Weather Prediction Analysis of Sales in Walmart

Chenghao Ding - cd4
Yahan Jiang - yahanj2
Yinan Wu - yinanw2
Yue Wu - yue16

Introduction

Walmart owns a huge number of supermarkets as many as 11450 around the world, which spread over 27 countries. Often it is hard for stores' to replenish and manage inventories in advance especially when there are extreme weather events approaching. For example, a soaring increase demand of umbrellas is expected if a big thunderstorm was going to take place. It is disappointing that if you show up in Walmart in a snow storm night but find out the ice melt is out of stock.

However, accurately predicts the level of inventory needed is so difficult that all current state-of-art vendor tools used still can not make sure the store be fully prepared for those extreme climate events. Instead of relying on those ad-hoc and time-consuming process, a systematic efficient method is essential to help Walmart stores' managers better prepare those products for customers. Therefore, this Kaggle competition asks us to predict the sales of 111 potentially weather-sensitive products such as umbrellas, bread, and milk from 45 stores on stormy weather.

In this Kaggle competition, two years' sales data for 111 products sold at 45 different Walmart stores are provided as training data. It contains the dates of sales or weather and number of items sold as well as the corresponding store_nbr and item_nbr. Also, 20 weather stations' weather data are provided to help us identify the severe storm weather conditions for those 45 stores. The "key" csv file connects 45 stores with the 20 weather stations geographically. This weather file includes the stationID, date of observation and the local climatological data, such as the maximum or minimum daily temperature, snowfall, precipitation in inches. Thus, we are given the thorough climate information where the 45 stores are located for those years we are interested in. The NOAA weather documentation are also provided as a guide to explain the weather data. Any day in which more than an inch of rain or two inches of snow was observed is defined as a weather event. We are asked to predict the units sold for a window of ±3 days surrounding each storm. The test data contains the sales date that needs to be predicted at specified stores and items.

There are 4617600 entries in training data, and 526918 entries in test data. It turns out that a lot of value in "units" column is zero, where sales number zero doesn't necessarily mean there was no demand for this product. We also find that some Walmart stores are close to the same weather station. Finally, the competition gives a submission format with a column of "id" included. The "id" is made up of three pieces representing a store_nbr, item_nbr, and date respectively. The prediction results should be in the "units" column and then submitted to Kaggle to check for your scores. The score is evaluated as root mean squared logarithmic error (RMSLE). In other words, we are aiming to build a model get a score as low as possible for best prediction.

Exploratory Data Analysis

The exploratory data analysis is divided into 2 parts, one is the analysis for the 'train' dataset, and the other is the analysis for weather-related data. Discussion in this part will follow such a pattern.

We import the 'train.csv' file into R and notice that there are 4617600 observations of 4 variables, namely 'date', 'store_nbr', 'item_nbr' and 'units'. The range of the variable 'date' is from '2012-01-01' to '2014-10-31', which means the timespan is almost 3 years. There are 45 stores and 111 items. Even the items are numbered from 1 to 111, we should be cautious when dealing with these items since the same number does not stand for the same item in different stores. Below is a brief head of the 'train' dataset.

```
> head(train)
# A tibble: 6 x 4
  date       store_nbr item_nbr units
  <date>         <dbl>    <dbl> <dbl>
1 2012-01-01         1        1     0
2 2012-01-01         1        2     0
3 2012-01-01         1        3     0
4 2012-01-01         1        4     0
5 2012-01-01         1        5     0
```

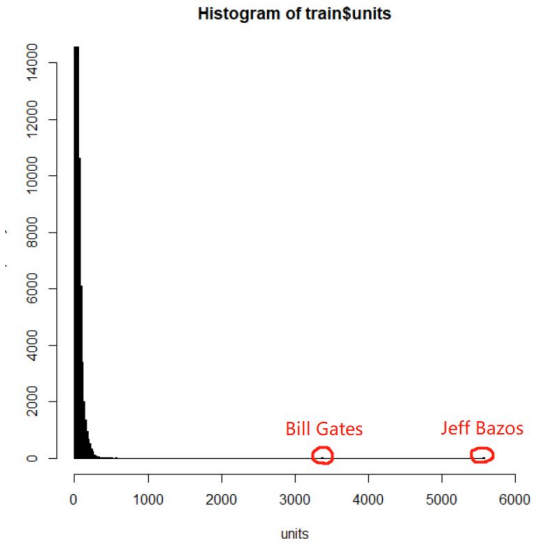
In order to conduct exploratory analysis more accurately, we give a specific number to mark a specific item in a specific store. The method we adapt in this step is to generate a new variable called 'itemid', which is 1000*store_nbr+item_nbr. For instance, the 'itemid' of first item in the first store is 1*1000+1=1001. 'Itemid' can help us distinguish one item from another. Below is a brief head of the 'train' dataset after generating 'itemid'.

```
> head(train)
# A tibble: 6 x 5
  date       store_nbr item_nbr units itemid
  <date>         <dbl>    <dbl> <dbl>  <dbl>
1 2012-01-01         1        1     0   1001
2 2012-01-01         1        2     0   1002
3 2012-01-01         1        3     0   1003
4 2012-01-01         1        4     0   1004
```

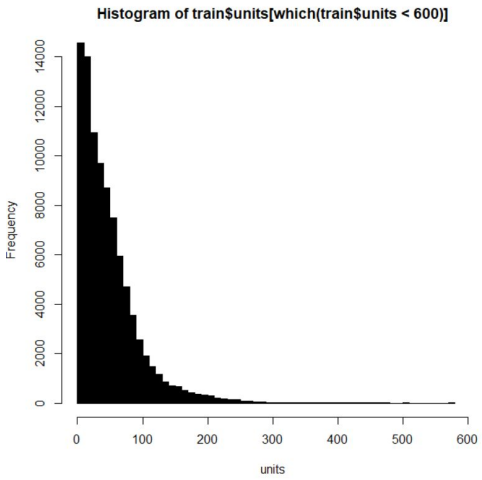
Another feature that we should focus on is the overwhelming amount of 0 in the observations of units. According to the summary, the majority of 'units' in this observation is 0. (4498904 out of 4617600, 97.43%) Considering that not much information in these '0-unit' observations, particularly with a huge waste of computation power, we choose to delete all '0-unit' observations, which leaves us 118696 valid observations. One thing we should notice is that there are 4995 'itemid's before eliminating '0-unit' observations, and only 255 'itemid's left after the cleaning. For over 4000 items not covered in the latter dataset but exist in our 'test' file, we just predict their 'units' as 0.

That being said, we conduct some basic analysis on this processed dataset. The first variable we look into is 'units'. In order to have an initial and figurative impression on the 'units' variables,

we draw a histogram and describe it with some statistics. One thing we should notice is that while most of the observations fall into the range below 600, there are two observations significantly larger than others, with one being 3369 and the other being 5568. These could be outliers, or considering the double numbers of their first and second digits, they might be miscoded. (Or maybe they really take off, like Jeff and Bill.)



The discussion of whether these two observations should be excluded in regressions is left in upcoming chapters. However, with due respect to Mr. Bazos and Mr. Gates, we have to exclude them to take a closer look at the 'common observations'. Among 'common observations', the maximum value is 577 with a single observation, and the minimum value is 1 with 13548 observations. Most of these 'units' are less than 100. （108881 out of 118694, 91.73%）'Units' are more likely to follow a Poisson distribution (theoretically, it should be a Poisson since it is about frequency) rather than a normal distribution. Below is the histogram and distribution.

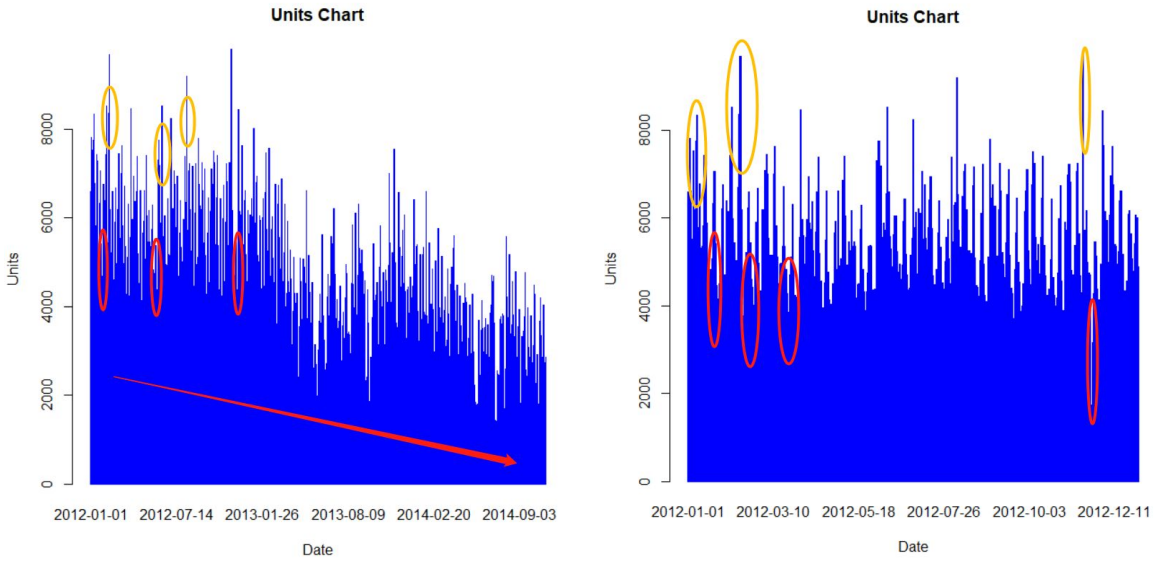



```
> freq_table <- table(cut(train$units,breaks=c(0,100,200,300,400,500,600)))
> freq_table

  (0,100] (100,200] (200,300] (300,400] (400,500] (500,600]
   108881      8394      1250       150        17         2
```

As to the descriptive statistics, some of the indicators might not be reasonably interpreted due to the disobedience of normal distribution. However, when we look at the median of this variable, it fits the conclusion we come up with in the previous paragraph.

Obs.	Mean	Std.Dev	Median	Min	Max
118694	38.32	44.8	24	1	577

When we take a look at this variable from the perspective of time-series, some new discoveries are made. We sum up the number of all items sold in all stores for everyday and draw a histogram for it in a 3-year long period. We notice that, there is a decreasing trend for all items sold in all stores per day in this 3-year long period. If we take a look at the histogram within every individual year, we can notice that both the beginning and the ending of each month always witness a peak for units sold, while in the middle of each month, not that many units are sold. Also, some festivals and holidays should be paid attention to, such as Black Friday, days before Christmas and Memorial Day, etc.

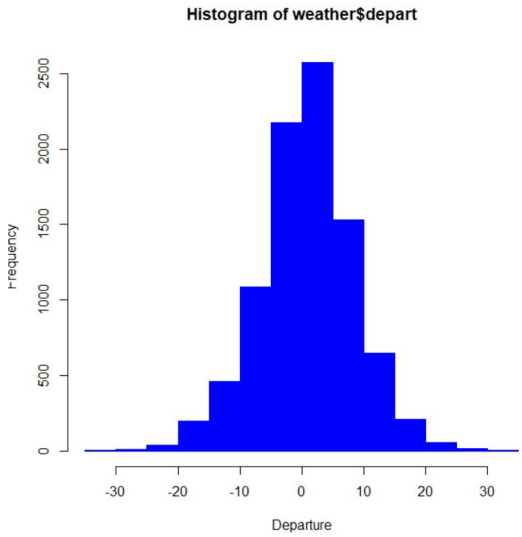


On the left is the chart for ‘units’ from 2012 to 2014, and chart for ‘units’ in 2012 is on the right. All three years share this similar pattern. Such chronological features can be applied in our models. In fact, we indeed use a lot of time features in our following models. We encode the day of the week into two categories, 1 stands for weekend and 0 stands for weekdays. We also encode national holidays as 1, and regular days as 0. Some so-called ‘shopping season’ have several days before the exact holiday also encoded as 1.

The second part of your exploratory analysis is weather related data. There are lots of work for data cleaning before we get to the exploratory analysis. Initially, we have to turn the character data into numeric data, and we can notice there are many missing values, which were represented by 'M'. Also, 'T' in the records of 'Preciptotal' and 'Snowfall' stands for 'Trace', which can also be regarded as 0. With all these works be done, now we take a look at weather data.

One thing we care about is the completeness of the dataset. We notice that for the Station 5, there are only 852 observations, while other 19 stations all have 1035 observations, which is complete for the time period. We are also interested in the distribution of rain and snow and extremities in climate, particularly when it comes to the comparison of historical data.

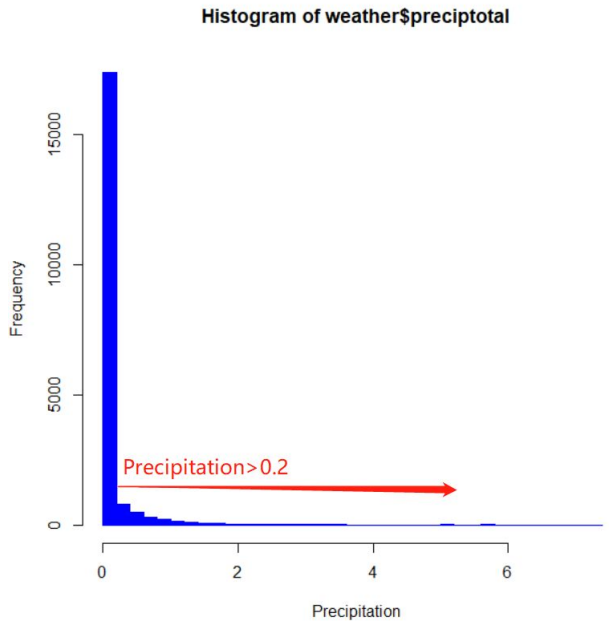
First thing we look into is 'Depart', which is a departure from the long-term average temperature of that day. Theoretically speaking, it should follow a normal distribution with a mean of 0 if we neglect the global warming effect, and it does indeed follow approximately. We assume that extreme change of temperature would affect people's willingness of outdoor activities, and we generate a new variable called 'departlvl' to encode that into two categories, 1 stands for a departure greater than 8 degrees and 0 for less than or equal to 8 degrees. Below is the histogram, distribution and descriptive statistics for 'Depart'.




```
> freq_table <- table(cut(weather$depart,breaks=c(-40,-30,-20,-10,0,10,20,30,40)))
> freq_table

(-40,-30] (-30,-20] (-20,-10]   (-10,0]    (0,10]   (10,20]   (20,30]   (30,40]
        5        45       654      3262      4102       859        74         5
```

Obs.	Mean	Std.Dev	Median	Min	Max
9006	1.36	7.65	2	-35	33

Another thing we are interested is the precipitation, which is represented as 'preciptotal' in this dataset. We also assume that extreme precipitation will affect people's willingness to go shopping. For the majority of precipitations, they are less than 0.2. In this case we generate a new variable called 'Preclevel', 1 stands for precipitation greater or equal to 0.2, and 0 for those are not. In fact, there are only 2349 observations greater or equal to 0.2 among 19657 observations, taking up 11.95%. Below is the histogram and descriptive statistics of 'Preciptotal'.



Obs.	Mean	Std.Dev	Median	Min	Max
19657	0.1	0.34	0	0	7.36

Based on the exploratory analysis we have done, we come up with several potential attempting plans for our model:

- There is a chronological feature in a long term, and monthly feature is quite obvious. Holidays and several days before holidays should be carefully dealt;
- Weekdays and weekends might also be an influential factor;
- We are trying to focus on the impact of extreme weather on sales volume, and we have generated categorical indicators for respective variables.

Linear Regression Model / Diagnostics

We first merge the key data and training data by using the common variable, which is the store number. Then, we merge the new data set with the weather data by the common variables, which are station number and date.

Our goal is to predict the sales of each product around the time of major weather events. The sales of different products will be affected by different weather conditions. For example, an umbrella will sell more on rainy days compared to other products. We decided to use ANOVA to see the difference of the sale for each product. To do this, we create a new variable named itemid. Itemid is created by using the store number times 1000 and then add it with item number. In our first linear model, we choose to use the unit as our response variable. Our explanatory variables are factor(itemid), temperature average (tavg), depart level (departlv), snowfall, precipitation level (preclevel), holidays (date2), and weekdays (week).

There are seasonal trends in the retail industry, like Christmas, Thanksgiving Holidays, and regular sales, so we use holidays(date2), weekdays(week) as our explanatory variables. The sales in the retail industry will not only be affected by seasonal trends, but also the weather. Based on a report, it is said that the weather conditions will affect the consumers' purchasing desire as well as the channel that they buy the product (weatherads, 2019). As a result, we chose to use explanatory variables like temperature average, depart level, snowfall and precipitation level to represent the weather features. However, there are so many missing values in the depart level and snowfall, so we decided not to use these two variables in our model.

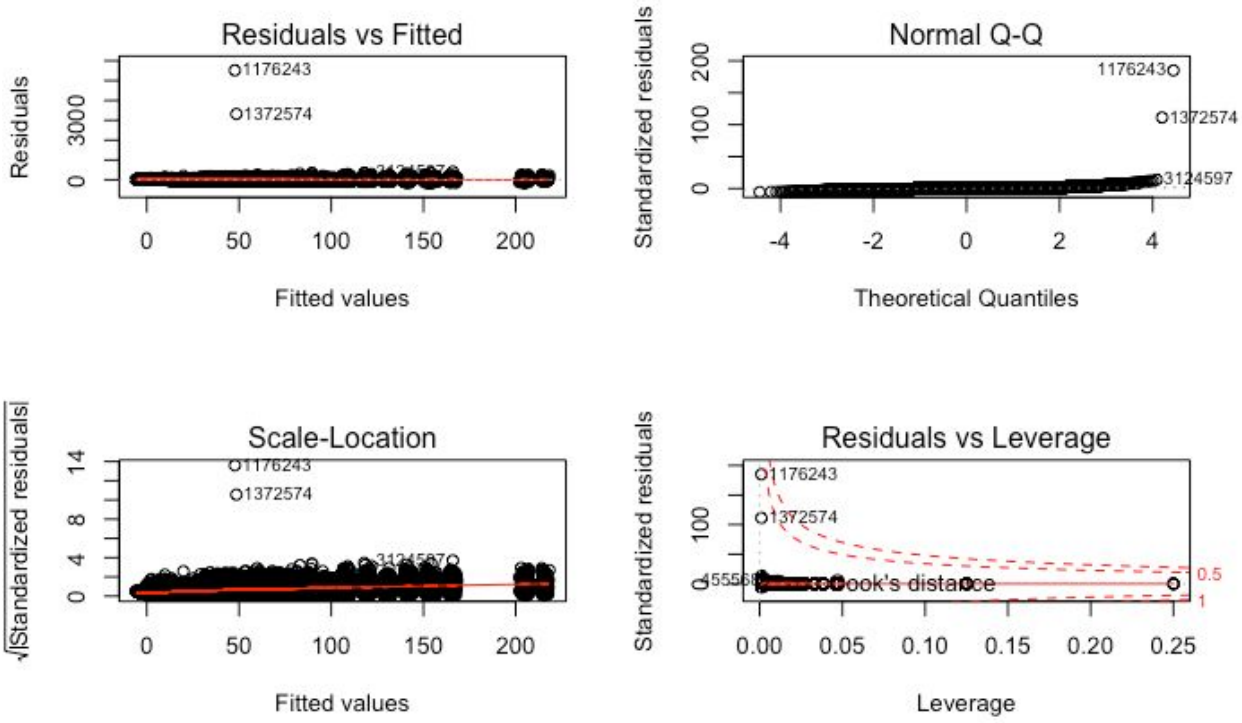
Our first linear model is:

$$\text{units} \sim \text{factor(itemid)} + \text{tavg} + \text{preclevel} + \text{date2} + \text{week}$$

In our summary, the R square is 0.6314, which means that there is 63.14% of the variance for a dependent variable that is explained by the independent variables. The p-value in our summary is less than 0.05, which means that the coefficients of variables are significantly different than 0. The coefficient for average temperature is -0.058926, which means the average temperature increases by one degree, the sales units will decrease by 0.0589. If it is a rainy day, the sales units will decrease by 0.03113. If the day is a holiday, the sales units will decrease by 1.459948. If the day is a weekend, the sales units will increase by 10.625883.

Multiple R-squared	Adjusted R-squared	F-statistic	DF	p-value
0.6314	0.6306	765.2	255, 113892	< 2.2e-16

We then plot the diagnostics plots of our first linear model. From the qq plot, this model meets the normality assumption as it seems to mostly lie on a straight line. There are two points obviously far away from other points, which are 1176243 and 1372574. By using 2p/n as our cutoff, we found that there are 12 high leverage points. If we set the cutoff value of studentized residual as 3, then there are over 65 observations are outliers. If we set the cut off value of Cook's Distance as 0.5, there will be no influential points. Therefore, using 1 as a cutoff, there are also no influential points.



To check the collinearity, we used the VIF function. Based on the results, all predictions have VIF less than 5, so we concluded that collinearity is not an issue here. We then did a Breusch-Pagan Test to check the homoscedasticity assumption of our model, and the p-value is less than 0.05, so there is a heteroskedasticity issue existing in our model.

	GVIF	Df	GVIF^(1/(2*Df))
Factor (itemid)	1.355475	251	1.000606
tavg	1.348575	1	1.161282
preclevel	1.018010	1	1.008965
date2	1.027918	1	1.013863
week	1.009858	1	1.004917

Our first linear model scoring 0.26788 on Kaggle.

Improvements

Outliers

To improve the linear regression model, we should first solve the issue of outliers as it is an obvious problem. According to the diagnostics plot of the previous model, two outliers are quite far away from other points and we found them in the dataset. One's units is 5568, the other one is 3369, which is much greater than other units. The units excluding these two outliers are all below 1000, so we considered these two extreme values as input mistakes.

Interaction of week and date2(HOLIDAY)

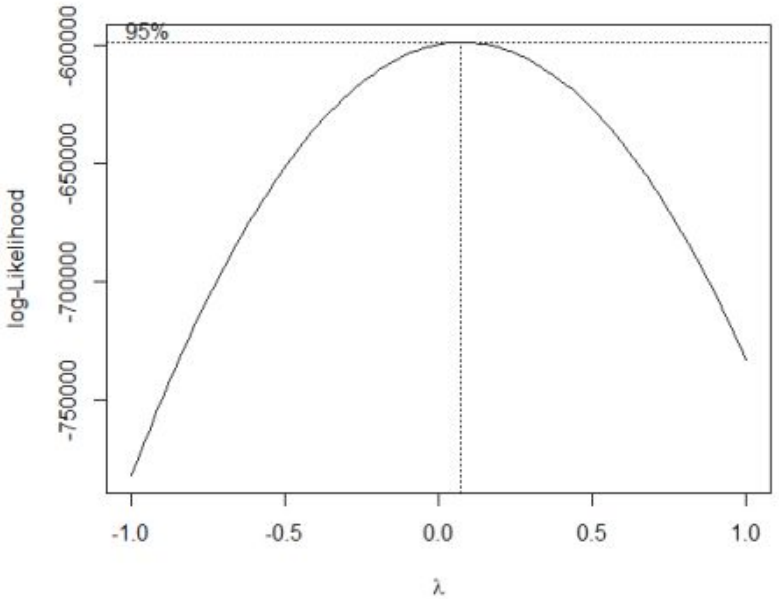
As we mentioned in the previous section, we generated two new variables in terms of date: "week" and "date2". 0 in "week" means weekday and "1" means weekend. 0 in "date2" means non-holiday and "1" means holiday. We have discussed the meaning of using these two variables. In order to further improve the model, interaction of these two variables should be taken into consideration. We made this table to clearly and concisely explain why we would use interaction of these two variables.

Description	Week	Holiday	Interaction
Weekday and Non-holiday	0	0	0
Weekend and Non-holiday	1	0	0
Weekday and Holiday	0	1	0
Weekend and Holiday	1	1	1

The differences between these four scenarios seem to be reasonable in reality, but the fourth scenario is actually not significantly different from the third scenario in reality. Both scenarios mean holiday and the only difference is whether it is weekend. Nobody cares if it is weekend when it is a holiday. Taking Christmas Day for example, people tend to make many purchases between or before Christmas since it is holiday no matter if it is weekend or not. However, without the interaction in the model, the coefficient of "Week" will still be added if the holiday is a weekend and that is not what we expect, which makes the interaction necessary. On the other hand, the difference between the second scenario and the fourth scenario exists since holidays sometimes mean more than just weekends. Some holidays like Thanksgiving motivates people to buy many things while weekends are mostly for buying some daily supplies. The differences between the first three scenarios can be dealt with by coefficients of "Week" and "Date2". What we expect is that the interaction term could counteract the influence of weekend when it comes to the fourth scenario of weekend and holiday.

Tranformation

Transformation is supposed to be made based on the full model. We used Box-Cox to explore the appropriate transformation. Here follows the plot of Box-Cox in R studio.



As we can see from the plot, the confidence interval and estimated point of lambda is around 0, which means log transformation could be a good choice.

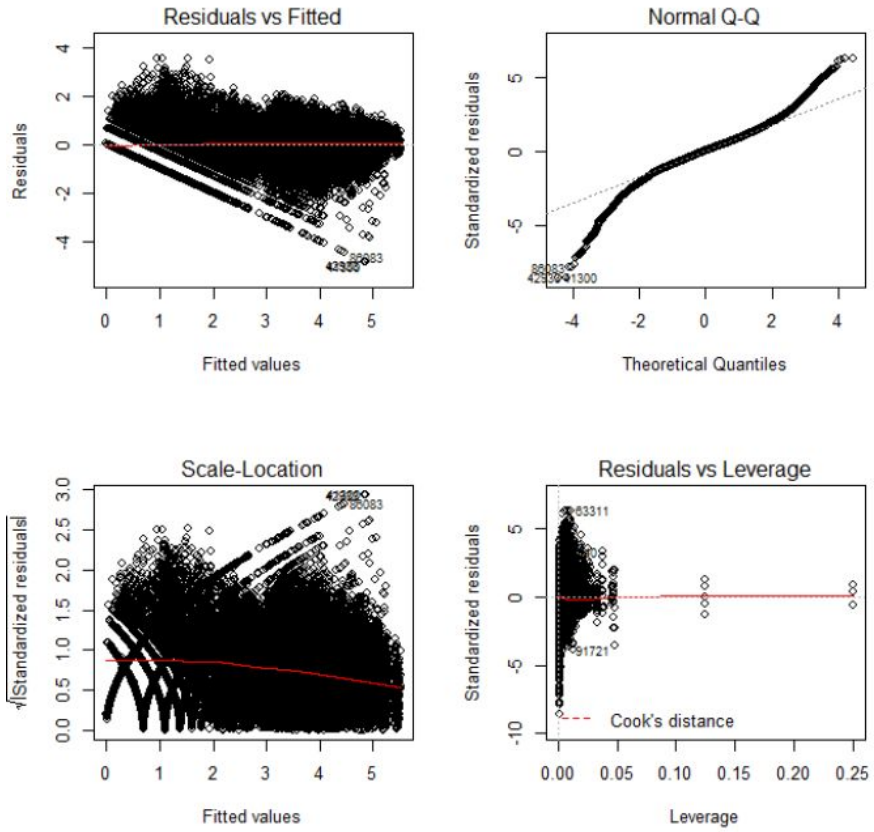
After all these modifications of the model, the first improved model should be:

$$\log(\text{units}) \sim \text{week}*\text{date2} + \text{factor}(\text{itemid}) + \text{preclevel} + \text{tavg}$$

Then we also made the summary and the diagnostics plots of this model. They are as follows:

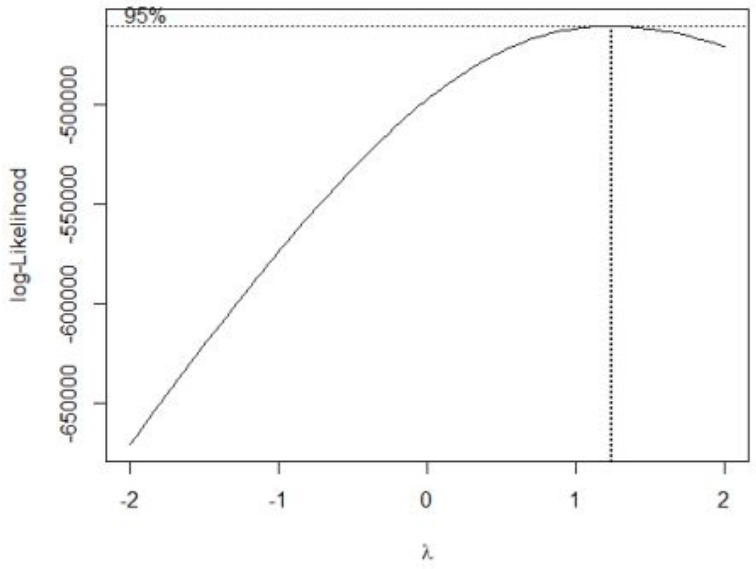
Multiple R-squared	**Adjusted R-squared**	**F-statistic**	**DF**	**p-value**
0.8705	0.8702	2991	256, 113889	< 2.2e-16

The R-square looks pleasant and the p-value of the general model shows that the model is significant. According to the long list of the variables (because of the large quantity of factors), all of the predictors are significant.



Even though the two extreme outliers have been removed, there are still many residuals whose absolute value is greater than 3. We finally decided not to remove them or the model cannot make predictions in test file. The points in QQ plot seem not to look like a straight line, we can hardly consider the residuals to be normally distributed. According to the distribution of the points, we doubt that the issue of heteroskedasticity still exists. The p-value of the Breusch-Pagan Test is lower than 0.05, which means homoscedasticity assumption is not satisfied in this model. We also used VIF to check the issue of multicollinearity and the maximum of the VIFs is 2.077766, so there is no problem of multicollinearity.

Although we already made the log transformation, the transformation is based on the full model. In order to make sure that removing the outliers and the interaction term will not influence the transformation, we did Box-Cox again after the modeling. Here follows the plot:



The lambda is around 1 so we will say that no more transformations are needed.

When we used this model to make predictions in test file, the score in Kaggle is 0.21036, which is much better than the full model. We believe the log transformation is what makes the score lower since we actually submitted the prediction every time we made a single modification, so we know the log transformation contributed a lot to the reduction of the score.

improvement1.csv a day ago by Yinan Wu	0.21248	0.21036

AIC/BIC

After we chose to do the log transformation, we used AIC and BIC methods to select the best model. Here's following the model that was chosen under each method:

Aic_backward: log(units) ~ week * date2 + factor(itemid) + preclevel + tavg + week + date2

Bic_backward: log(units) ~ week + date2 + factor(itemid) + preclevel + tavg

Aic_forward: log(units) ~ factor(itemid) + week + tavg + date2 + preclevel + week:date2

Bic_forward: log(units) ~ factor(itemid) + week + tavg + date2 + preclevel

We then did cross validation over these models and got the sum of SSE under each model. For the aic backward and forward model, the sum of SSE is 36681.75, while the bic backward and

forward is 36681.56. In conclusion, we thought that this model: log(units) ~ factor(itemid) + week + tavg + date2 + preclevel is better as it had a lower SSE.

The Kaggle score of this model is 0.22364.

WLS

Since we cannot solve the issue of heteroskedasticity by using any models previously discussed, WLS is considered as a way to solve the problem. The problem is to determine the weight based on the dateset. According to what we learn and what we use in the models as predictors, we have to divide all the observations into groups based on item, date and weather condition (each group has the same itemid, date and weather condition), and then generate the standard deviation as a new variable. However, each itemid on a certain day which has certain weather condition has just one single record, which means standard deviation cannot be calculated. So we just use itemid divide them into groups (each group has the same itemid) and generate the standard deviation. Then we know all the observations' standard deviations and the number of observations in each group. We tried to use n*i*/sd^2 as the weight of the model but things did not become better. Actually the R-square becomes lower and Breusch-Pagan Test was still not passed. Here follows the results of the WLS model.

Multiple R-squared	**Adjusted R-squared**	**F-statistic**	**DF**	**p-value**
0.7716	0.7711	1503	256, 113890	< 2.2e-16

The failure of WLS might be because of the method we generated the weights, or WLS is just not suitable for this data. After discussion, we decided not to use this method and try some other ways. We believe Poisson Regression can be a good option.

Poisson Regression

For this dataset, we are going to predict the sales units according to various predictors. As we learned, Poisson Regression can be very useful when dealing with count variables such as sales units. In addition, the log transformation we used above is likely to result in the problem of heteroscedasticity. Poisson Regression made a log transformation while it is also a generalized linear model, which hopefully may solve our heteroskedasticity issue.

The formula we use in Poisson Regression is:

$$\text{units} \sim \text{week}*\text{date2} + \text{factor(itemid)} + \text{preclevel} + \text{tavg}$$

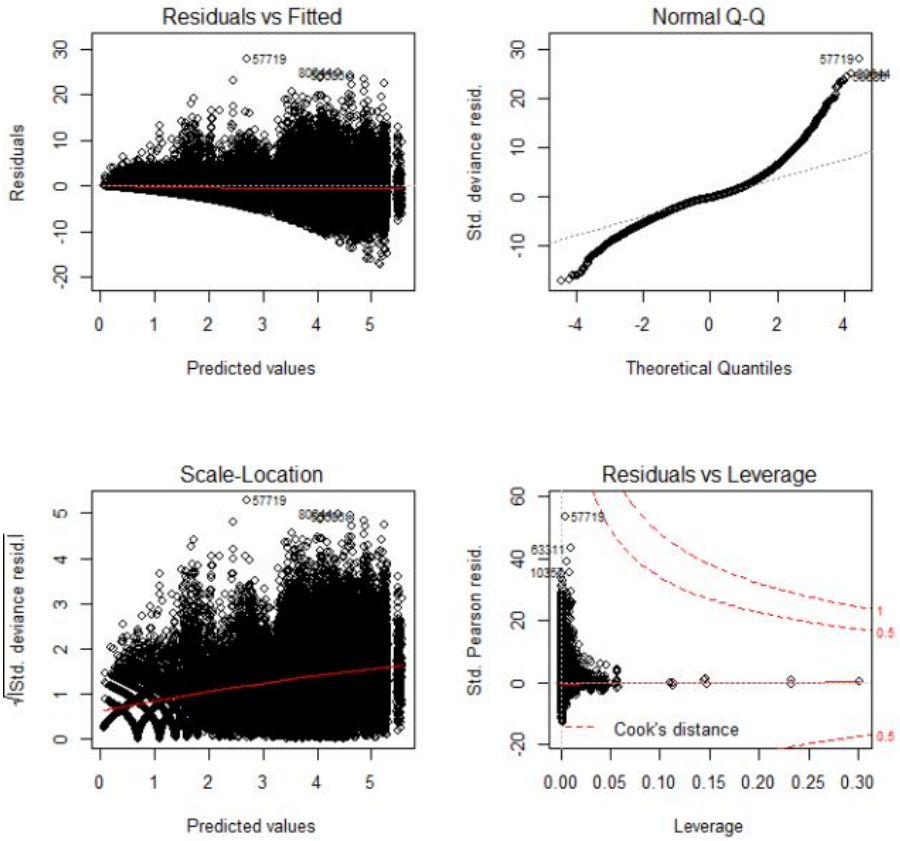
We did not use log transformation here because GLM Poisson Regression in R would do it automatically.

Here is the result of the model:

Null Deviance	**Residual Deviance**	**Null Deviance DF**	**Residual Deviance DF**
5039824	863003	114145	113889

As to the interpretation of this model, the first thing we should notice is the significant decrease of deviance, namely the difference between Null Deviance (ND) and Residual Deviance (RD). ND, as we discussed in class, is an extended version of RSS with no predictors, while RD is the RSS with predictors. Both of them follows a Chi-square distribution. According to the property of additivity of Chi-square distribution, the difference between ND and RD, which is 4176821, also follows a Chi-square distribution with a degree of freedom of 256, ND_DF-RD_DF, which in this case, 114145 minus 113889, and the p-value under the null hypothesis, which refers that the model with no predictors does not significantly differ from our model, is almost 0. No chance for us to accept such a hypothesis. With such a result, we can confidently come to the conclusion that our model is of great significance.

The diagnostics plot is as follows:



According to the plot, we can still clearly see that residuals are not normally distributed and the heteroskedasticity problem no doubt still exists. After using VIF to check for multicollinearity issue, we found that the maximum value of VIFs is 0.8468092, which means no multicollinearity problem exists. However, the heteroskedasticity problem still exists according to Breusch-Pagan Test.

By using this model, we got a score of 0.25622 from Kaggle, which is lower than the first improved model. The score on Kaggle is a good index for us to evaluate our models, but we still want to make a cross-validation to explore which model is better.

poisson.csv just now by Yinan Wu	0.25905	0.25622

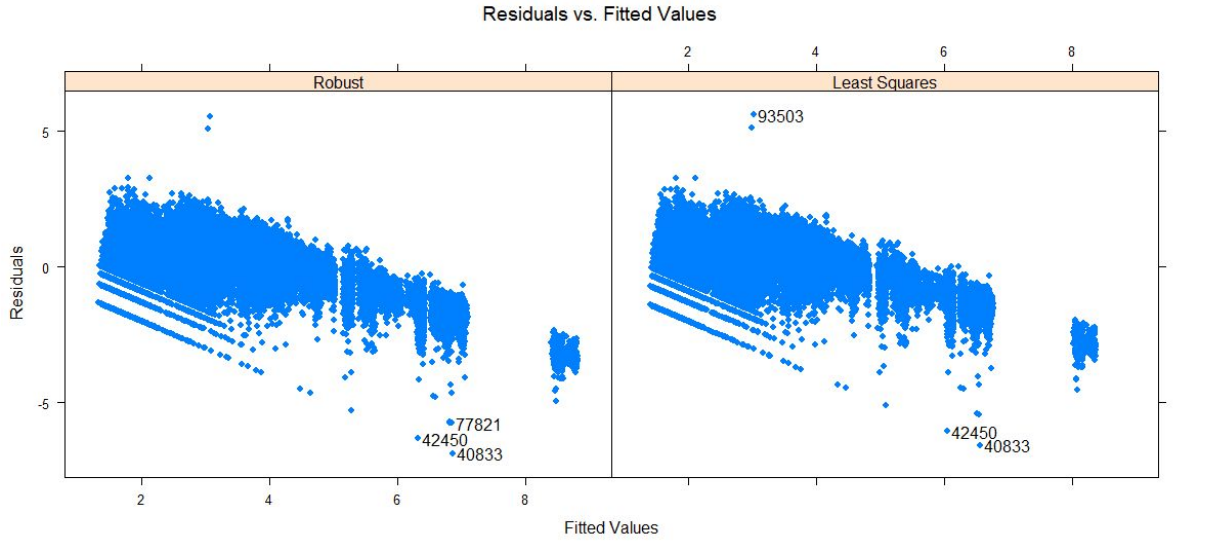
Robust Linear Regression

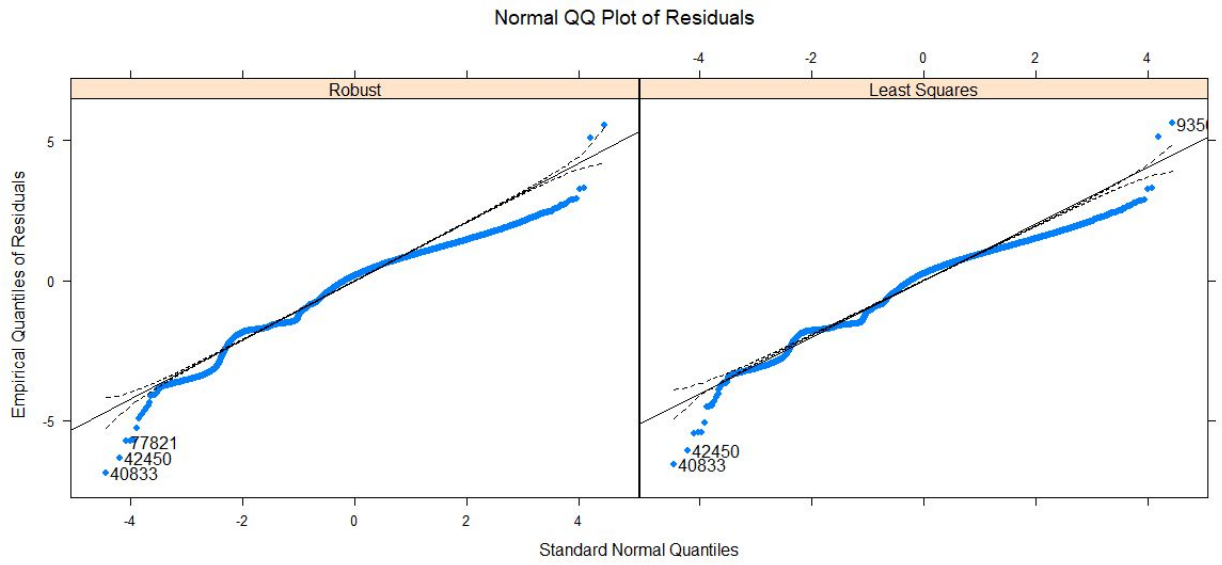
There are so many outliers in our model, so we decided to use the Robust Linear Regression method. Robust Regression works under less restrictions and it will give a better coefficient

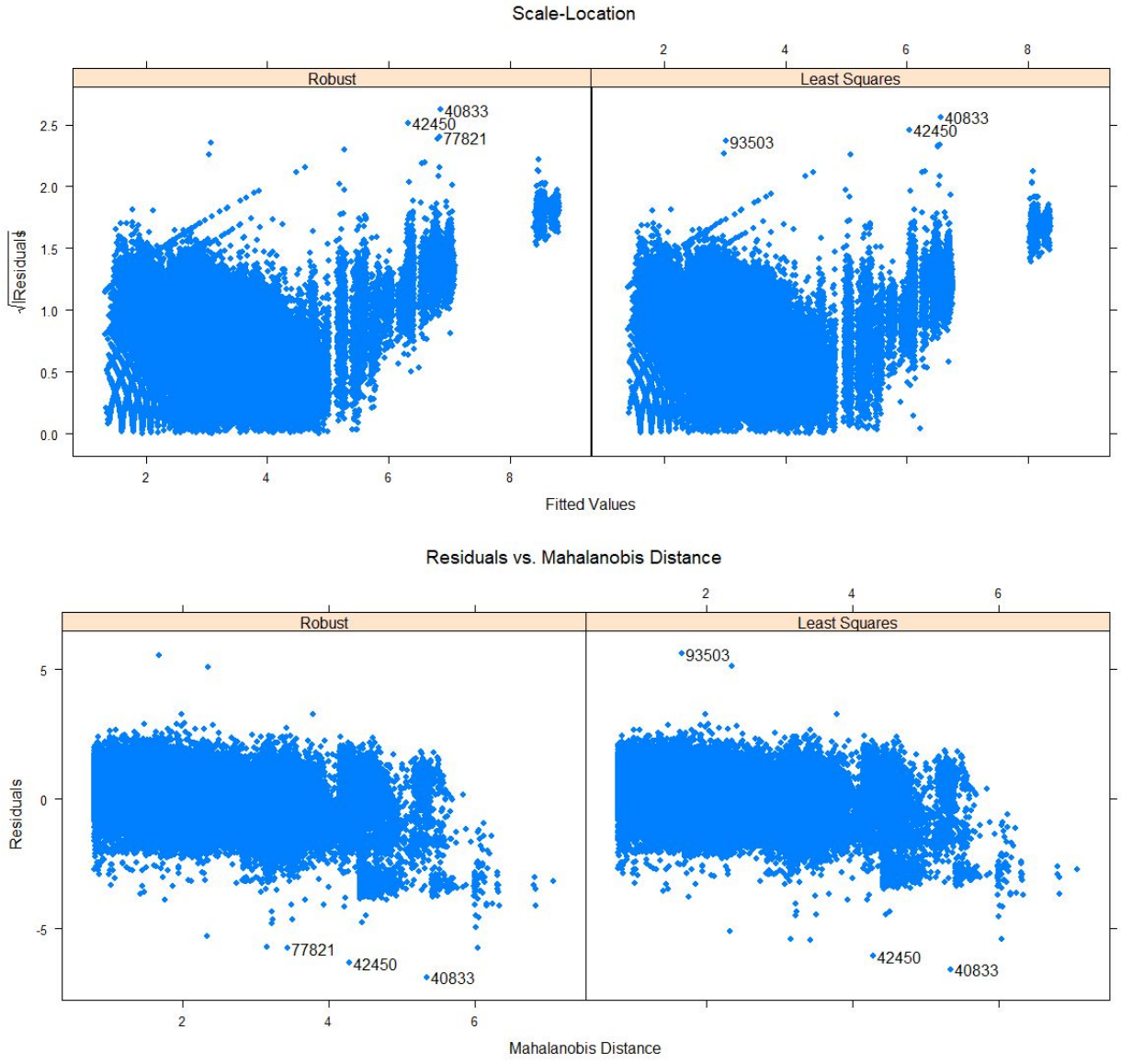
estimations when outliers are present in the data(NCSS Statistical Software). Below is the formula for our Robust Linear Regression model:

rlm(log(units) ~ avgunits + tavg + preclevel + date2 + week)

There are many levels for factor(itemid), which makes it hard to do the robust regression by using this variable. Instead, we used a new variable that is the average units (avgunits) for each itemid to replace it.. By using average units, it can help us with less calculation. Also, this new variable has a similar effect as the factor(itemid). We then use fit.models function to check the goodness of fit for our new model. The right side plot represents the least square model, while the left side represents the Robust Linear Regression model. From the plot, we concluded that our Robust Regression model did not help to improve our model. All plots look very similar to each other, so we decided not to use the Robust Linear Regression model for a better improvement.







Cross-Validation

We used three models in cross-validation, which are the first improved model, the second improved model and the poisson regression model.It does not necessarily mean that the second improved model is better. The difference of the first and the second improved model is only the interaction term. Here lists the three models:

The first improved model:

$$\log(\text{units}) \sim \text{week}*\text{date2} + \text{factor}(\text{itemid}) + \text{preclevel} + \text{tavg}$$

The second improved model:

$$\log(units) \sim week + date2 + factor(itemid) + preclevel + tavg$$

The Poisson regression model:

$$units \sim week*date2 + factor(itemid) + preclevel + tavg\ (family=poisson)$$

The sums of SSE of each model are in the following table:

Model	**First Improved Model**	**Second Improved Model**	**Poisson Regression Model**
Sum of SSE	36681.75	36681.56	40748.85

According to the result, the SSEs of first the second model are not obviously different. The reason might be that the interaction of the weekend and holiday is not really significant. Poisson Regression Model is not better than the other models even though it is generalized linear model. This result is actually consistent with the result given by Kaggle scores though the score of the first improved model is better than than that of the second improved model while the SSE of the second improved model is a little lower than that of the first model. If we have to choose one from them to make predictions, the first improved model will be the best choice.

Even though the scores in Kaggle is not that unpleasant, we still did not solve the problem of heteroskedasticity in this section. Hopefully we could avoid this problem and get better results in the extra models.

Improvements - Extra Methods

Extra Model 1: Linear adjustments on Projection Pursuit Regression (PPR)

As we have discussed in previous parts, there are some chronological features in the 'train' dataset. In fact, with these chronological features alone, we can create quite interesting models. One of them is PPR, Projection Pursuit Regression. Theoretically speaking, PPR model consists of linear combinations of nonlinear transformations of linear combinations of explanatory variables, which sounds very Sheldonic. To be concise, the PPR model is a universal estimator, as it can approximate any continuous function in P-dimension real space, where P is the number of parameters. Or, to be more concise, it can give a very rough approximation of what we want as long as some relevant variables being put into this PPR function. Below is the formula for PPR.

$$y_i = \beta_0 + \sum_{j=1}^{r} f_j(\beta_j^{\mathrm{T}} x_i) + \varepsilon,$$

With that being said, here comes my main idea. As we have discussed, units has chronological features, such as the long-term decreasing trend and monthly features. Therefore, for each item in each store (namely 'itemid' as we discussed), we can establish a PPR model just with variable 'date' to approximately predict the units sold. Such approximations are regarded as a baseline for our prediction. Then we calculate the residuals of these predictions and establish a linear model with time features to simulate the relationship between residuals and time variable. The whole idea is quite similar to GBDT, which is also an ensemble learning. Figuratively speaking, it is like a series circuit, and GBDT has hundreds of lights (trees) being set automatically, while we only set two lights (regressions) manually. The algorithm is listed below, and with parameters calculated from the 'train' dataset, we predict 'units' in the 'test' dataset. Some details such as log-exponential transformation are also included in this formula. There is no necessity to elaborately interpret the results of such a model, since it just reflects an approximate functional relationship between date and units sold on that day.

$$
\begin{aligned}
&For\ Each\ ItemID\ In\ the\ Train\ Dataset:\\
&Log(Units) = PPR(Time)\\
&Baseline = Prediction(PPR(Time)) = Log(\hat{Units})\\
&where\ Time = days(Date -' 2012-01-01')\\
&Residuals = Log(Units) - Baseline\\
&Estimate\ Residuals\ with\ linear\ model:\\
&Residuals = LR(TimeFeatures)\\
&TimeFeatures\ include\ Weekday, Weekend, Holiday, and\ interaction\ items\\
&Prediction = exp(Baseline + \hat{Residuals}) - 1\\
&\\
&For\ The\ Same\ ItemID\ In\ the\ Test\ Dataset:\\
&With\ parameters\ derived\ from\ Train\ dataset:\\
&Prediction\ for\ Units\ is\ calculated\ as\\
&Prediction = exp(Baseline + \hat{Residuals}) - 1\\
&where\ both\ Baseline\ and\ \hat{Residuals}\ can\ be\ calculated\ with\ TimeFeatures\ in\ Test
\end{aligned}
$$

There are some small changes we need after running such a program. Some of the prediction values might be negative, which is not reasonable, and we change them into 0. It just shows that my (since I am in charge of the model) absence of work on data-cleaning and inaccuracy of this algorithm, to some extent.

The main advantages of this method are simple and computationally-cheap. No further process is needed with an algorithms in 10 lines. (It could be less if you are familiar with data.frame in R) Also, there is only one real variable in this model, and other variables are derived from it. You can do all the manipulations within one single table. You do not have to wait for the results and it just takes several milliseconds for R to do it. Most importantly, this method is quite universal, as the PPR model is a universal estimator. It can give you a not-so-bad prediction instantly, just like Maruchan noodles, fast, convenient though not that delicious and nutritious, but not too bad.

There are definitely some improvements for us in the future if we plan to continue working on this pot of noodles, I mean the models. Extreme observations should be excluded or at least winsorized. Weather variables should also be included since these are weather-sensitive items. Linear approximation for residuals can be improved with insights from diagnosis. But due to the limited time spent on this method, we have to stop at this stage, alas.

Score Report:

Submission and Description	Private Score	Public Score
testupload111.csv a few seconds ago by YueJosephWu add submission details	0.17955	0.17738

Extra Model 2: K-nearest neighbor (KNN)

KNN algorithm is a very broadly used non-parametric method for classification and regression. First of all, it calculate the Manhattan distance or Euclidean distance from every test data points to every train data points, and then find out K nearest neighbors for each test data. Finally, it make predictions for each test data by taking the majority vote from their respective neighbors. Unlike the boosted tree algorithm, each observation in a recognized neighbor have equal weights of voting. KNN is a simple but efficient approach to try always.

In details, only the storeID, ProductID and date information are used in KNN approach. I found that, although there are about 111*45 storeIDproductID combination in the training set, only 255 of them have nonzero sale data in the training set. So, I just focused on these 255 storeIDproductID, and always predict 0 for the remaining.

For these 255 storeIDproductID, I extracted the training sales data for each of them, thus there are 255 separate small data sets, and apply the knn method with date as the only feature on each of these extracted data set to predict the sales for the corresponding storeIDproductID combination in the test set.

	date	store_nbr	item_nbr	units	Difference
188348	2012-02-08	25	93	1	-38
207884	2012-02-12	25	93	1	-34
251840	2012-02-21	25	93	2	-25
276260	2012-02-26	25	93	3	-20
383708	2012-03-19	25	93	1	2
...	...	...	...	...	...
1903964	2013-01-20	25	93	5	309
1918949	2013-01-23	25	93	2	312
1928939	2013-01-25	25	93	5	314
1948919	2013-01-29	25	93	2	318
1973894	2013-02-03	25	93	1	323

145 rows × 5 columns

Fig, A sample training data of storeIDproductID=(25,93)

In order to use data information as the only feature to calculate the distance from the test data to train data. The date in both test and training data was first transformed to numeric by assigning an arbitrary starting date. In this case, “2012-3-17” are chosen as a starting date, and the number of days that are between any date from training data and this starting date are calculated. This date difference found will be the new feature used in KNN method.

	date	store_nbr	item_nbr	units	Difference
0	2012-01-01	1	1	0	-76
1	2012-01-01	1	2	0	-76
2	2012-01-01	1	3	0	-76
3	2012-01-01	1	4	0	-76
4	2012-01-01	1	5	0	-76

Fig A sample training data of the difference between two dates

The scikit-learn library are used. It is a free software machine learning library for the Python programming language. It contains KNN classifier to implement the k-nearest neighbors vote with the number of neighbors to use as a parameter. In this case, I have experimented with several different K value to find out the optimal K. Since a too large K would increase the bias but decrease the variance, while a small K might otherwise overfit the training data and make predictions on test data worse. Thus, it is essential to adjust the K to fit both training and test data well.

What we do here is running different cases of K to help select the optimal model parameters and demonstrate the performance of KNN algorithm as the parameter K is varied.

Below is a table of model's performance when K is varied.

Table 1. RMSLE as a function of different K

K parameter in KNN	Root Mean Squared Logarithmic Error (RMSLE)
1	0.23809
3	0.21179
7	0.20473
10	0.19790
14	0.19551
21	0.19066
24	0.18807

From the above table, it is evident that when k is 24, the KNN model gets the best performance for a score as 0.18807. However, since the least number of observations in the grouped training dataset is 24. So, the maximum possible K is 24.

Appendix

R code:

```
#r package
library(tidyverse)
library(MASS)
library(lmtest)
library(car)
library(robustbase)
library(dplyr)
#import data
train <- read_csv("/Users/jiangyahan/Desktop/project/train.csv")
key <- read_csv("/Users/jiangyahan/Desktop/project/key.csv")
weather <- read_csv("/Users/jiangyahan/Desktop/project/weather.csv")
test <- read_csv("/Users/jiangyahan/Desktop/project/test.csv")
train$itemid <- train$store_nbr*1000+train$item_nbr
test$itemid <- test$store_nbr*1000+test$item_nbr
kandt <- merge(train,key, by="store_nbr")
ktw <- merge(kandt,weather, by=c("station_nbr","date"))


ktw$tavg <- as.numeric(ktw$tavg)
ktw$depart <- as.numeric(ktw$depart)
ktw$dewpoint <- as.numeric(ktw$dewpoint)
ktw$wetbulb <- as.numeric(ktw$wetbulb)
ktw$heat <- as.numeric(ktw$heat)
ktw$cool <- as.numeric(ktw$cool)
ktw$stnpressure <- as.numeric(ktw$stnpressure)
ktw$sealevel <- as.numeric(ktw$sealevel)
ktw$resultspeed <- as.numeric(ktw$resultspeed)
ktw$resultdir <- as.numeric(ktw$resultdir)
ktw$avgspeed <- as.numeric(ktw$avgspeed)

ktw$preciptotal[ktw$preciptotal=='T'] <- 0
ktw$preciptotal <- as.numeric(ktw$preciptotal)
ktw$snowfall[which(ktw$snowfall=='T')] <- 0
ktw$snowfall <- as.numeric(ktw$snowfall)
```

```
ktw$preclevel <- ktw$preciptotal
ktw$preclevel[which(ktw$preclevel>0.2)] <- 1
ktw$preclevel[which(ktw$preclevel<=0.2)] <- 0

ktw$departlv <- ktw$depart

ktw$departlv[which(abs(ktw$departlv)<=8)] <- 0
ktw$departlv[which(abs(ktw$departlv)>8)] <- 1

ktw1 <- ktw[which(ktw$units!=0),]

ktw1$week <- weekdays(ktw1$date)
ktw1$week[which(ktw1$week=="Sunday"|ktw1$week=="Saturday")] <- 1
ktw1$week[which(ktw1$week == "Monday"|ktw1$week== "Tuesday"
|ktw1$week=="Wednesday" |ktw1$week=="Thursday"|ktw1$week=="Friday")] <-0
ktw1$date2 <- ktw1$date
ktw1$date2 <- as.numeric(ktw1$date2)

ktw1$date2[which(ktw1$date=='2012-01-01')] <- 1
ktw1$date2[which(ktw1$date=='2012-01-02')] <- 1
ktw1$date2[which(ktw1$date=='2012-01-16')] <- 1
ktw1$date2[which(ktw1$date=='2012-02-20')] <- 1
ktw1$date2[which(ktw1$date=='2012-03-28')] <- 1
ktw1$date2[which(ktw1$date=='2012-05-28')] <- 1
ktw1$date2[which(ktw1$date=='2012-07-04')] <- 1
ktw1$date2[which(ktw1$date=='2012-09-03')] <- 1
ktw1$date2[which(ktw1$date=='2012-10-08')] <- 1

ktw1$date2[which(ktw1$date=='2012-10-29')] <- 1
ktw1$date2[which(ktw1$date=='2012-10-30')] <- 1
ktw1$date2[which(ktw1$date=='2012-10-31')] <- 1

ktw1$date2[which(ktw1$date=='2012-11-11')] <- 1
ktw1$date2[which(ktw1$date=='2012-11-12')] <- 1
ktw1$date2[which(ktw1$date=='2012-11-19')] <- 1
ktw1$date2[which(ktw1$date=='2012-11-20')] <- 1
ktw1$date2[which(ktw1$date=='2012-11-21')] <- 1
ktw1$date2[which(ktw1$date=='2012-11-22')] <- 1
ktw1$date2[which(ktw1$date=='2012-11-23')] <- 1
```

```
ktw1$date2[which(ktw1$date=='2012-11-24')] <- 1
ktw1$date2[which(ktw1$date=='2012-11-25')] <- 1

ktw1$date2[which(ktw1$date=='2012-12-23')] <- 1
ktw1$date2[which(ktw1$date=='2012-12-24')] <- 1
ktw1$date2[which(ktw1$date=='2012-12-25')] <- 1
ktw1$date2[which(ktw1$date=='2012-12-26')] <- 1
ktw1$date2[which(ktw1$date=='2012-12-27')] <- 1

ktw1$date2[which(ktw1$date=='2013-01-01')] <- 1
ktw1$date2[which(ktw1$date=='2013-01-21')] <- 1
ktw1$date2[which(ktw1$date=='2013-02-18')] <- 1
ktw1$date2[which(ktw1$date=='2013-05-27')] <- 1
ktw1$date2[which(ktw1$date=='2013-07-04')] <- 1
ktw1$date2[which(ktw1$date=='2013-09-02')] <- 1
ktw1$date2[which(ktw1$date=='2013-10-14')] <- 1

ktw1$date2[which(ktw1$date=='2013-10-29')] <- 1
ktw1$date2[which(ktw1$date=='2013-10-30')] <- 1
ktw1$date2[which(ktw1$date=='2013-10-31')] <- 1

ktw1$date2[which(ktw1$date=='2013-11-11')] <- 1
ktw1$date2[which(ktw1$date=='2013-11-25')] <- 1
ktw1$date2[which(ktw1$date=='2013-11-26')] <- 1
ktw1$date2[which(ktw1$date=='2013-11-27')] <- 1
ktw1$date2[which(ktw1$date=='2013-11-28')] <- 1
ktw1$date2[which(ktw1$date=='2013-11-29')] <- 1
ktw1$date2[which(ktw1$date=='2013-11-30')] <- 1
ktw1$date2[which(ktw1$date=='2013-12-01')] <- 1

ktw1$date2[which(ktw1$date=='2013-12-23')] <- 1
ktw1$date2[which(ktw1$date=='2013-12-24')] <- 1
ktw1$date2[which(ktw1$date=='2013-12-25')] <- 1
ktw1$date2[which(ktw1$date=='2013-12-26')] <- 1
ktw1$date2[which(ktw1$date=='2013-12-27')] <- 1

ktw1$date2[which(ktw1$date=='2014-01-01')] <- 1
ktw1$date2[which(ktw1$date=='2014-01-20')] <- 1
ktw1$date2[which(ktw1$date=='2014-02-17')] <- 1
```

```
ktw1$date2[which(ktw1$date=='2014-05-26')] <- 1
ktw1$date2[which(ktw1$date=='2014-07-04')] <- 1
ktw1$date2[which(ktw1$date=='2014-09-01')] <- 1
ktw1$date2[which(ktw1$date=='2014-10-13')] <- 1

ktw1$date2[which(ktw1$date=='2014-10-29')] <- 1
ktw1$date2[which(ktw1$date=='2014-10-30')] <- 1
ktw1$date2[which(ktw1$date=='2014-10-31')] <- 1

ktw1$date2[which(ktw1$date=='2014-11-11')] <- 1
ktw1$date2[which(ktw1$date=='2014-11-24')] <- 1
ktw1$date2[which(ktw1$date=='2014-11-25')] <- 1
ktw1$date2[which(ktw1$date=='2014-11-26')] <- 1
ktw1$date2[which(ktw1$date=='2014-11-27')] <- 1
ktw1$date2[which(ktw1$date=='2014-11-28')] <- 1
ktw1$date2[which(ktw1$date=='2014-11-29')] <- 1
ktw1$date2[which(ktw1$date=='2014-11-30')] <- 1

ktw1$date2[which(ktw1$date=='2014-12-23')] <- 1
ktw1$date2[which(ktw1$date=='2014-12-24')] <- 1
ktw1$date2[which(ktw1$date=='2014-12-25')] <- 1
ktw1$date2[which(ktw1$date=='2014-12-26')] <- 1
ktw1$date2[which(ktw1$date=='2014-12-27')] <- 1

ktw1$date2[which(ktw1$date2!=1)] <- 0
#first linear model
full_model <- lm(units ~  factor(itemid) + tavg + preclevel + date2 + week, data = ktw1)
summary(full_model)
### diag ###
par(mfrow = c(2,2))
plot(full_model)
#Check for high leverage points, outliers, and influential points
p = 4804
n = nrow(ktw1)
lev = hatvalues(full_model)
lev[(lev > 2*p/n)]

#outliers
student.res = rstandard(full_model)
```

```
student.res[abs(student.res) > 3]

#influential points
cooksd = cooks.distance(full_model)
cooksd[cooksd>0.5]
#vif
vif(full_model)
#bptest
bptest(full_model)
#improvement

library(robust)
avgunits <- aggregate(ktw1$units, list(ktw1$itemid),mean)
colnames(avgunits) <- c("itemid","avgunits")
ktw2 <- merge(ktw1,avgunits,by = "itemid")
ktwo <- ktw2[which(ktw2$units!=5568),]
ktwo <- ktwo[which(ktwo$units!=3369),]

kt <- dplyr::select(ktwo,units,avgunits,tavg,preclevel,date2,week)
fitm <- lmrob(units ~ avgunits  + tavg +  preclevel + date2 + week, data = kt)
summary(fitm)
fitmm <- rlm(units ~ avgunits  + tavg +  preclevel + date2 + week, data = kt)
fit.compare = fit.models(list(Robust = "rlm",
                              "LS" = "lm"), formula = units ~ avgunits  + tavg +  preclevel + date2 + week,
data = kt)
par(mfrow = c(2,2))
plot(fit.compare)
##improvement##
improve <- lm(log(units) ~ week*date2 + preclevel + tavg + factor(itemid)+week+date2, data =
ktwo)
par(mfrow=c(2,2));plot(improve)
summary(improve)
bptest(improve)
boxcox(improve)
max(vif(improve))
aic_backward = step(improve, trace = 0)
n = nrow(kt)
bic_backward = step(improve, k = log(n), trace = 0)
```

```
model_intercept = lm(log(units) ~ 1, data=ktwo)
aic_forward = step(model_intercept, scope = list(upper = full_model), direction =
"forward",trace = 0)
bic_forward = step(model_intercept, scope = list(upper = full_model), k = log(n), direction =
"forward",trace = 0)
aic_forward
cv <- function(model) {
  SSE.predict <- numeric(10)
  folds <- rep_len(1:10, nrow(kt))

  for (k in 1:10) {
    test.index <- which(folds == k)
    data.train <- kt[-test.index, ]
    data.test <- kt[test.index, ]
    lm.temp <- lm(model, data = data.train)
    SSE.predict[k] <-
      crossprod(predict(lm.temp, data.test) - log(data.test$units))
  }
  return(sum(SSE.predict))
}
cv(aic_backward)
cv(bic_backward)
cv(aic_forward)
cv(bic_forward)
improve2 <- lm(log(units) ~ week + date2 + preclevel + tavg + factor(itemid), data = ktwo)
windows();par(mfrow=c(2,2));plot(improve2)
summary(improve2)
bptest(improve2)
boxcox(improve2)
max(vif(improve2))
attach(ktwo)
sdv <- tapply(units,list(itemid),sd)
idnum <- data.frame(table(ktwo$itemid))
sdvdata <- data.frame(idnum,data.frame(sdv))
sdvdata$wt <- sdvdata$sdv/sqrt(sdvdata$Freq)
colnames(sdvdata)=c("itemid","n","sd","wt")
ktwo2 <- merge(ktwo,sdvdata, by="itemid")
improve3 <- lm(log(units) ~ week*date2 + preclevel + tavg + factor(itemid), weights = wt , data
= ktwo2)
```

```
summary(improve3)
bptest(improve3)
pos <- glm(units ~ week*date2 + factor(itemid) + preclevel + tavg,family = poisson, data =
ktwo)
summary(pos)
max(vif(pos))
bptest(pos)
windows();par(mfrow=c(2,2));plot(pos)
cv1 <- function(model) {
  SSE.predict <- numeric(10)
  folds <- rep_len(1:10, nrow(kt))

  for (k in 1:10) {
    test.index <- which(folds == k)
    data.train <- kt[-test.index, ]
    data.test <- kt[test.index, ]
    lm.temp <- glm(model, family = poisson,data = data.train)
    SSE.predict[k] <-
      crossprod(predict(lm.temp, data.test) - (data.test$units))
  }
  return(sum(SSE.predict))
}
cv1(pos)
cv(improve)
cv(improve2)
cv(improve3)
#test
test <- read_csv("/Users/jiangyahan/Desktop/project/test.csv")
test$itemid <- test$store_nbr*1000+test$item_nbr
train <- read_csv("/Users/jiangyahan/Desktop/project/train.csv")
train$itemid <- train$store_nbr*1000+train$item_nbr
samplesub <- read_csv("/Users/jiangyahan/Desktop/project/sampleSubmission(1).csv")
train1 <- train[which(train$units!=0),]
avgunits <- aggregate(test$units,list(test$itemid),mean)
colnames(avgunits) <- c("itemid","avgunits")
test <- merge(avgunits,test, by = "itemid")
ID <- data.frame(table(train1$itemid))
colnames(ID) <- c('itemid','freq')
```

```
test$week <- weekdays(test$date)
test$week[which(test$week=="Sunday" | test$week=="Saturday")] <- 1
test$week[which(test$week == "Monday"|test$week== "Tuesday" |test$week=="Wednesday"
|test$week=="Thursday"|test$week=="Friday")] <-0



test$date2 <- test$date
test$date2 <- as.numeric(test$date2)


test$date3 <- test$date
test$date3 <- day(test$date3)



test$date2[which(test$date=='2012-01-01')] <- 1
test$date2[which(test$date=='2012-01-02')] <- 1
test$date2[which(test$date=='2012-01-16')] <- 1
test$date2[which(test$date=='2012-02-20')] <- 1
test$date2[which(test$date=='2012-03-28')] <- 1
test$date2[which(test$date=='2012-05-28')] <- 1
test$date2[which(test$date=='2012-07-04')] <- 1
test$date2[which(test$date=='2012-09-03')] <- 1
test$date2[which(test$date=='2012-10-08')] <- 1

test$date2[which(test$date=='2012-10-29')] <- 1
test$date2[which(test$date=='2012-10-30')] <- 1
test$date2[which(test$date=='2012-10-31')] <- 1
```

```
test$date2[which(test$date=='2012-11-11')] <- 1
test$date2[which(test$date=='2012-11-12')] <- 1
test$date2[which(test$date=='2012-11-19')] <- 1
test$date2[which(test$date=='2012-11-20')] <- 1
test$date2[which(test$date=='2012-11-21')] <- 1
test$date2[which(test$date=='2012-11-22')] <- 1
test$date2[which(test$date=='2012-11-23')] <- 1
test$date2[which(test$date=='2012-11-24')] <- 1
test$date2[which(test$date=='2012-11-25')] <- 1

test$date2[which(test$date=='2012-12-23')] <- 1
test$date2[which(test$date=='2012-12-24')] <- 1
test$date2[which(test$date=='2012-12-25')] <- 1
test$date2[which(test$date=='2012-12-26')] <- 1
test$date2[which(test$date=='2012-12-27')] <- 1

test$date2[which(test$date=='2013-01-01')] <- 1
test$date2[which(test$date=='2013-01-21')] <- 1
test$date2[which(test$date=='2013-02-18')] <- 1
test$date2[which(test$date=='2013-05-27')] <- 1
test$date2[which(test$date=='2013-07-04')] <- 1
test$date2[which(test$date=='2013-09-02')] <- 1
test$date2[which(test$date=='2013-10-14')] <- 1

test$date2[which(test$date=='2013-10-29')] <- 1
test$date2[which(test$date=='2013-10-30')] <- 1
test$date2[which(test$date=='2013-10-31')] <- 1

test$date2[which(test$date=='2013-11-11')] <- 1
test$date2[which(test$date=='2013-11-25')] <- 1
test$date2[which(test$date=='2013-11-26')] <- 1
test$date2[which(test$date=='2013-11-27')] <- 1
test$date2[which(test$date=='2013-11-28')] <- 1
test$date2[which(test$date=='2013-11-29')] <- 1
test$date2[which(test$date=='2013-11-30')] <- 1
test$date2[which(test$date=='2013-12-01')] <- 1

test$date2[which(test$date=='2013-12-23')] <- 1
```

```
test$date2[which(test$date=='2013-12-24')] <- 1
test$date2[which(test$date=='2013-12-25')] <- 1
test$date2[which(test$date=='2013-12-26')] <- 1
test$date2[which(test$date=='2013-12-27')] <- 1

test$date2[which(test$date=='2014-01-01')] <- 1
test$date2[which(test$date=='2014-01-20')] <- 1
test$date2[which(test$date=='2014-02-17')] <- 1
test$date2[which(test$date=='2014-05-26')] <- 1
test$date2[which(test$date=='2014-07-04')] <- 1
test$date2[which(test$date=='2014-09-01')] <- 1
test$date2[which(test$date=='2014-10-13')] <- 1

test$date2[which(test$date=='2014-10-29')] <- 1
test$date2[which(test$date=='2014-10-30')] <- 1
test$date2[which(test$date=='2014-10-31')] <- 1

test$date2[which(test$date=='2014-11-11')] <- 1
test$date2[which(test$date=='2014-11-24')] <- 1
test$date2[which(test$date=='2014-11-25')] <- 1
test$date2[which(test$date=='2014-11-26')] <- 1
test$date2[which(test$date=='2014-11-27')] <- 1
test$date2[which(test$date=='2014-11-28')] <- 1
test$date2[which(test$date=='2014-11-29')] <- 1
test$date2[which(test$date=='2014-11-30')] <- 1

test$date2[which(test$date=='2014-12-23')] <- 1
test$date2[which(test$date=='2014-12-24')] <- 1
test$date2[which(test$date=='2014-12-25')] <- 1
test$date2[which(test$date=='2014-12-26')] <- 1
test$date2[which(test$date=='2014-12-27')] <- 1



test$date2[which(test$date2!=1)] <- 0
```

```
test1 <- merge(test,key,by="store_nbr")
testmatch1 <- merge(test1,weather,by=c("station_nbr","date"))
testmatch <- merge(testmatch1,ID,by='itemid')
length(table(testmatch$itemid))

testmatch$preciptotal[testmatch$preciptotal=='T'] <- 0
testmatch$preciptotal <- as.numeric(testmatch$preciptotal)
testmatch$preclevel <- testmatch$preciptotal
testmatch$preclevel[which(testmatch$preclevel>0.2)] <- 1
testmatch$preclevel[which(testmatch$preclevel<=0.2)] <- 0




testmatch$depart <- as.numeric(testmatch$depart)
testmatch$departlv <- testmatch$depart

testmatch$departlv[which(abs(testmatch$departlv)<=8)] <- 0
testmatch$departlv[which(abs(testmatch$departlv)>8)] <- 1
testmatch$month <- month(testmatch$date)
testmatch2 <- merge(testmatch,avgunits,by="itemid")
testmatch2$tavg <- as.numeric(testmatch2$tavg)

predtstfull <- predict(full_model,testmatch2)

predtst1 <- predict(improve,testmatch2)
predtst1 <- exp(predtst1)
predtst2 <- predict(improve2,testmatch2)
predtst2 <- exp(predtst2)
predtst3 <- predict(pos,testmatch2)
predtst3 <- exp(predtst3)

testfull <- data.frame(testmatch2,predtstfull)
testuploadf <- merge(test,testfull, by=c('itemid','date'), all=TRUE)

testim1 <- data.frame(testmatch2,predtst1)
testupload1 <- merge(test,testim1, by=c('itemid','date'), all=TRUE)

testim2 <- data.frame(testmatch2,predtst2)
```

```
testupload2 <- merge(test,testim2, by=c('itemid','date'), all=TRUE)

testim3 <- data.frame(testmatch2,predtst3)
testupload3 <- merge(test,testim3, by=c('itemid','date'), all=TRUE)

testuploadf$predtstfull[which(is.na(testuploadf$predtstfull))] <-0
testupload1$predtst1[which(is.na(testupload1$predtst1))] <-0
testupload2$predtst2[which(is.na(testupload2$predtst2))] <-0
testupload3$predtst3[which(is.na(testupload3$predtst3))] <-0

testuploadf <- arrange(testuploadf,date,itemid)
testupload1 <- arrange(testupload1,date,itemid)
testupload2 <- arrange(testupload2,date,itemid)
testupload3 <- arrange(testupload3,date,itemid)
###We change the right part of the code below everytime we write a new csv.
samplesub$units <- testupload1$predtst1


write.csv(samplesub,"/Users/jiangyahan/Desktop/project/imp2.csv")
```

Code for Extra Model 1 (Both R and STATA):

STATA first (mainly about data pre-processing):

```
//Notice: this 'train' dataset has delete all 0-unit obs
insheet using train.csv, clear //Import the csv file
gen uniid=1000*store_nbr+item_nbr //Generate uniid, same as itemid in previous code
gen logunits=log(units) //Natural log of units, var in regression
gen grpnum=0 //Initialize group number for uniid, 255 in total
sort uniid date //Sorting before write-in
egen grpnum=group(uniid)//Write-in
//Now we come to date
gen numdate=date(date,"YMD") //Numerical date, beginning someday on 1960 or 1970, cannot remember exactly
gen defnumdate=numdate-18993 //Then I want it start on 20120101
gen dow=dow(numdate) //Which day of the week
gen weekend=0 //Whether it is weekend
replace weekend=1 if dow==0 | dow==6
gen holiday=0 //Whether it National Holiday
//With a long long list
replace holiday = 1 if date == "2012/1/2" | date== "2012/1/16" | date== "2012/2/20"
replace holiday = 1 if date == "2012/5/28" | date== "2012/7/4" | date== "2012/9/3"
replace holiday = 1 if date == "2012/10/8" | date== "2012/11/12" | date== "2012/11/22"
replace holiday = 1 if date == "2012/12/25"

replace holiday = 1 if date == "2013/1/1" | date== "2013/1/21" | date== "2013/2/18"
replace holiday = 1 if date == "2013/5/27" | date== "2013/7/4" | date== "2013/9/2"
replace holiday = 1 if date == "2013/10/14" | date== "2013/11/11" | date== "2013/11/28"
replace holiday = 1 if date == "2013-12-25"

replace holiday = 1 if date == "2014/1/1" | date== "2014/1/20" | date== "2014/2/17"
replace holiday = 1 if date == "2014/5/26" | date== "2014/7/4" | date== "2014/9/1"
replace holiday = 1 if date == "2014/10/13" | date== "2014/11/11" | date== "2014/11/27"
replace holiday = 1 if date == "2014/12/25"
//Whether it is around Black Friday
//U.S. Predecessors will be angry to death if they know their offsprings are so consumeristic
gen arnd_bf=0

replace arnd_bf=1 if date=="2012/11/19"| date== "2012/11/20" | date=="2012/11/21"
replace arnd_bf=1 if date=="2012/11/22"| date== "2012/11/23" | date=="2012/11/24"
```

```
replace arnd_bf=1 if date=="2012/11/25"

replace arnd_bf=1 if date=="2013/11/25"| date== "2013/11/26" | date=="2013/11/27"
replace arnd_bf=1 if date=="2013/11/28"| date== "2013/11/29" | date=="2013/11/30"
replace arnd_bf=1 if date=="2013/12/1"

replace arnd_bf=1 if date=="2014/11/24"| date== "2014/11/25" | date=="2014/11/26"
replace arnd_bf=1 if date=="2014/11/27"| date== "2014/11/28" | date=="2014/11/29"
replace arnd_bf=1 if date=="2014/11/30"
//Which year, month, and day of month it belongs to
gen year=year(numdate)
gen month=month(numdate)
gen day=day(numdate)

//This is for the train dataset, however it is the same for test dataset, the only difference is that no units in that case
//Therefore I would not repeat such steps again.
```

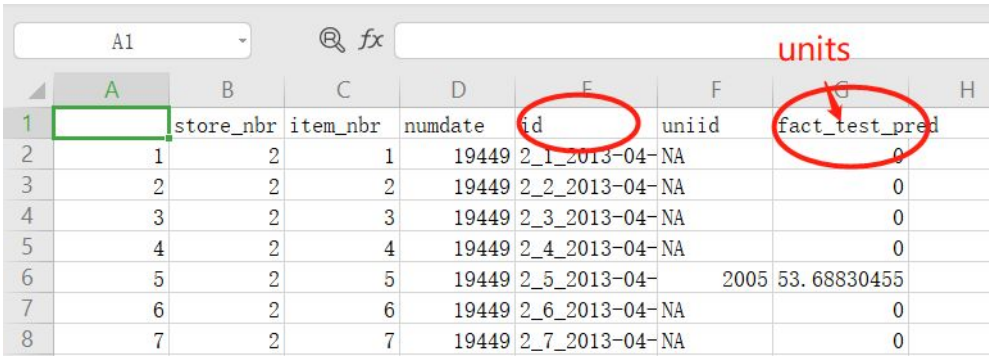
R code (mainly about modelling):

```
setwd("D:/RStudio")
library(dplyr)
##Import necessary library
ppr.train<-read.csv("ppr_data.csv",header = T)
ppr.test<-read.csv("ppr_test.csv",header = T)
##Import pre-processed data
predictmat<-c(0)
resmat<-c(0)
pprtrainpred<-c(0)
##You cannot write data in a null-matrix
##So create a matrix with one element:0
rng <- 1:255
##rng is grpnum in fact, from 1 to 255
for(k in rng){
  ##for each grpnum
  subppr.train<-ppr.train[which(ppr.train$grpnum==k),]
  ppr.trainmodel<-ppr(logunits~defnumdate,data=subppr.train,nterms=3,max.terms=5)
  ##train a ppr model with parameters above
  res.ppr<-subppr.train$logunits-predict(ppr.trainmodel,subppr.train)
  ##store the residual, which will be used in next regression
```

```
  ppr.trainpred<-predict(ppr.trainmodel,subppr.train)
  ##store the baseline for train model

  subppr.test<-ppr.test[which(ppr.test$grpnum==k),]
  ##subset for test dataset
  ppr.testpred<-predict(ppr.trainmodel,subppr.test)
  ##predict with trained ppr model for baseline
  pprtrainpred<-c(pprtrainpred,ppr.trainpred)
  ##just append the data to the pre-set 1 element matrix, 3 of them
  predictmat<-c(predictmat,ppr.testpred)
  resmat<-c(resmat,res.ppr)
}
##delete the first element, which is a pre-set 0
pprtrainpred<-pprtrainpred[-1]
predictmat<-predictmat[-1]
resmat<-resmat[-1]

##build the residual linear regression model
res.lm<-lm(resmat~dow+weekend+holiday+arnd_bf+year+month+day+store_nbr+item_nbr+sto
re_nbr:arnd_bf+item_nbr:arnd_bf,data=ppr.train)
##prediction of residuals with linear model
res.pred<-predict(res.lm,ppr.test)
##baseline+residuals=prediction
test_blend_pred<-predictmat+res.pred
##back to the original form from log-transformation
fact_test_pred<-exp(test_blend_pred)-1

##Some administrative issue, such as writing data, export file, etc
output<-data.frame(ppr.test$numdate,ppr.test$uniid,ppr.test$store_nbr,ppr.test$item_nbr,fact_tes
t_pred)
head(output)
colnames(output)<-c("numdate","uniid","store_nbr","item_nbr","fact_test_pred")

testsample<-read.csv("testsample.csv",header = T)
testupload<-merge(testsample,output,by=c("store_nbr","item_nbr","numdate"),all=TRUE)
##those uniid do not match, thus generating missing values, just fit as 0
testupload$fact_test_pred[which(is.na(testupload$fact_test_pred))] <-0
##turn negative into 0, slight cleaning
testupload$fact_test_pred[which(testupload$fact_test_pred<0)]<-0
```

testupload$fact_test_pred[which(testupload$numdate==19717)]<-0
##sort it into the order we are going to submit
testupload<-arrange(testupload,numdate,store_nbr,item_nbr)
write.csv(testupload,"D:/RStudio/testupload111.csv")

Notice that the original output file should look like this, and we just manually delete several columns within Excel (keep the circled two columns), and rename a column. It definitely can be done within R, however, Excel is also very convenient.



Code for Extra Model 2 (Python):

In [1]:
```
%reset -f
import os
import pandas as pd
import datetime
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
# We do this to ignore several specific Pandas warnings
import warnings
warnings.filterwarnings("ignore")
os.chdir('C:\\Users\\cd7\\Desktop\\stat425\\final\\train.csv')
```

In [2]:
```
dftrain = pd.read_csv('train.csv')
# dftrain.head()

dftrain['Difference'] = (pd.to_datetime(dftrain['date']) - pd.to_datetime('2012-03-17')).dt.days
# dftrain.head()

os.chdir('C:\\Users\\cd7\\Desktop\\stat425\\final')
dftest = pd.read_csv('test.csv')
# dftest.head()

dftest['Difference'] = (pd.to_datetime(dftest['date']) - pd.to_datetime('2012-03-17')).dt.days
# dftest.head()

clean1 = dftrain[dftrain.units!= 0]

count_series = clean1.groupby(['store_nbr', 'item_nbr']).size()

# clean1.groupby(['store_nbr', 'item_nbr']).count()
```

In [3]:
```
data1 = clean1.groupby(['store_nbr', 'item_nbr']).get_group((1,9))

gb = clean1.groupby(['store_nbr', 'item_nbr'])
dflist = [gb.get_group(x) for x in gb.groups]

data2 = dftest.groupby(['store_nbr', 'item_nbr']).get_group((2,5))

joinkey = list(gb.groups.keys())

gb2 = dftest.groupby(['store_nbr', 'item_nbr'])
joinkey2 = list(gb2.groups.keys())

join = list(set(joinkey) & set(joinkey2))

test1 = dftest.groupby(['store_nbr', 'item_nbr']).get_group(joinkey[0])

testdf = [dftest.groupby(['store_nbr', 'item_nbr']).get_group(x) for x in join]

traindf = [gb.get_group(x) for x in join]
```

In [4]:
```
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=1)
# neigh = KNeighborsClassifier(n_neighbors=10)
# neigh = KNeighborsClassifier(n_neighbors=21)
# neigh = KNeighborsClassifier(n_neighbors=24)
```

In [5]:
```
for i in range(len(join)):
    neigh.fit(traindf[i]['Difference'].values.reshape(-1, 1), traindf[i]['units'])  # fit for store_nbr=1, item_nbr=9
    x = neigh.predict(testdf[i]['Difference'].values.reshape(-1, 1)) # prediction for store_nbr=1, item_nbr=9
    testdf[i]['Predict'] = x
    testdf[i] = testdf[i]['Predict']
```

Reassign the prediction results into a excel sheet for final submission of a score

In [6]:
```
new_testdf = pd.concat(testdf)
new_testdf = new_testdf.sort_index()

new_testdf2 = new_testdf.reindex(range(0, len(dftest)), fill_value=0)

new_testdf2.to_csv('Submission.csv',index=False,header=['units'])

dftrain.shape
```

Reference:

The Impact of Weather on Retail Sector in the UK. (n.d.). Retrieved from http://www.weatherads.io/blog/2015/august/the-impact-of-weather-on-retail-sector-in-the-uk.

Robust Regression . (n.d.). Retrieved December 15, 2019, from https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Robust_Regression.pdf

PPR in R:
https://www.rdocumentation.org/packages/stats/versions/3.6.1/topics/ppr

KNN in Wikipedia:
https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html