	2013-01-01
1	2013-01-02
2	2013-01-03
3	2013-01-04
4	2013-01-05

dayofmonth	dayofyear	dayofweek	weekofyear	month	quarter	year	is_month_start	is_month_end
1	1	1	1	1	1	2013	1	0
2	2	2	1	1	1	2013	0	0

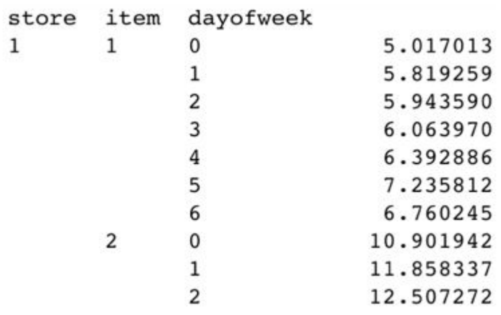
First step, we extract different time features from the date information. Here we get the predictor of month, quarter, year, it month start, is month end, day of week, day of month, day of year, week of year.

5.2 Aggregate Sales Features



```
store  item  quarter
1      1     1           15.334812
             2           22.617582
             3           23.315217
             4           18.556522
       2     1           41.425721
             2           60.309890
             3           61.276087
             4           49.432609
```

We group by store, item, quarter predictors and compute mean sales of these groups. In this way, we can know what's the mean sales of store 1, item 1 in quarter 1.

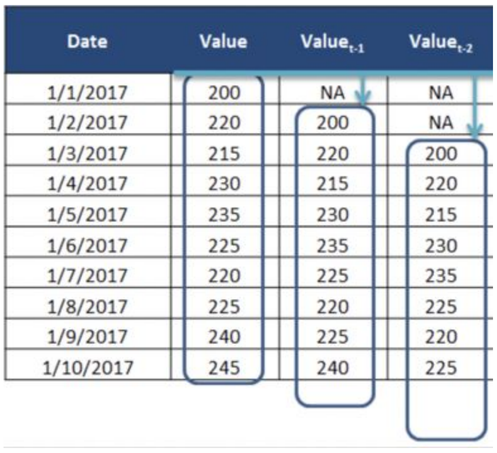


```
store  item  dayofweek
1      1     0            5.017013
             1            5.819259
             2            5.943590
             3            6.063970
             4            6.392886
             5            7.235812
             6            6.760245
       2     0           10.901942
             1           11.858337
             2           12.507272
```

This example shows we group by store, item and dayofweek and computed sales standard deviation in these groups.

In this way, we can computes min, max, mean, median, standard deviation of sales aggregated by different time features.

5.3 Sales Lag Features

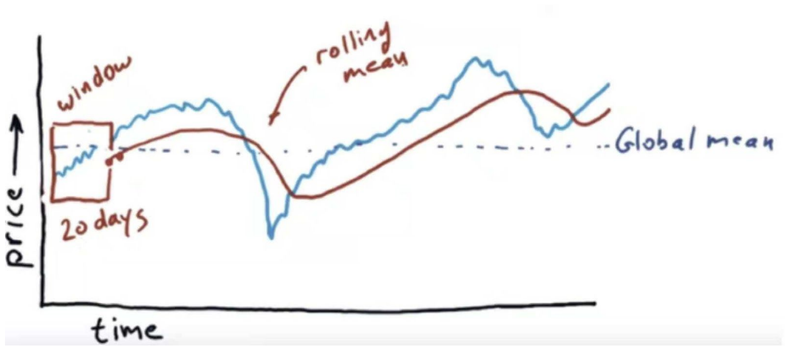


Date	Value	Value$_{t-1}$	Value$_{t-2}$
1/1/2017	200	NA	NA
1/2/2017	220	200	NA
1/3/2017	215	220	200
1/4/2017	230	215	220
1/5/2017	235	230	215
1/6/2017	225	235	230
1/7/2017	220	225	235
1/8/2017	225	220	225
1/9/2017	240	225	220
1/10/2017	245	240	225

For each item, we shift it to get the lag of sales. For example, when we set the lag to be 30 days, we want to know whether sales one month before have an influence on sales today. We can choose different lags to get different sales lag predictors.

We use pandas fillna with next observation to fill the na that the shift created. In this way, we can avoid introduce the trend.

5.4 Sales Rolling Mean



In this step, we move the sliding window from the first timestamp to the last to compute the rolling mean of sales.

$$s_i = \frac{(s_i + s_{i-1} + \ \dots + s_{i-n+1})}{\text{n}}$$

Here we can control the window size and with each window size can get one predictor.

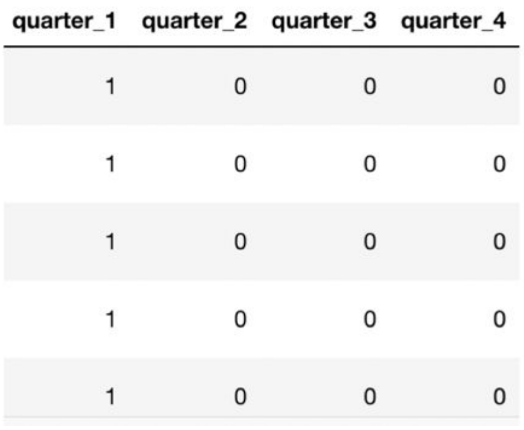
5.5 Exponentially Weighted Moving Average with Lag.

A weighted average of the last n prices, where the weighting decreases exponentially with each previous price/period. In other words, the formula gives recent prices more weight than past prices.

$$s_i = \beta s_i + (1 - \beta) s_{i-1}$$

We can apply this EWM method on different sales lags features computed in step 1.3. Also, we can change the weight to get different predictors.

5.6 One Hot Encoding



quarter_1	quarter_2	quarter_3	quarter_4
1	0	0	0
1	0	0	0
1	0	0	0
1	0	0	0
1	0	0	0

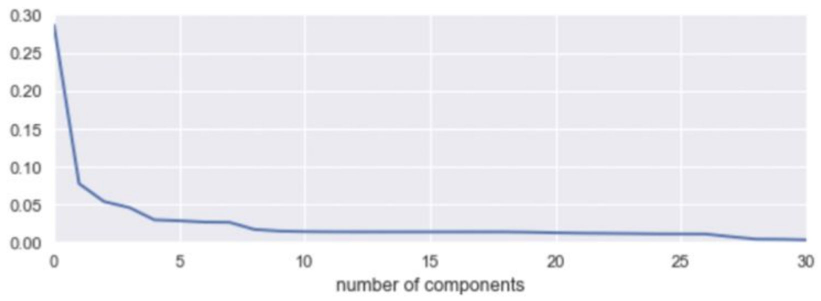
In this step, we apply one hot encoding on categorical features store, item, month, quarter, year. For example, the quarter predictor has been converted to four predictors and the value turns to 0 and 1 to represent whether this row belong to this quarter.

5.7 Min Max Normalization

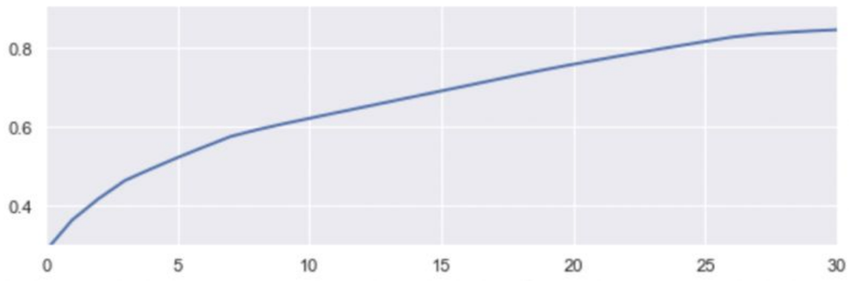
In this step, we transform features by scaling each feature to (0,1). This estimator scales and translates each feature individually such that it is in the given range on the training set

$$\frac{X - \min(x)}{\max(x) - \min(x)} = \tilde{X}$$

5.8 PCA



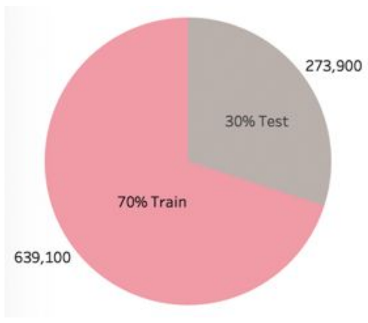
In this step, we apply PCA to our 195 dimensions dataset. Here, we set the number of components to 30. The line plot above shows the explained variance of each component. We can find that the first component explains about 30% variance of original dataset. When it turns to the 20th component, the explained variance rate is about 1%.



In our project, we let n_component to 20 which explain 76% variance of original dataset.

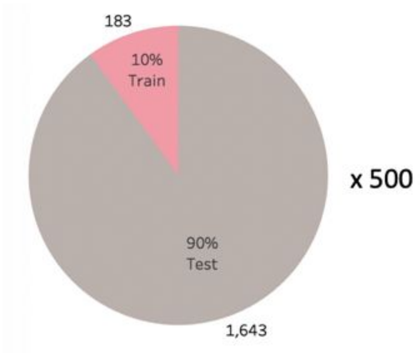
6.Train Test Split

6.1 Initial trial:



Firstly, we set 30% as test data and 70% as train data. So the number of training data is 639,100. But this cause the problem that our model training process is too slow because of too much training data.

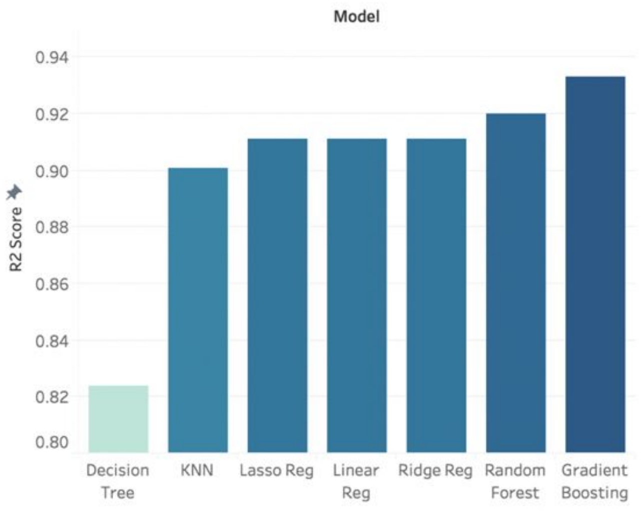
6.2 Modified Version



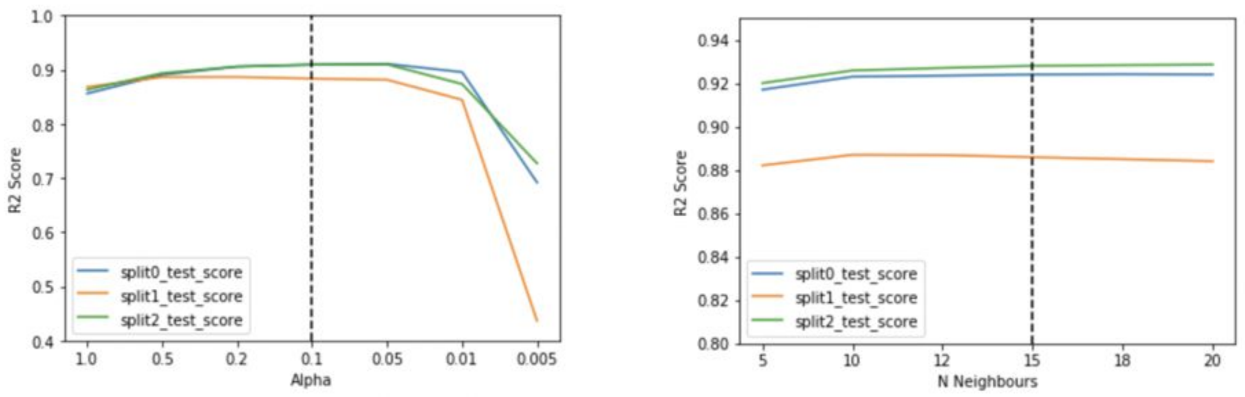
For each item with 1826 rows data, we sample 10% into training set, 90% into test set. So now our training data decrease to 91,000. The model training step turns much faster and we also ensure that each 50 item in 10 store data has been selected.

7.Model Training Result

7.1 Data with Normalization and PCA



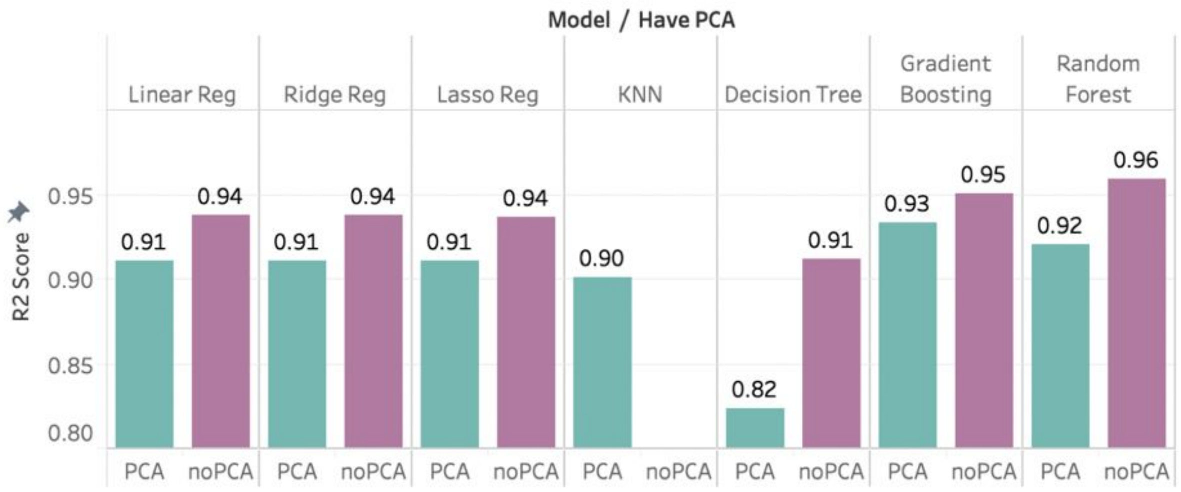
We fit different model to our training data, and the bar plot shows the predicted sales R2 score on test data. We can find that gradient boosting model performs best in this case, followed by random forest. Lasso, Ridge and Linear Regression R2 score on test set all equals to 0.91.



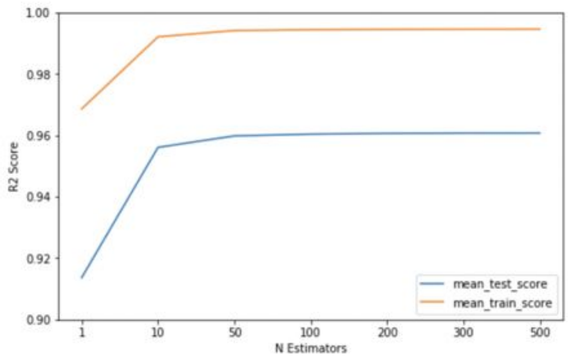
We do GridSearch and K-Fold cross validation to select best parameters for Lasso and KNN. In the plot above, three different colors represent 3 splits of cross validation. The column is the R2 score on validation set, which is 1/3 of training data.

From the plot, we can see that when Alpha equals 0.1, N Neighbours equal 15 two models have the highest mean test score.

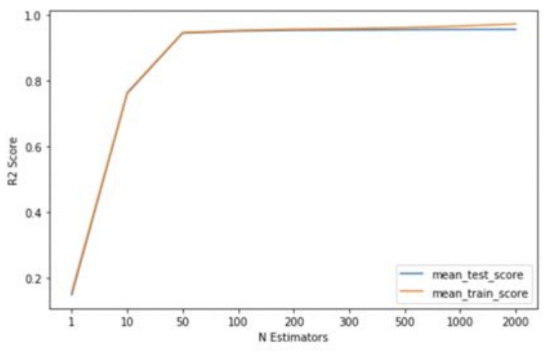
7.2 Data without Normalization and PCA



But Randy in class said that we actually do not have to apply PCA before training tree model. Because tree model has good ability to deal with high dimensional data. So we delete PCA and normalization step and retrain the model. We can see that each model's R2 score increase a little bit except for KNN, which suffers from curse of dimensionality. Maybe because during PCA, we choose n-estimators to be 20 and cumulative explained variance is 76%. In this case, we lose a lot of information.



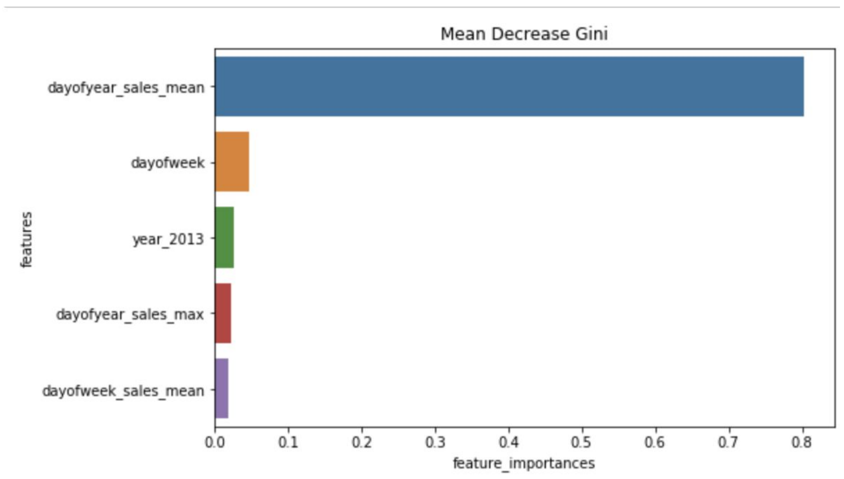
Random Forest search for optimal n estimator



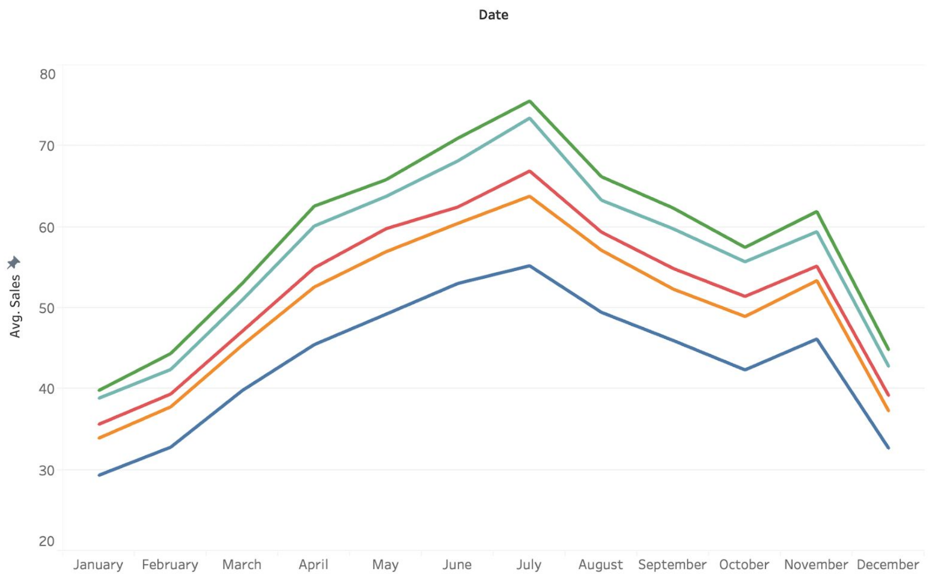
Gradient Boosting search for optimal n estimator

For Non-PCA part, we use grid search and cross validation to find the best number of N Estimators. In the line plot above, the orange line is the cross-validation mean train score and the blue line is mean test score. For each N Estimator, we do 3-fold cross validation. Both of these two model, we can see that initially when we increase n-estimators, the R2 score on train and test both increases. After n-estimator is greater than 100, the model prediction R2 score on test set increases very slow as n-estimator increases.

An interesting observation here is that when n_estimator is 1, random forest r2 score on test set gets 0.91 but Gradient Boosting only gets 0.1. Then we check the document and find that the individual tree in two algorithms has different parameter setting such as max-depth. The max-depth default value of gradient boosting 3 but in random forest, it doesn't give the depth limitation. So their non-linear fitting ability is different, the individual tree of gradient boosting is a more weak learner.

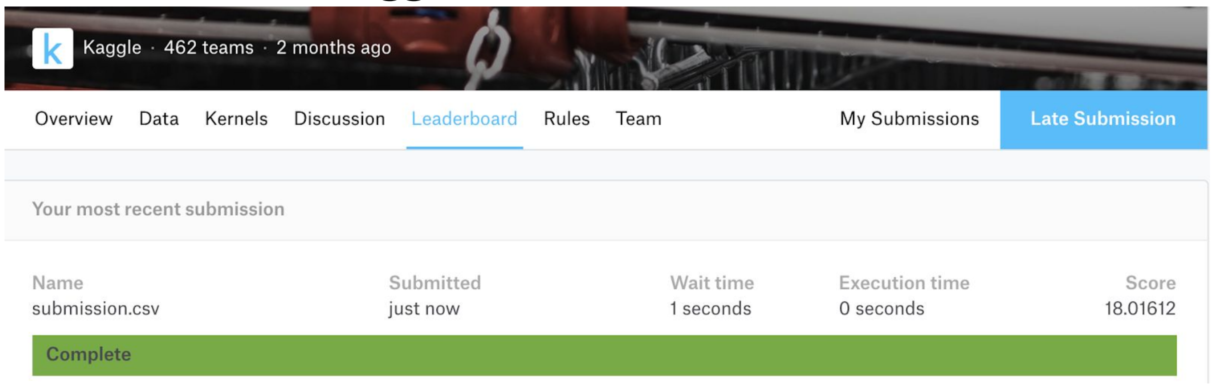


This bar plot shows the feature importance during train our best r2 score model, it shows that the mean sales group by day of year feature is the most important.



Maybe it results from data seasonality. Here the color represents different year. As year increases, the monthly average sales shift upward. So with today's sales value, we can predict next year this day's sales value.

8.Predict on Kaggle's Test Set



Your most recent submission

Name	Submitted	Wait time	Execution time	Score
submission.csv	just now	1 seconds	0 seconds	18.01612

Complete

Here we upload our random forest kernel to kaggle and get the SMAPE error of 18.16.

9.Conclusion

In this project, we get familiar with the basic pipeline to do the kaggle competition. From exploratory data analysis to feature engineering, model training and validating.

Our best model is when use non-pca dataset fitted with Random Forest model with 500 estimators. The R2 score on our own test set reached 0.96, which looks great. But when we predict on kaggle's test set, which prevents data snooping, we know that a great effort we still have to make to beat more teams.