Stat 425 Final Project

By Mark Belsis and Alex Pope Funches

mbelsis2 and funches2

(Control + Left Click) to Jump to a Section

Table of Contents

Introduction

The goal of this project is to accurately analyze and predict the sale of weather sensitive products sold by Walmart with minimal error. Walmart is based in 27 countries and operates 11,450 stores. The datasets we will be working with were collected by Walmart from 45 different store locations and 20 corresponding weather stations. The merged train-weather dataset is the primary dataset we will be working with and is comprised of 4,617,600 observations with 23 variables. Within the observations are the 111 products sold in 45 Walmart locations, the store number and item number are used as references and specific to each store due to the geographical and supplier differences. Each weather station has its own unique id like each store and item and the stations record weather characteristics like the amount of snow/precipitation, average temperature, wind speed, sea level pressure, and significant events like thunderstorms, sandstorms, and rain on any given day.

To best predict the sale of weather sensitive products across the changing seasons, climates, and cultures we will first attempt to create a multiple linear regression model that we will use to predict the number of units sold of a particular item at a particular store. After we have created a multiple linear regression model, depending on its performance we may look to create further improvements to the model or take an entirely different approach in order to create a reliable model that allows us to predict units sold as best we can.

Exploratory Data Analysis

Merging Data and Cleaning Zero Mean Item-Store Combinations

To merge the data, first the station_nbr variable was added to the training data via the key and the corresponding store_nbr. Once that was done the training and weather data were easily merged in one line using the merge function as shown in Figure1 below:

```
full_training_data = merge(train_data_withstation,weather, by = c("date","station_nbr"))
```

Figure1: merge function to merge the training and weather datasets

This was done before any modifications to any datasets besides adding the variable station_nbr to the training dataset. With this step complete the merge function can match the date and station_nbr variables to appropriately merge the data. For clarity moving forward, the full_training_data dataset will simply be referred to as the merged dataset.

To start off, one important idea worth considering is if any items were potentially discontinued or no longer held at any of the stores. If this is the case, then the unit sales of these items at certain stores would most definitely be 0 for any date we are going to predict or train our models on. Using the describe.by function from the psych package we can get descriptive statistics of the merged dataset by specific variables which in this case are store_nbr and item_nbr.

	item <fctr>	group1 <fctr>	group2 <fctr>	vars <dbl>	n <dbl>	mean <dbl>
X11	1	1	1	1	929	0
X12	2	2	1	1	929	0
X13	3	3	1	1	929	0
X14	4	4	1	1	929	0
X15	5	5	1	1	929	0
X16	6	6	1	1	929	0
X17	7	7	1	1	929	0
X18	8	8	1	1	929	0
X110	10	10	1	1	929	0
X111	11	11	1	1	929	0

1-10 of 4,740 rows | 1-10 of 16 columns

Figure2: describe.by results where group1 represents item_nbr while group2 represents store_nbr

As seen in Figure2 above, there are 4,740 different item-store combinations of the possible 4995 combinations that have a units mean of 0 indicating that these items were likely discontinued at these stores. If we remove these item-store combinations from the merged dataset we reduce the number of observations from 4,617,600 to 236,038 which is much more manageable. As for predictions, regardless of the model we will predict unit sales of 0 for these item-store combinations which should improve the performance of our models.

Removing Variables

Now before a scatterplot is generated, we'll take some time to consider which variables are worth keeping. Believing the time specific variables of sunrise and sunset to be insignificant predictors, both will be removed and since both contain variables contained many missing data entries this supports the removal. Under the same line of thought variables like depart, dewpoint, wetbulb, heat, and cool will also be removed as they don't seem like important predictors that can contribute to the prediction and explanation of unit sales. So far this leaves us with the cleaned merged dataset consisting of 236,038 observations and 16 variables.

Scatterplot, Correlation, and More Removed Variables

With the remaining 16 variables a scatterplot will be generated to see if there are any important characteristics to explore or be wary of.

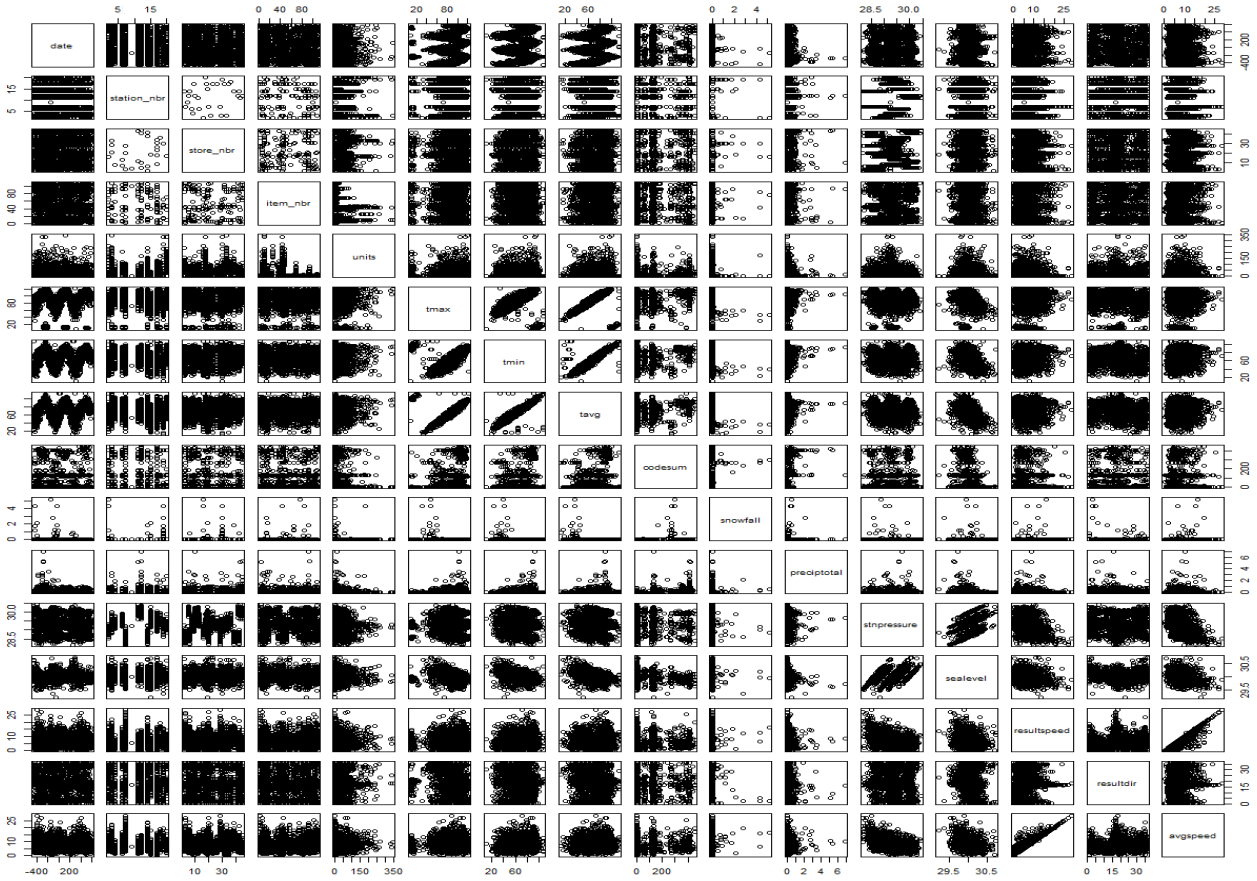


Figure3: Pairwise scatterplot of the 16 variables in consideration

As we can see from Figure3, there are quite a few variables that seem to have strong correlation. To confirm these scatterplots, correlation coefficients were generated as seen in Figure4 below. Tmax, tmin, and tavg all share correlation coefficients close to 0.5 so tmax and tmin will be removed. Avgspeed and resultspeed are also highly correlated nearing a coefficient of 1 so resultspeed will be removed because of the correlation but resultdir will also be removed because there is no discernable pattern in the pairwise plot of resultdir and units so it doesn't seem to be very important in predicting unit sales. This brings the running total of variables down to 12.

cor.resultspeed.avgspeed	0.929523802027815
cor.resultspeed.resultdir	0.0847243009785245
cor.sealevel.stnpressure	0.439555755445427
cor.tmax.tavg	0.454758874317043
cor.tmax.tmin	0.421407538620914
cor.tmin.tavg	0.924239540340003

Figure4: Correlation coefficients between various variables

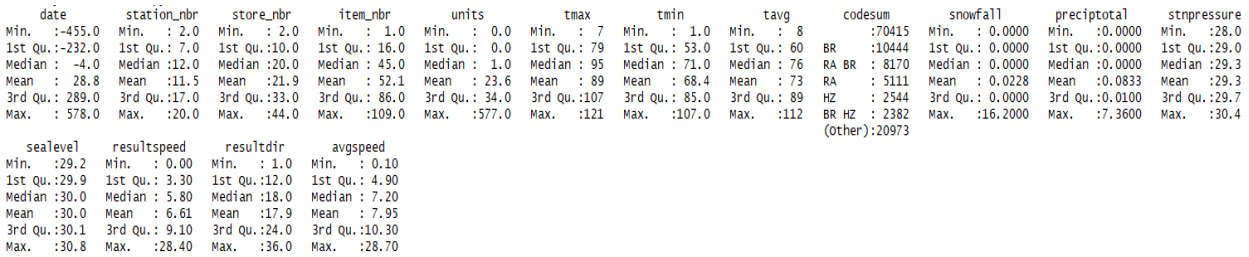
The remaining pairwise plots in Figure3 with units in mind show some interesting behavior that warrants them to be left in for model building.

Data Summaries and Removing codesum

One of the last things that will be done before moving forward with creating models is to check the structure of the merged dataset itself. Using the summary function on the merged dataset we find that Figure5 below shows quite a few issues. All the variables besides codesum should be numerical but R has interpreted many of them as factors rather than continuous variables. This will create many problems in model creation and model selection so as seen in the 'Cleaning' section of the R code (see Figure18 at end of Appendix), all the variables except codesum were converted into numeric variables and missing observations containing an "M" for Missing or "T" for Trace were removed from the dataset. Figure6 below confirms that we fixed all issues with the now numerical variables. An interesting observation in the factor corrected merged dataset in Figure6 is that the minimum of store_nbr is now 2 indicating that stores within the first weather station have been omitted.

```
     date         station_nbr     store_nbr       item_nbr         units              tmax            tmin            tavg          codesum         snowfall      preciptotal      stnpressure
Min.   :-455.0   Min.   : 1.0   Min.   : 1.0   Min.   :  1.0   Min.   :   0.0   M      :  7555   M      :  7551   M      : 10800          :131325   0.0    :134398   0.00   :141262   M      :  7566
1st Qu.:-219.0   1st Qu.: 7.0   1st Qu.:12.0   1st Qu.: 18.0   1st Qu.:   0.0   86     :  5436   73     :  5097   81     :  5136   BR     : 15755   M      : 97587    T    : 29692   29.93  :  2815
Median :  16.0   Median :12.0   Median :22.0   Median : 50.0   Median :   1.0   79     :  5396   74     :  5023   83     :  5080   RA     : 15727    T     :  2392   0.01   :  8033   29.28  :  2812
Mean   :  38.9   Mean   :11.2   Mean   :22.8   Mean   : 53.2   Mean   :  19.3   91     :  5346   72     :  4918   69     :  4933   RA BR  : 15177   0.2    :   194   M      :  6959   29.26  :  2704
3rd Qu.: 299.0   3rd Qu.:15.0   3rd Qu.:34.0   3rd Qu.: 88.0   3rd Qu.:  24.0   83     :  5216   59     :  4913   71     :  4880   BR HZ  :  3873   0.1    :   161   0.02   :  4285   29.39  :  2670
Max.   : 578.0   Max.   :20.0   Max.   :45.0   Max.   :111.0   Max.   :5568.0   81     :  5167   71     :  4719   79     :  4815   SN BR  :  3828   0.5    :   142   0.03   :  2852   30.03  :  2639
                                                                                 (Other):201922   (Other):203817   (Other):200394   (Other): 50353   (Other):  1164   (Other): 42955   (Other):214832
   sealevel       resultspeed       resultdir         avgspeed
M      : 10269   M      :  4828   17     : 16389   M      :  7109
30.00  :  5603   4.1    :  3018   18     : 15850   6.9    :  4322
29.94  :  5465   3.1    :  2921   16     : 12829   5.8    :  3899
29.96  :  5336   2.1    :  2835   19     : 11328   8.1    :  3856
29.97  :  5283   2.8    :  2782   15     :  9693   4.6    :  3227
30.06  :  5265   3.5    :  2773   20     :  8284   6.6    :  3076
(Other):198817   (Other):216881   (Other):161665   (Other):210549
```

Figure5: Summary of the merged dataset before factor corrections



```
      date          station_nbr      store_nbr       item_nbr         units            tmax            tmin            tavg          codesum          snowfall         preciptotal      stnpressure
 Min.   :-455.0   Min.   : 2.0   Min.   : 2.0   Min.   :  1.0   Min.   :  0.0   Min.   :  7   Min.   :  1.0   Min.   :  8          :70415   Min.   : 0.0000   Min.   :0.0000   Min.   :28.0
 1st Qu.:-232.0   1st Qu.: 7.0   1st Qu.:10.0   1st Qu.: 16.0   1st Qu.:  0.0   1st Qu.: 79   1st Qu.: 53.0   1st Qu.: 60   BR     :10444   1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.:29.0
 Median :  -4.0   Median :12.0   Median :20.0   Median : 45.0   Median :  1.0   Median : 95   Median : 71.0   Median : 76   RA BR  : 8170   Median : 0.0000   Median :0.0000   Median :29.3
 Mean   :  28.8   Mean   :11.5   Mean   :21.9   Mean   : 52.1   Mean   : 23.6   Mean   : 89   Mean   : 68.4   Mean   : 73   RA     : 5111   Mean   : 0.0228   Mean   :0.0833   Mean   :29.3
 3rd Qu.: 289.0   3rd Qu.:17.0   3rd Qu.:33.0   3rd Qu.: 86.0   3rd Qu.: 34.0   3rd Qu.:107   3rd Qu.: 85.0   3rd Qu.: 89   HZ     : 2544   3rd Qu.: 0.0000   3rd Qu.:0.0100   3rd Qu.:29.7
 Max.   : 578.0   Max.   :20.0   Max.   :44.0   Max.   :109.0   Max.   :577.0   Max.   :121   Max.   :107.0   Max.   :112   BR HZ  : 2382   Max.   :16.2000   Max.   :7.3600   Max.   :30.4
                                                                                                                (Other):20973

    sealevel      resultspeed      resultdir       avgspeed
 Min.   :29.2   Min.   : 0.00   Min.   : 1.0   Min.   : 0.10
 1st Qu.:29.9   1st Qu.: 3.30   1st Qu.:12.0   1st Qu.: 4.90
 Median :30.0   Median : 5.80   Median :18.0   Median : 7.20
 Mean   :30.0   Mean   : 6.61   Mean   :17.9   Mean   : 7.95
 3rd Qu.:30.1   3rd Qu.: 9.10   3rd Qu.:24.0   3rd Qu.:10.30
 Max.   :30.8   Max.   :28.40   Max.   :36.0   Max.   :28.70
```

Figure6: Summary of the merged dataset after factor corrections

The last consideration before we move into model construction is codesum. Normally a factored variable would not be an issue in creating models, however in this case the sheer number of different combinations of factors that come from codesum make it immensely difficult to build a comprehensible model from and even more difficult to construct predictions from. For this reason, codesum will be the last variable removed from the merged dataset. As for the actual data from the summaries shown in Figure6, none of the values particularly stand out as an issue. Date was converted early on into a numerical variable with respect to 3/31/2013 which is a day before the test dataset starts. This was done because it makes building models and predictions easier and the date was selected because making predictions is easier since turning the date into a numeric in the test dataset will have 4/1/2013 start at 1 which simplifies predictions. The minimums, maximums, and means of all the variables seem to be reasonable so it is unlikely there were any errors in converting the variables.

Linear Regression Model / Diagnostics

Preliminary Model

As seen in at the beginning of the Exploratory Data Analysis section, the training and weather station datasets were merged using the merge function by the date and station_nbr variables. As a remainder, after cleaning the dataset of 0 units means, removing variables we perceived as inadequate, removing variables with high correlation, and removing observations with missing data entries, the grand total of our dataset leaves us with 102,039 observations having 11 variables. In the code section (see Figure18) you will see this dataset saved as data_set_ols and it is the dataset we will be working with to build this model. For simplicity the data_set_ols will still be referred to as the merged dataset as it is just a greatly refined version of the original dataset.

The baseline linear model that we will be using to construct our first model will use all the remaining variables in the merged dataset which results in the following linear model:

units ~ date + station_nbr + store_nbr + item_nbr + tavg + snowfall + preciptotal
+ stnpressure + sealevel + avgspeed

This is simply done by using the built-in linear model function, lm, with the model shown above. Upon running the model, we get a nice list of estimate coefficients and model statistics (see Figure7). Our preliminary linear model has 8 predictors that are statistically significant up to an alpha value of 0.01, 1 predictor (preciptotal) statistically significant up to an alpha value of 0.10, and 2 predictors that were not statistically significant (intercept and snowfall). Now once we look at our model statistics, we see that even though our model has many nice significant predictors, it does very poorly in describing the variation in our dataset given that the Multiple R-Squared and Adjusted R-Squared were both 0.162. These scoring statistics raise quite a few

alarms and indicates that not only is there something wrong with the preliminary model diagnostically speaking, but it is likely that it is going to preform very poorly when it comes to predictions.

Interpretations

Before we talk about the diagnostics of this model, we'll take a moment to look at some of the model estimates (See Figure7). It should be quite apparent that quite of few of these predictors should not be linear in nature so there are some obvious shortcomings of this model. For one, date as we know cycles since we go by a yearly calendar but the way in which we have created a numerical value of date makes it a continuous variable and so it doesn't really cycle. That's why when we see an estimate of -0.0102 for date it makes us realize that this linear estimate is likely not very good unless sales are decreasing year by year. Of course, each of these estimates is interpreted as if all other variables are held constant so a 1-unit increase in the variable we are considering would lead to an increase or decrease in the number of units predicted by that variable estimate. Station_nbr, store_nbr, and item_nbr could be interrupted the same way but considering that these are basically id variables not much importance should be placed on them. Tavg is 0.037 which is positive which makes sense since the warmer it is the more likely one would go shopping. Snowfall and preciptotal are both negative which again makes sense since worsening weather conditions make it less like for one to go shopping. Stnpressure and sealevel are very big estimates so either they have a very big effect or there may be some underlying conflictions such as the possibility that there are more store locations at certain sealevels and stnpressures values so obviously there would be more unit sales however we do not have geographical data to check this so we'll just take them at face value. Lastly, we

have avgspeed at 0.176 and surprisingly this estimate is positive which would indicate that as the average wind speed goes up so do unit sales which doesn't entirely make sense.

Diagnostics

Before we run predictions, it would be wise to check the diagnostics of this model, so we have an idea of what to fix in further iterations and improvements. Using the built-in variance inflation factor function (see Figure8) we find that none of the involved variables have any significant VIF's so multicollinearity is almost certainly not an issue in this model. Looking at the diagnostic plots (see Figure9) we find the bulk of the issues with this model.

Firstly, in the Residuals vs Fitted plot it isn't even necessary to do the Breusch-Pagan test since the residuals are clearly heteroscedastic. The Normal Q-Q plot shows that the residuals are nowhere near normally distributed with many points deviating from the normal line, so the normality assumption of our model fails meaning all the significant predictors of the model are pretty much worthless. A glance at the Residuals vs Leverage plot would indicate that there don't seem to be any outliers not that it much matters at this point.

Given all the shortcomings of this model we are still going to run predictions using the predict function since we are still trying to get a baseline with this preliminary model. Again, it is worth reminding that although we will be using the predict function to get predicted values for every observation in the test dataset, we will be going through the test dataset afterwards and setting 0's for the predicted units for all of the item-store combinations that had mean 0 for units. This is what we observed in the Data Exploratory section so it would make sense to continue predicting 0's for them. The test data set will then be converted into the prediction data set, which will ultimately be exported to a csv and submitted to Kaggle, by adding the id and dropping the remaining unnecessary variables leaving only the id and units predicted.

(See section ‘Prediction - Using Baseline Linear Model’ in the R code, Figure18)

Score

After everything described above is finished and we submit the csv to Kaggle, we get a private score of 0.42542 and public score of 0.42385 for our preliminary model which was a little better than anticipated (see Figure10 for Kaggle score confirmation). What likely really helped our score was the fact that we predicted all 0’s for the item-store combinations that had units mean 0. Overall moving forward there is room to improve to make the model less complex, have a better performance, and have fewer diagnostic issues which leads us to the next section.

Improvements

Linear Model Log Transformed

Model

Given that in our preliminary model the residuals were not normal and thus we failed to assume the assumption of normality, our first approach to hopefully fix the residuals is some type of transformation. BoxCox was the first method considered however given the negative values in a few of the coefficient estimates, trying to correct the values and making them positive brings in quite a few complications in interpretation so a simple log transformation was decided. The new model that shall be considered is a log-linear model defined as:

$$\text{Log(units+1)} \sim \text{date} + \text{station\_nbr} + \text{store\_nbr} + \text{item\_nbr} + \text{tavg} + \text{snowfall} + \text{preciptotal} + \text{stnpressure} + \text{sealevel} + \text{avgspeed}$$

The addition of the constant within the log is necessary since many of the units observations are 0's so taking the log would give negative infinity. This isn't a problem so long as we account for it when making predictions. Running the lm function on this new model yields a decently improved model with somewhat better model statistics.

From the summary report (See Figure11) we see that all are predictors are significant except the intercept and only one is significant at an alpha of 0.01 while the rest were significant at an alpha of 0.001. Looking at the model statistics, we see that the new log transformed model has a slight improved Multiple R-Squared and Adjusted R-Squared scores at 0.324 and 0.323 respectively which is still not great, but it is an improvement.

Diagnostics

Just to confirm that nothing horribly wrong occurred VIF's were rechecked (See Figure12) and again none of the VIF values raise any concern of multicollinearity. Looking at

the diagnostics plots (See Figure13) we see right away that the residuals are still heteroscedastic, but the normality of the residuals has made great improvement. Now given the curves at the ends of the Q-Q plot, it would appear that our residuals are still not normally distributed but it could be argued that given the size of the data set that we could say it is approximately normal. Regardless, since there are no obvious outliers we will move onto predictions.

Predictions

As was mentioned before, after the function predict is ran and all the predicted values are put into the test dataset, the predictions belonging to any of the item-store combinations that had 0 mean for units will be reset to predict 0's. The test data set will again have the id added to it and have unnecessary variables removed so that only the id and prediction remain which will than be exported to a csv and submitted to Kaggle. Before submitting it is important that the predictions are rid of the log transform by using exp(prediction) -1 to undo the transformation.

(See section 'Prediction - Using Log transformed model' in the R code, Figure18)

Score

After submitting the csv to Kaggle, we get a private score of 0.35388 and public score of 0.35238 which is up from are preliminary model which is a good start (See Figure14 for score confirmation). This much of an improvement was somewhat unexpected so in hopes of increasing our score even more we will search for further improvements.

AIC Variable Selection on the Log-Linear Model

In hopes of improving our log-linear model, variable selection was preformed on the log-linear model using forward and backward selection looking at AIC. Sadly (as Figure15 and Figure16 show), there was no real improvement to be made in terms of variable selection. Both the forward and backward model selections both favored the full model which is the precisely the one that was already used. Although variable selection was not done using BIC instead, it is very likely that the results would have been the same.

Given that variable selection yielded no improvements, we will examine a different method that is quite a bit simpler in the way it approaches the data.

K Nearest Neighbor (Final Model)

Approach

In Stat 448, we were introduced to the idea of K Means Clustering which is an algorithm that focuses on classifying data based on group means thus highlighting underlying characteristics and patterns of the dataset. This approach is particularly appealing since it doesn't assume much but simply looks for patterns within the data that regular people may not find. Given that K Means Clustering doesn't exactly apply here since it requires labeling data into typically factor like variables, we will use the analogous approach of K Nearest Neighbor which can be applied regression problems.

Model

Unlike our previous linear or log-linear models, K Nearest Neighbor simply looks at the data. It does not focus on all these different types of predictors nor does it rate the significance of predictors or require assumptions like normality. K Nearest Neighbor simply applies the same

concept of K Means Clustering but can be used in a regression problem. It basically looks for underlying patterns in the data, which in this case will just be the date and the units sold, by making groupings of the k nearest neighbors. For example, if k=2 than it will build its model looking for characteristics within groupings of the 2 nearest neighbors of any particular data point. Unlike the linear or log linear model produced earlier, K Nearest Neighbor doesn't produce some straightforward simple relationship but can still be used to make predictions which is exactly what we'll be using it for. For the selection of k, the code seen in Firgure18 section 'Prediction - KNN model' was run with a for loop that tested k values between 1 and 30 by submitting the predictions to Kaggle. K=30 received the best score so it is the one that will be used.

Prepping

K Nearest Neighbor requires a training dataset which will be our merged data set with all the 0 units mean item-store combinations removed. The only variables we will require are the numeric date values and the units sold. The dates will be the independent variable and units will be the dependent variable or the response. Luckily, the way that we apply K Nearest Neighbors, we will produce a prediction data frame that has all 0's and we will add in the predicted values for each of the item-store combinations that do not have 0 units means. It is important to note that just like we did in the log-linear model, we will be using a log plus 1 transformed units response variable because typically K Nearest Neighbors requires relatively nice or normalized data. Performing K Nearest Neighbors without the transformation ends with very bad results.

(See section 'Prediction - KNN model' in the R code, Figure18)

Prediction

Once we run the K Nearest Neighbors algorithm and have all the predicted values in our prediction dataset with the id's in as we have done before, the units will need to be transformed back using exp(prediction)-1 again just like in the log-linear model. There isn't much in terms of diagnostics that we can preform so we will just submit the predictions to Kaggle to see how well it does. The Kaggle score for K Nearest Neighbors with k=30 achieved a private score of .10605 and public score of .10734 which is by far the best score achieved and thus is our final model. (See Figure17 for score confirmation)

(In order of references to)

Figure7: Summary statistics for the preliminary linear model

```
Call:
lm(formula = units ~ ., data = data_set_ols)

Residuals:
   Min     1Q Median     3Q    Max
 -63.3  -20.2    -6.9    5.4  534.5

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 30.11076   23.03054    1.31     0.19
date        -0.01022    0.00037  -27.60  < 2e-16 ***
station_nbr -0.25584    0.02135  -11.98  < 2e-16 ***
store_nbr    0.21444    0.00885   24.22  < 2e-16 ***
item_nbr    -0.44805    0.00321 -139.57  < 2e-16 ***
tavg         0.03758    0.00699    5.38  7.6e-08 ***
snowfall    -0.33156    0.30394   -1.09     0.28
preciptotal -0.59965    0.36436   -1.65     0.10 .
stnpressure -9.87943    0.28234  -34.99  < 2e-16 ***
sealevel    10.02621    0.82125   12.21  < 2e-16 ***
avgspeed     0.17641    0.03100    5.69  1.3e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 38.2 on 120028 degrees of freedom
Multiple R-squared:  0.162,     Adjusted R-squared:  0.162
F-statistic: 2.33e+03 on 10 and 120028 DF,  p-value: <2e-16
```

Figure8: VIF for preliminary linear model

date	station_nbr	store_nbr	item_nbr	tavg	snowfall	preciptotal	stnpressure	sealevel	avgspeed
1.01618	1.09161	1.08180	1.02336	1.42991	1.03544	1.04957	1.62739	1.81012	1.37957

Figure9: Diagnostic plots for preliminary linear model

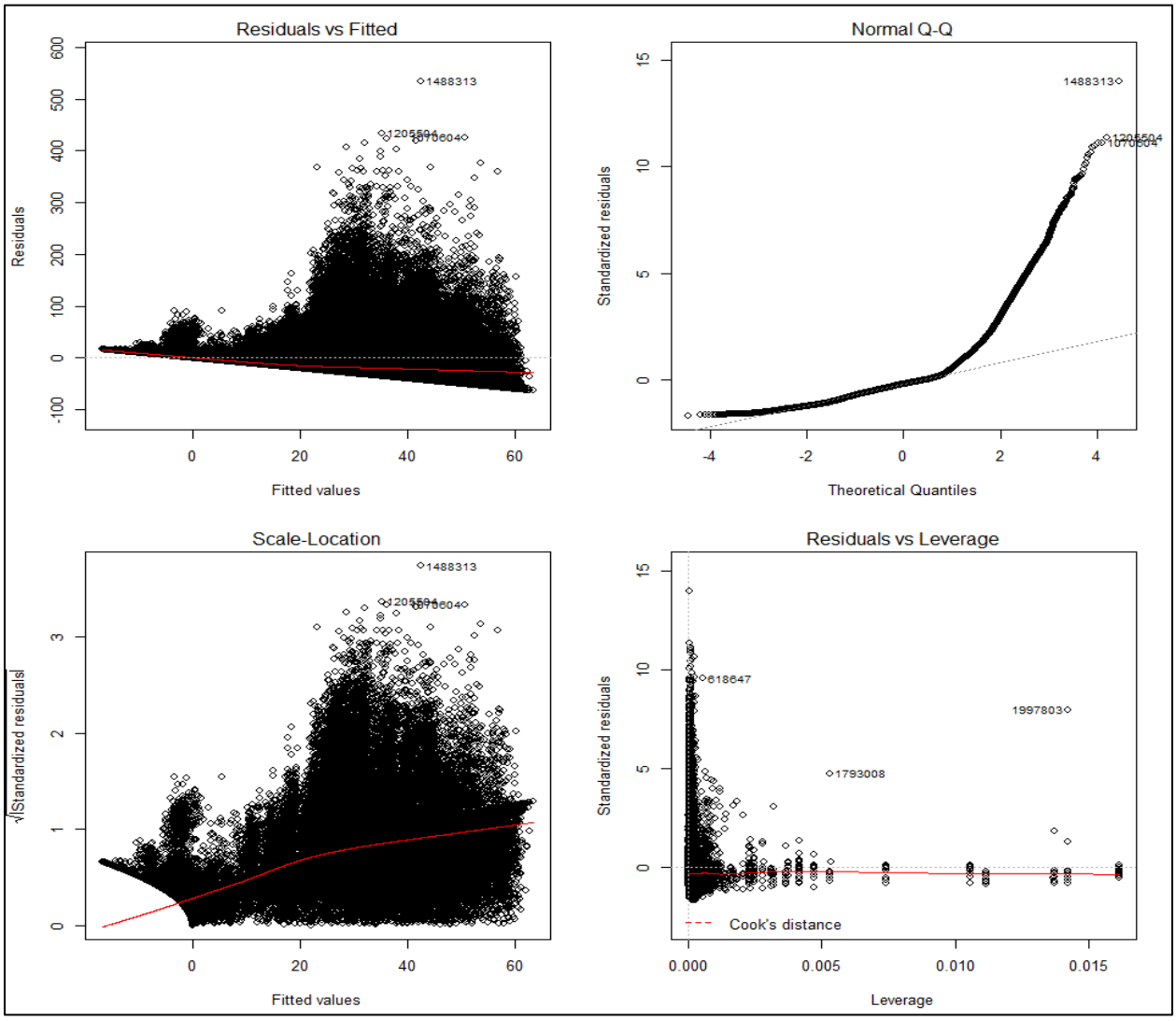


Figure10: Kaggle score for the preliminary linear model

Submission and Description	Private Score	Public Score
submit_preliminary_linear_final.csv a few seconds ago by Mark Belsis Preliminary linear model FINAL	0.42542	0.42385

Figure11: Summary statistics for log-linear model

```
Call:
lm(formula = log(units + 1) ~ ., data = data_set_ols)

Residuals:
    Min      1Q  Median      3Q     Max
-3.646 -0.875 -0.179  1.036  4.256

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  8.73e-01   9.08e-01    0.96  0.33625
date        -2.94e-04   1.46e-05  -20.14  < 2e-16 ***
station_nbr -1.38e-02   8.42e-04  -16.42  < 2e-16 ***
store_nbr    9.26e-03   3.49e-04   26.55  < 2e-16 ***
item_nbr    -2.97e-02   1.27e-04 -234.82  < 2e-16 ***
tavg         2.76e-03   2.75e-04   10.02  < 2e-16 ***
snowfall    -4.51e-02   1.20e-02   -3.76  0.00017 ***
preciptotal  3.73e-02   1.44e-02    2.59  0.00947 **
stnpressure -1.73e-01   1.11e-02  -15.53  < 2e-16 ***
sealevel     2.38e-01   3.24e-02    7.35  2.0e-13 ***
avgspeed     6.08e-03   1.22e-03    4.97  6.6e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.51 on 120028 degrees of freedom
Multiple R-squared:  0.324,     Adjusted R-squared:  0.323
F-statistic: 5.74e+03 on 10 and 120028 DF,  p-value: <2e-16
```

Figure12: VIF for log-linear model

date	station_nbr	store_nbr	item_nbr	tavg	snowfall	preciptotal	stnpressure	sealevel	avgspeed
1.01618	1.09161	1.08180	1.02336	1.42991	1.03544	1.04957	1.62739	1.81012	1.37957

Figure13: Diagnostic plots for log-linear model

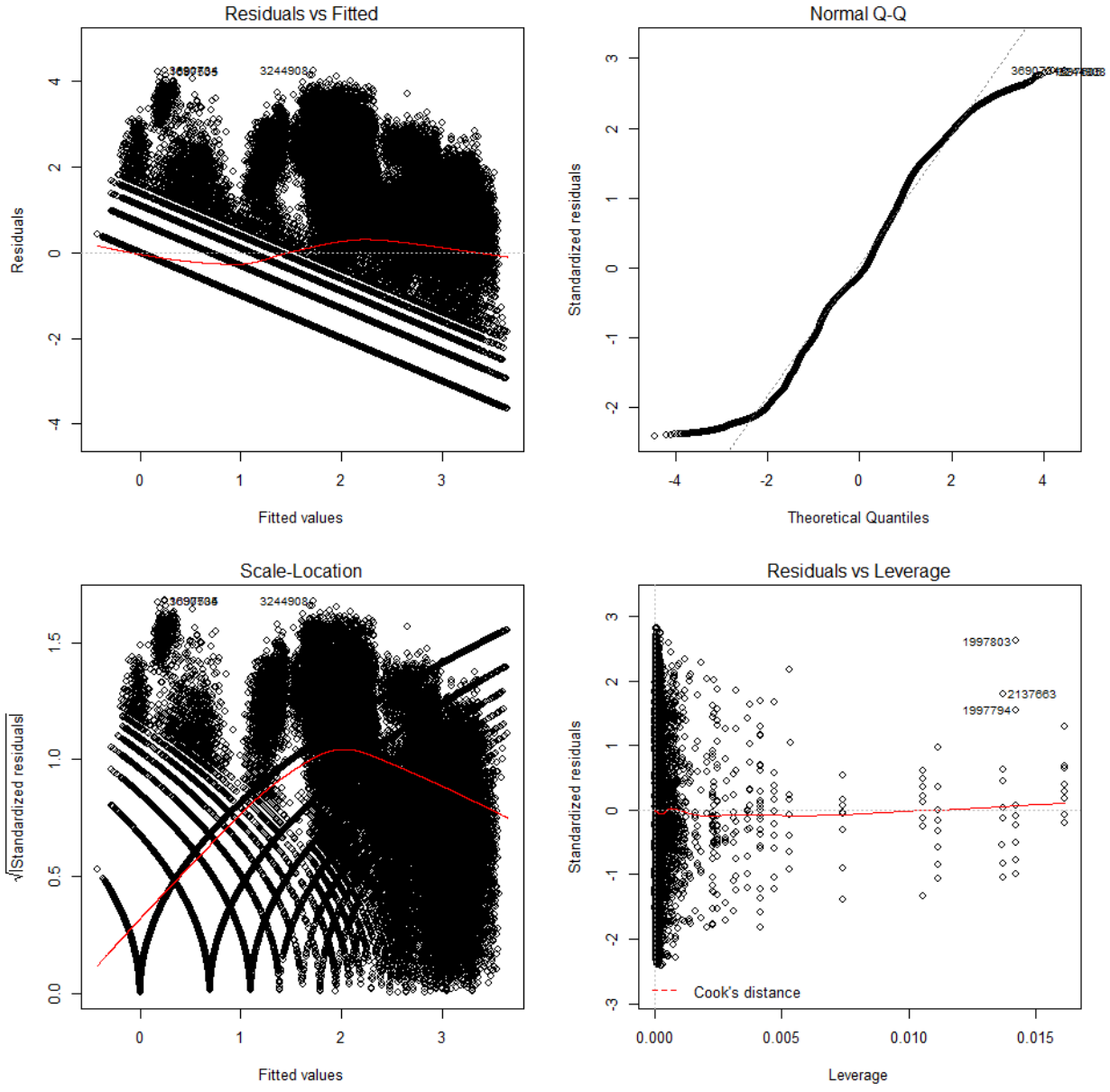


Figure14: Kaggle Score for log-linear model

Submission and Description	Private Score	Public Score
submit_loglinear_final.csv 5 minutes ago by Mark Belsis Log Linear Model FINAL	0.35388	0.35238

Figure15: Backward selection for log-linear model

```
Start:  AIC=98523.2
log(units + 1) ~ date + station_nbr + store_nbr + item_nbr +
    tavg + snowfall + preciptotal + stnpressure + sealevel +
    avgspeed

              Df Sum of Sq    RSS    AIC
<none>                      272706  98523
- preciptotal  1        15 272721  98528
- snowfall     1        32 272738  98535
- avgspeed     1        56 272762  98546
- sealevel     1       123 272828  98575
- tavg         1       228 272934  98622
- stnpressure  1       548 273254  98762
- station_nbr  1       613 273319  98791
- date         1       921 273627  98926
- store_nbr    1      1601 274307  99224
- item_nbr     1    125276 397981 143897
```

Figure16: Last step of forward selection for log-linear model

```
Step:  AIC=98528
log(units + 1) ~ item_nbr + store_nbr + stnpressure + date +
    station_nbr + tavg + sealevel + avgspeed + snowfall

              Df Sum of Sq    RSS   AIC
+ preciptotal  1      15.3 272706 98523
<none>                     272721 98528

Step:  AIC=98523.2
log(units + 1) ~ item_nbr + store_nbr + stnpressure + date +
    station_nbr + tavg + sealevel + avgspeed + snowfall + preciptotal
```

Figure17: Kaggle score for K Nearest Neighbor (k=30) [BEST SCORE]

Submission and Description	Private Score	Public Score
submit_KNN_k30_FINAL.csv 6 minutes ago by Mark Belsis K Nearest Neighbor (k=30) FINAL	0.10605	0.10734

Figure18: All R Code (Highlighted areas are the beginning and ends of code blocks)

```
---
title: "Final_Project"
author: "Mark Belsis"
date: "November 18, 2019"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```


```{r Data Introduction, echo=TRUE , eval=TRUE, warning = FALSE, message = FALSE}

weather = read.csv("weather.csv")   # Weather data for each date between 1/1/2012 and 3/31/2013 (20,517 observations, 20 variables)
train_data = read.csv("train.csv")  # Dates between 1/1/2012 and 3/31/2013, provides date, store ID, item ID, and number of units sold on that given day (4,617,600 observations, 4 variables)
test_data = read.csv("test.csv")    # Dates after 4/1/2013 to 10/16/2014 (526,917 observations, 3 variables)
key = read.csv("key.csv")           # Matches a store ID number with a weather station ID (45 observations, 2 variables)

```


```{r Library Initilization, echo=TRUE , eval=TRUE, warning = FALSE, message = FALSE}
#install.packages("doBy")
#install.packages("psych")
#install.packages("FNN")
library(psych)
library(doBy)
library(car)
library(FNN)
```
```

```{r Data Initilization, echo=TRUE , eval=TRUE, warning = FALSE, message = FALSE}

#Here we are going to make a vector of the station_nbr's that corresponds to the store_nbr in the training data set so we need a vector of length 4,617,600 rows.
station_nbr = numeric(nrow(train_data))


#Here we are going through each observation and attch the correct corresponding station_nbr using the key.csv
for (i in 1:nrow(train_data) ){
  station_nbr[i] = key$station_nbr[which(train_data$store_nbr[i] == key$store_nbr)]
}

# Combining the station_nbr with the training data.
train_data_withstation = cbind(train_data,station_nbr)


# Now we merge all the weather data with the training data using merge and the option 'by' which will match each observation with the correct weather data using the date and station_nbr.
full_training_data = merge(train_data_withstation,weather, by = c("date","station_nbr"))

# Now unforutanely the above merge fucntion doesnt really care about the nice ordered strucutre of the data so it scrables the stores order so in order to fix this just for easier viewing we will
#   reorder the date store_nbr and item_nbr
full_training_data =
full_training_data[order(full_training_data$date,full_training_data$store_nbr,full_training_data$item_nbr),]
row.names(full_training_data) <- 1:nrow(full_training_data) ###### CHECK

# To confirm that we havent accidently messed up any of the data we'll check that each of these columns below have the same values by taking the sum which should be 4617600 which is what we get
sum((full_training_data$store_nbr == train_data$store_nbr) & (full_training_data$item_nbr == train_data$item_nbr) & (full_training_data$date == train_data$date) )

```

```{r Data Prepping (Removing zeros), echo=TRUE , eval=TRUE, warning = FALSE, message = FALSE}

# First we are gonna use the psych library to get a description of the units using the item_nbr and store_nbr varialbes so we can see which combinations have 0 unit sales
#   Which would indicate a product that is not sold at a particular store so we can ignore it
desc = describe.by(full_training_data$units,
list(full_training_data$item_nbr,full_training_data$store_nbr),mat=TRUE)

# Two data_frames that contain the stores/items to remove and the stores/items to save
toremove = desc[which(desc$mean==0),]
toremove
tokeep = desc[which(desc$mean!=0),]
tokeep

# Copy of the full data to manipulate
full_training_data_nozeros = full_training_data

# Painfully long for loop that goes in and removes each entry that belongs to a item/store comboniation that ultimately had 0 unit sales since which we'll just predict them to have 0 sales
for ( i in 1:nrow(toremove) ){
  full_training_data_nozeros = full_training_data_nozeros[-which((toremove$group1[i] == full_training_data_nozeros$item_nbr) & (toremove$group2[i] == full_training_data_nozeros$store_nbr)),]
  #if ( i %% 10 == 0 ){ print(i)} # Allows me to keep track of the progress to ensure the loop hasnt frozen
}

# Create a copy of the full data with nozeros so we dont have to rerun the above for loop
train_data_copy = full_training_data_nozeros

# Renumber the rows
row.names(train_data_copy) <- 1:nrow(train_data_copy)

# Use a new decription fucntion cause there were errors with the desribe.By one just to confirm that all the remaining stores/item combinations have non-zero unit means
desc_zeros = summaryBy(units ~ store_nbr + item_nbr, data = train_data_copy,
  FUN = function(x) { c(m = mean(x), s = sd(x)) } )


```

```{r Cleaning, echo=TRUE , eval=TRUE, warning = FALSE, message = FALSE}

options(digits=6) # Setting the number of significant digits since as.numeric will drop decimals without it

train_data_copy = full_training_data_nozeros # Copy in the data
train_data_copy$date = as.numeric(train_data_copy$date) - 456 # Set date as numerics with respect to 3/31/2013 (456 days from 1/1/2012) since this is the day before the test dates begin to occur

train_data_copy = train_data_copy[,-c(9:15)] # Delete Depart, Dewpoint, wetbulb, heat, cool, sunrise, sunset as

summary(train_data_copy)
# We see that lots of the data contains factors and missing values such as M for Missing or T for Trace. To make our lives easier we will remove these entries although Trace could just be set to zero
#  We will also convert all but codesum from factors to numerical values as they shoud be.

# Removing observations with "M" from tavg,tmin,tmax and setting as numeric
train_data_copy = subset(train_data_copy, tmax != "M" | tmin != "M" | tavg != "M" )
train_data_copy$tavg = as.numeric(train_data_copy$tavg)
train_data_copy$tmin = as.numeric(train_data_copy$tmin)
train_data_copy$tmax = as.numeric(train_data_copy$tmax)

# Removing observations with "M" and "T" from snowfall, preciptotal and setting as numeric
train_data_copy$snowfall = as.character(train_data_copy$snowfall)
train_data_copy$preciptotal = as.character(train_data_copy$preciptotal)
train_data_copy = subset(train_data_copy, snowfall != "M")
train_data_copy = subset(train_data_copy, snowfall != "T")
train_data_copy = subset(train_data_copy, preciptotal != "M")
train_data_copy = subset(train_data_copy, preciptotal != "T")
train_data_copy$snowfall = as.numeric(as.character(train_data_copy$snowfall))
train_data_copy$preciptotal = as.numeric(as.character(train_data_copy$preciptotal))

# Removing observations with "M" from stnpressure,sealevel,resultspeed,resultdir,avgspeed and setting as numeric
train_data_copy$stnpressure = as.character(train_data_copy$stnpressure)
train_data_copy$sealevel = as.character(train_data_copy$sealevel)
train_data_copy$resultspeed = as.character(train_data_copy$resultspeed)
train_data_copy$resultdir = as.character(train_data_copy$resultdir)
train_data_copy$avgspeed = as.character(train_data_copy$avgspeed)
train_data_copy = subset(train_data_copy, stnpressure != "M")
train_data_copy = subset(train_data_copy, sealevel != "M")
train_data_copy = subset(train_data_copy, resultspeed != "M")
train_data_copy = subset(train_data_copy, resultdir != "M")
```

```
train_data_copy = subset(train_data_copy, avgspeed != "M")
train_data_copy$stnpressure = as.numeric(as.character(train_data_copy$stnpressure))
train_data_copy$sealevel = as.numeric(as.character(train_data_copy$sealevel))
train_data_copy$resultspeed = as.numeric(as.character(train_data_copy$resultspeed))
train_data_copy$resultdir = as.numeric(as.character(train_data_copy$resultdir))
train_data_copy$avgspeed = as.numeric(as.character(train_data_copy$avgspeed))
train_data_copy = na.omit(train_data_copy)

summary(train_data_copy) # And as we can see all variables beside codesum are now numerical values
```

```{r Diagnostics and Data Discovery, echo=TRUE , eval=TRUE, warning = FALSE, message = FALSE, fig.width = 13, fig.height = 13}

##### PAIRWISE PLOTS #####
# Given the amount of points taking a small subset allows us to plot some data and discern any chararteristics of the data
sampling = sample(1:nrow(train_data_copy), round(nrow(train_data_copy)/50))
train_data_copy_plotsamples = train_data_copy[sampling,]

pairs(~.,data=train_data_copy_plotsamples) # Pairwise plots of each of the remaining variables
############################


##### Correlation? #####
cor.tmax.tmin = cor(train_data_copy$tmin,train_data_copy$tmax)
cor.tmax.tavg = cor(train_data_copy$tmax,train_data_copy$tavg)
cor.tmin.tavg = cor(train_data_copy$tmin,train_data_copy$tavg)
cor.sealevel.stnpressure = cor(train_data_copy$sealevel,train_data_copy$stnpressure)
cor.resultspeed.resultdir = cor(train_data_copy$resultspeed,train_data_copy$resultdir)
cor.resultspeed.avgspeed = cor(train_data_copy$resultspeed,train_data_copy$avgspeed)
########################
```

```{r Baseline Linear Model, echo=TRUE , eval=TRUE, warning = FALSE, message = FALSE, fig.width = 10, fig.height = 10}

data_set_ols = train_data_copy[,-c(6:7,9,14:15)]
```

```
# The model we will be using as our baseline linear model will include the variables date, station_nbr, store_nbr, item_nbr, units (Response), tavg, snowfall, preciptotal, stnpressure, sealevel, and avgspeed
#       The reason for choosing these is stated in the paper

##### Model #####  --- Kaggle score: Private = 0.42542, Public = 0.42385
selected_model = lm(units ~ ., data = data_set_ols )
summary(selected_model)
#################

##### Diagnostics Model #####
par(mfrow=c(2,2))
plot(selected_model)
vif(selected_model)
############################

##### Improved Model ##### --- Kaggle score: Private = 0.35388, Public = 0.35238
selected_model_improved = lm(log(units+1) ~ ., data = data_set_ols )
summary(selected_model_improved)
#########################

##### Diagnostics Improved Model #####
par(mfrow=c(2,2))
plot(selected_model_improved)
vif(selected_model_improved)
#####################################

```

```{r Prediction - Using Baseline Linear Model, echo=TRUE , eval=TRUE, warning = FALSE, message = FALSE, fig.width = 10, fig.height = 10}

prediction = test_data # Our predictions will be built off of the test data set since it contains all the dates and item/store combos we need to predict
id = numeric(nrow(prediction)) # The Kaggle submission requires a id

# The for loop below will be used to create a new column in prediction with all the id's by assigning the ids to a vector to be combined later
for (i in 1:nrow(prediction)){
  id[i] = paste0(prediction[i,2],"_",prediction[i,3],"_",prediction[i,1])
}

prediction = cbind(prediction,id) # And its later, here we combine the vector and prediction
units = numeric(nrow(prediction)) # Now we are going to do the same thing as above but for the units we will be predicting (These are just place holders)

for (i in 1:nrow(prediction)){
  units[i] = 0
}

prediction = cbind(prediction,units) # Again combine them

# Since we have no weather data for any of these dates we need to predict we will use the means
average_tavg = rep(mean(data_set_ols$tavg))
average_snowfall = rep(mean(data_set_ols$snowfall))
average_preciptotal = rep(mean(data_set_ols$preciptotal))
average_stnpressure = rep(mean(data_set_ols$stnpressure))
average_sealevel = rep(mean(data_set_ols$sealevel))
average_avgspeed = rep(mean(data_set_ols$avgspeed))

# We do know the dates, stores, items, and stations we will be predicting for
dates = as.numeric(prediction$date)
stores = prediction$store_nbr
items = prediction$item_nbr

station = numeric(nrow(prediction))
for (t in 1:nrow(prediction)){
  station[t] =  key[which(key$store_nbr == stores[t]),2]
}

# Simple prediction data frame to be used in the predict function
pred_data = data.frame(date = dates, store_nbr = stores, item_nbr = items, station_nbr = station , tavg = average_tavg, snowfall = average_snowfall, preciptotal = average_preciptotal
                       , stnpressure = average_stnpressure , sealevel = average_sealevel, avgspeed = average_avgspeed)
```

```
predict_fit = predict(selected_model, pred_data, interval = "prediction", level = 0.95)  # Predict values
prediction$units = round(predict_fit[,1])

# This for loop will go through every value in prediction for units and if there is a negative value it will turn it into 0 since we arent going to predict negative unit sales
for(i in 1:nrow(prediction)){
  #if (i %% 1000 == 0){print(i)} # Used to monitor progress
  if(prediction$units[i] < 0 )
    {
    prediction$units[i] = 0
    next}
}

# This for loop is going to go through all the store and item combinations that lead to 0 mean unit sales and reset them to 0 since our regression prediction is continous and changed their values
#   so we are gonna change them back to 0.
for(k in 1:nrow(toremove)){
    # if( k %% 10 == 0){print(k)} # Used to monitor progress
    prediction$units[which(prediction$store_nbr == toremove$group2[k] & prediction$item_nbr == toremove$group1[k])] = 0
}

submit = prediction[,-c(1:3)]  #We are going to delete the first 3 columns since they arent suppose to be in the submisson file
write.csv(submit,"submit_preliminary_linear_final.csv", row.names = FALSE)
```

```{r Prediction - Using Log transformed model, echo=TRUE , eval=TRUE, warning = FALSE, message = FALSE, fig.width = 10, fig.height = 10}

prediction = test_data # Our predictions will be built off of the test data set since it contains all the dates and item/store combos we need to predict
id = numeric(nrow(prediction)) # The Kaggle submission requires a id

# The for loop below will be used to create a new column in prediction with all the id's by assigning the ids to a vector to be combined later
for (i in 1:nrow(prediction)){
  id[i] = paste0(prediction[i,2],"_",prediction[i,3],"_",prediction[i,1])
}

prediction = cbind(prediction,id) # And its later, here we combine the vector and prediction
units = numeric(nrow(prediction)) # Now we are going to do the same thing as above but for the units we will be predicting (These are just place holders)

for (i in 1:nrow(prediction)){
  units[i] = 0
}

prediction = cbind(prediction,units) # Again combine them

# Since we have no weather data for any of these dates we need to predict we will use the means
average_tavg = rep(mean(data_set_ols$tavg))
average_snowfall = rep(mean(data_set_ols$snowfall))
average_preciptotal = rep(mean(data_set_ols$preciptotal))
average_stnpressure = rep(mean(data_set_ols$stnpressure))
average_sealevel = rep(mean(data_set_ols$sealevel))
average_avgspeed = rep(mean(data_set_ols$avgspeed))

# We do know the dates, stores, items, and stations we will be predicting for
dates = as.numeric(prediction$date)
stores = prediction$store_nbr
items = prediction$item_nbr

station = numeric(nrow(prediction))
for (t in 1:nrow(prediction)){
  station[t] =  key[which(key$store_nbr == stores[t]),2]
}

# Simple prediction data frame to be used in the predict function
pred_data = data.frame(date = dates, store_nbr = stores, item_nbr = items, station_nbr = station , tavg = average_tavg, snowfall = average_snowfall, preciptotal = average_preciptotal
                       , stnpressure = average_stnpressure , sealevel = average_sealevel, avgspeed = average_avgspeed)
```

```
predict_fit = predict(selected_model_improved, pred_data, interval = "prediction", level = 0.95)
# Predict values
prediction$units = round(exp(predict_fit[,1])+1) # Undo the log

# This for loop will go through every value in prediction for units and if there is a negative value
it will turn it into 0 since we arent going to predict negative unit sales
for(i in 1:nrow(prediction)){
  #if (i %% 1000 == 0){print(i)} # Used to monitor progress
  if(prediction$units[i] < 0 )
    {
    prediction$units[i] = 0
    next}
}

# This for loop is going to go through all the store and item combinations that lead to 0 mean unit
sales and reset them to 0 since our regression prediction is continous and changed their values
#   so we are gonna change them back to 0.
for(k in 1:nrow(toremove)){
    # if( k %% 10 == 0){print(k)} # Used to monitor progress
    prediction$units[which(prediction$store_nbr == toremove$group2[k] & prediction$item_nbr
== toremove$group1[k])] = 0
}

submit = prediction[,-c(1:3)]  #We are going to delete the first 3 columns since they arent
suppose to be in the submisson file
write.csv(submit,"submit_loglinear_final.csv", row.names = FALSE)
```

```{r Improvements - Variable Selection, echo=TRUE , eval=TRUE, warning = FALSE, message = FALSE}

log_linear_backward_selection_AIC = step(selected_model_improved)

null_model = lm(log(units + 1) ~ 1, data=data_set_ols )
log_linear_forward_selection_AIC = step(null_model, direction = c("forward"), scope = list(upper = selected_model_improved, lower = null_model))

```

```{r Prediction - KNN model, echo=TRUE , eval=TRUE, warning = FALSE, message = FALSE}

##### Same prediction setup as above #####
prediction = test_data
id = numeric(nrow(prediction))

for (i in 1:nrow(prediction)){
  id[i] = paste0(prediction[i,2],"_",prediction[i,3],"_",prediction[i,1])
}

prediction = cbind(prediction,id)


units = numeric(nrow(prediction))
for (i in 1:nrow(prediction)){
  units[i] = 0
}

prediction = cbind(prediction,units)
##########################################



##### K Nearest Neighbor #####   --- Kaggle score: Private = 0.10605, Public = 0.10734
train_data_copy = full_training_data_nozeros # Make a copy of the full_training data set with the zero means removed so we dont have to rerun the cancer loop
train_data_copy$date = as.numeric(train_data_copy$date) - 456 # Again set the dates in relation too 3/31/2013

knn_storeitem_combo = test_data # Create a copy of test data so as not to mess with it
knn_storeitem_combo$date = as.numeric(knn_storeitem_combo$date) # Set dates as numerics (These are the dates we want to predict)

# The loop below will cycle through all the entries in the tokeep list (Which is all the store and item combos that had non-zero unit means) and run the knn algorithm and compute the predicted number of units
#   sold for the dates within the test/prediction data set and will assign the predicted values (Without chaning the predicted zeros from the zero-mean combinations)
for ( i in 1:nrow(tokeep)){

  # We'll create a subset of the test data which are in the tokeep data set which contains the store and item combos that had non-zero unit means
  knn_store_item = subset(train_data_copy, train_data_copy$store_nbr == tokeep$group2[i] & train_data_copy$item_nbr == tokeep$group1[i])
```

```
  # Remove all the unused data, we only need date and units
  knn_store_item = knn_store_item[,-c(2:4,6:23)]
  knn_store_item_train = knn_store_item
  knn_store_item_test = subset(knn_storeitem_combo, knn_storeitem_combo$store_nbr ==
tokeep$group2[i] & knn_storeitem_combo$item_nbr == tokeep$group1[i])

  if (nrow(knn_store_item_test) == 0){next}

  # As we did in the improved linear model, we will preform a log transformation to the response
variables since it fixes unwanted behavior (Running knn without transformation produces very
poor model)
  knn_store_item_train$units = (log(1 + knn_store_item_train$units))

  # Actual KNN regression function from FNN library
  predict_knn_store_item = knn.reg( train = as.data.frame(knn_store_item_train[,1]), test =
as.data.frame(knn_store_item_test[,1]), y = knn_store_item_train[,2] , k=30)

  # Set the predicted values to units
  units = predict_knn_store_item$pred
  knn_store_item_test$units = units
  knn_store_item_test$units = round(exp(knn_store_item_test$units)-1) # Undo log
transformation

  # Set the predictions into prediction
  prediction$units[which(prediction$store_nbr == tokeep$group2[i] & prediction$item_nbr ==
tokeep$group1[i])] = knn_store_item_test$units

}

submit = prediction[,-c(1:3)]
write.csv(submit,"submit_KNN_k30_FINAL.csv", row.names = FALSE)
##################################

```