Final Report

Zak Kaplan and Jason Klotz

July 24, 2020

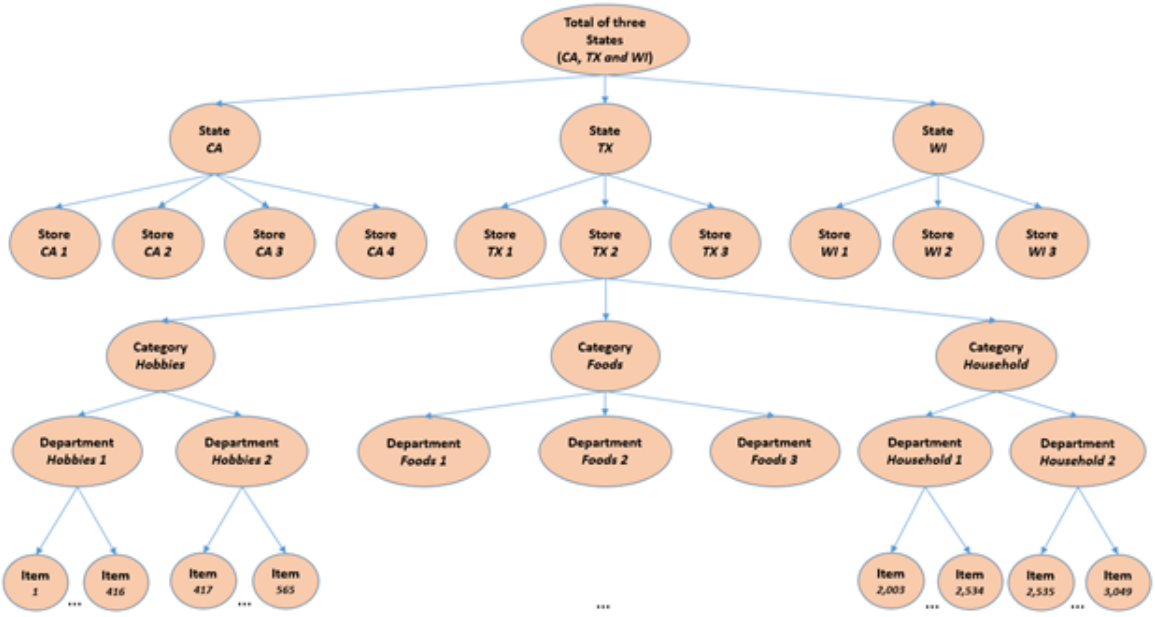
1 Problem Statement

For our project, our goal was to create a model or method to accurately and efficiently predict sales for a variety of items at Wal-Mart. We have access to the sales data for these items for over 5 years, with very little metadata about each item. We want to be able to forecast the next 28 days for each of the items with as little error as possible.

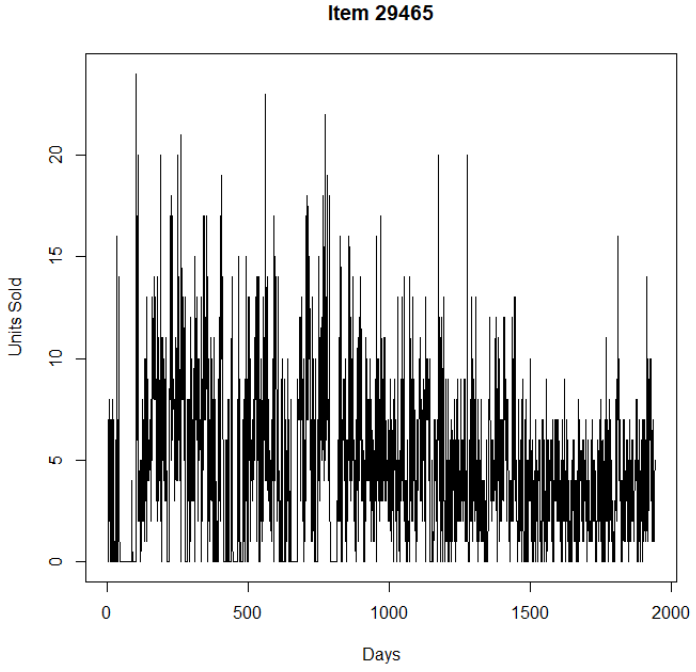
2 Introduction

2.1 Data

We were provided many different data files with our project. Information about the dates that the products were sold, daily unit sales and the information about each products sell price. We chose to focus mainly on the daily unit sale price data which consisted of 30490 items that were sold at 10 different Walmart locations. The 10 Walmart's are located in 3 different states: 4 in California, 3 in Wisconsin and 3 in Texas. These items are divided into 3 categories – Hobbies, Foods and Household – with 10 departments spread among them. We don't know specifically what each item is rather each item has a specific ID along with an ID for the department, category, store and state.



Below is an example of one of our items. You can see that the data can be very erratic and chaotic with large jumps up and down seemingly randomly. Almost all of our items have this kind of shape and dealing with this problem was a big part of our analysis. Many of the items in our data set have a median unit sale of 0, while some can sell as many as 600 units in one day.

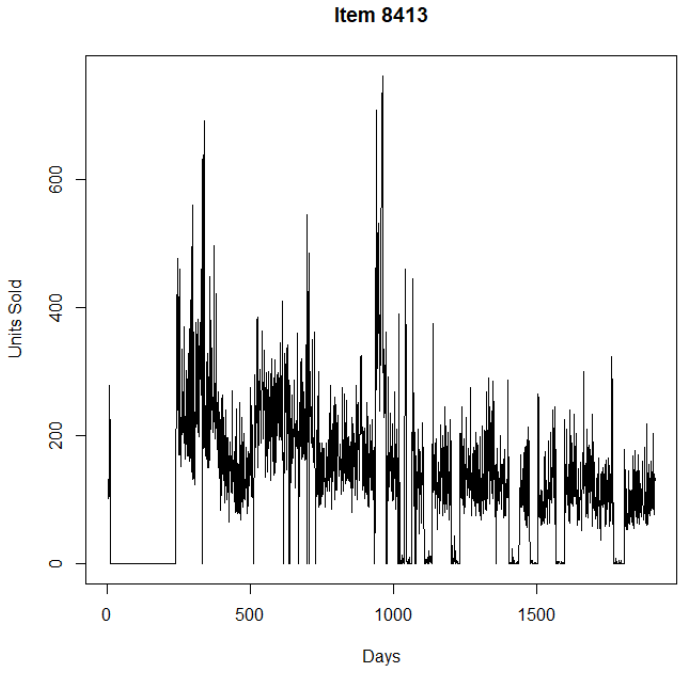


2.2 Problem

Our problem is to be able to predict what the unit sales will be for our 30490 different items. With the distribution of our data and the chaos that is apparent in it, it will be difficult to accurately predict for all of the items. Since there are so many different items with so many different trends, it will also be difficult to find a method that we can use on all of them to get accurate results.

One important problem that exists is the idea of seasonality. This means that some items will have weekly, bi-weekly, monthly, or yearly trends in their sales data. An item might have an abnormal seasonality like 5 days, 6 days, or any kind of time span you want to mention. Being able to find the seasonalities for each item will be key for us to tackle this problem effectively.

Many of the items in our data, have very unnatural trends where they dip down to selling zero items for periods of time. Below is a graph of one particular item where we can see this happening. This is probably the result of missing data, but we can't be sure. This leads to many problems and hinders our models considerably. The erratic jump between selling over 600 units and then selling 0 is more than likely unnatural.



3 Methodology

3.1 AIC

One common problem many data scientists have is finding the right model. We used aikaike information criterion (AIC) to let us judge how accurate our models were. AIC essentially estimates the quality of both models we used and provides us with the most accurate result. AIC does this very well while also dealing with the risk of overfitting and underfitting. AIC also helped us figure out how much of the dataset should be used, what the most relevant parts are and what kind of seasonality we should be aware of. Below is the AIC equation.

$$AIC = 2k - ln(\hat{L})$$

The model that has the lowest AIC value is the considered the better model. L is the goodness of fit, so having a large L value should lead to a low AIC value. However, K is the number of parameters of the model and discourages overfitting by acting as a penalty for the equation.

3.2 Neural Networks

The first method we decided to apply was a neural network. We used a feed forward neural network with lagged inputs and a single hidden layer. A feed forward neural network means that the outputs from each node do not cycle back into itself. Each node outputs to the next layer. We also used lagged inputs with means that we only use a select set of past values. The number of lags used for each item was determined by AIC. The hidden layer is the nodes that between the input and output layer of the neural network. Our hidden layer was made up nodes equal to half the number of input nodes.

We used the neural networks over the last 100 days of sales. We did this because for each item 20 neural networks were created. Using a smaller portion of the dataset only slightly increased our error but significantly decreased on calculation time. The results for each of the 20 networks were then averaged together to get the best results for each item.

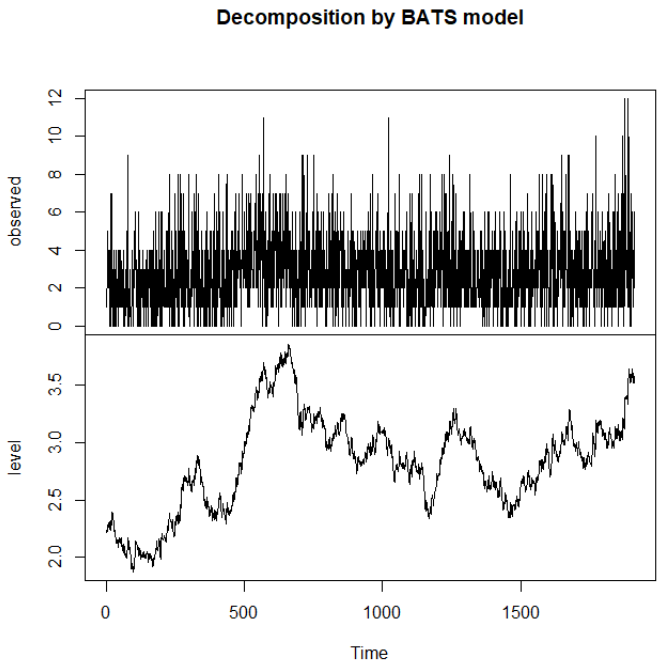
3.3 TBATS

Our next model that we tried was the TBATS model. This is the Trigonometric Box-Cox transformation with ARMA errors and trend and seasonal components. This sounds a bit complicated at the moment, but we will break the model down step by step to explain how it works, why it works, and then finally, how we applied it to our data. Let's first get into how the model works.

A Box-Cox Transformation is a way to transform non-normal variables into a normal shape. Having a normal distribution is obviously very important for many statistical methods and applying this to our data can often make our lives much simpler. It helps make certain computations faster and it helps in finding seasonalities. Below is the equation used to do a Box-Cox transformation.

$$y(\lambda) = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \lambda \neq 0 \\ log(y) & \lambda = 0 \end{cases}$$

Exponential smoothing is also used in this method. This is essentially way to help us remove outliers in our data. We don't want one exceptionally good or bad day to hinder the performance of our models. Exponential smoothing is also done to help us see the trends in the data more clearly. We do have to be sure that we are not removing too much information when we apply the exponential smoothing. Below is an example of exponential smoothing.



The ARMA model is the Autoregressive-moving-average model. The autoregressive part is given by the following equation. This equation makes any future point a function of all of the past points in the time series. ϕ_i are our parameters and the $epsilin_t$ is for the error.

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t$$

The moving average part of the model is given by the below equation. This time θ are our parameters and μ is the average. This models the average of the time series data over its duration and helps to account for some seasonalities. Again ϵ_t is our error term.

$$X_t = \mu + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

We combine these two equations to get our ARMA model as seen below. This gives us a model that can forecast based off of it previous points and can move following some seasonalities.

$$X_t = c + \epsilon_t \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

With all of this, we can finally create our BATS model which can be seen in the equations below.

$$y_t^{(\omega)} = \begin{cases} \dfrac{y_t^{\omega} - 1}{\omega}, & \omega \neq 0, \\ \log y_t, & \omega = 0, \end{cases}$$

$$y_t^{(\omega)} = \ell_{t-1} + \phi b_{t-1} + \sum_{i=1}^{T} s_{t-m_i}^{(i)} + d_t,$$

$$\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha d_t,$$

$$b_t = (1-\phi)b + \phi b_{t-1} + \beta d_t,$$

$$s_t^{(i)} = s_{t-m_i}^{(i)} + \gamma_i d_t,$$

$$d_t = \sum_{i=1}^{p} \varphi_i d_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t,$$

Now,we can look at the TBATS model. This is an upgrade of the BATS model as it includes trigonometric seasonality to the BATS model. Here, we use trig expressions based on Fourier series to help us find our seasonaloties. Using this methods, we are more likely to find odd seasonalities that the BATS method would be unable to find. The details of the model are below.

Model:

$$y_t^{(\lambda)} = l_{t-1} + \phi\, b_{t-1} + \sum_{i=1}^{T} s_{t-m_i}^{(i)} + d_t$$

$$l_t = l_{t-1} + \phi b_{t-1} + \alpha d_t$$

$$b_t = \phi\, b_{t-1} + \beta\, d_t$$

$$d_t = \sum_{i=1}^{p} \varphi_i d_{t-i} + \sum_{i=1}^{q} \theta_i e_{t-i} + e_t$$

Seasonal part:

$$s_t^{(i)} = \sum_{j=1}^{(k_i)} s_{j,t}^{(i)}$$

$$s_{j,t}^{(i)} = s_{j,t-1}^{(i)} \cos(\omega_i) + s_{j,t-1}^{*(i)} \sin(\omega_i) + \gamma_1^{(i)} d_t$$

$$s_{j,t}^{*(i)} = -s_{j,t-1}^{(i)} \sin(\omega_i) + s_{j,t-1}^{*(i)} \cos(\omega_i) + \gamma_2^{(i)} d_t$$

$$\omega_i = 2\pi j / m_i$$

Where:

$y_t^{(\lambda)}$ - time series at moment t (Box-Cox transformed)

$s_t^{(i)}$ - ith seasonal component

l_t - local level

b_t - trend with damping

d_t - ARMA(p,q) process for residuals

e_t - Gaussian white noise

Model parameters:

T - Amount of seasonalities

m_i - Length of ith seasonal period

k_i - Amount of harmonics for ith seasonal period

λ - Box-Cox transformation

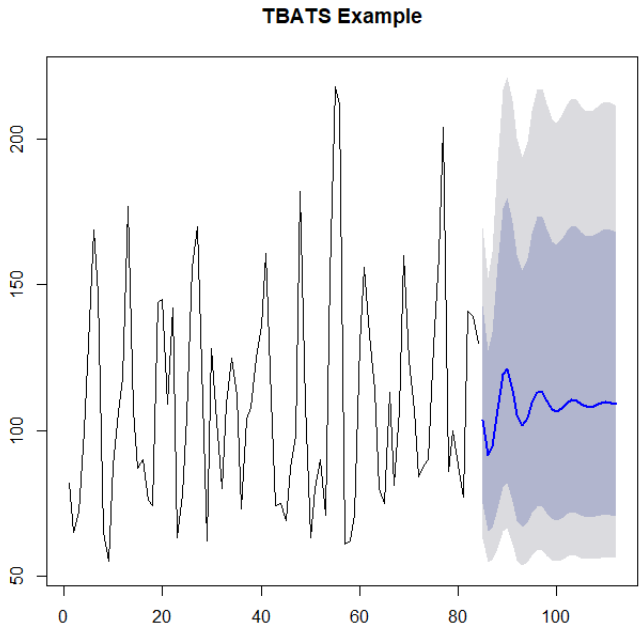
α, β - Smoothing

ϕ - Trend damping

φ_i, θ_i - ARMA(p, q) coefficients

$\gamma_1^{(i)}, \gamma_2^{(i)}$ - Seasonal smoothing (two for each period)

Now, we can finally apply our model to our data. We can see an example of its application below. While the ranges for possible values are correct, the values it predicts are not always the best. They usually hug close to the average too much. We need to be able to model the chaos of our data. We decided to rely more on the ranges for each day than the predicted values.



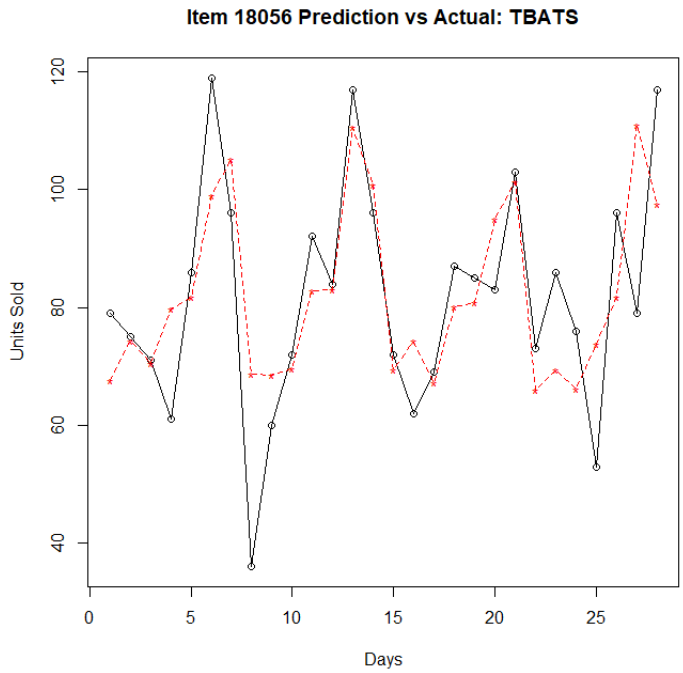
3.4 TBATS changed

We altered the TBATS method slightly to help us work with the natural chaos of our data. WE instead ran the algorithm over the last 12 weeks instead of the entire 5 years worth of data. This cuts down dramatically on computation time and leads to little or no increase in error. We then focused more on the ranges the model gives us and certain types of seasonalities.

We took a look at one week, two week, monthly and yearly seasonalities for this method. Over the 12 weeks we used for each item, we looked at the total amount of items sold on a corresponding day. If we were to look at weekly seasonality, then we looked at all the items sold on Monday, Tuesday, etc. If Tuesday had the most items sold, we then scaled our data accordingly with Tuesday occurring more towards the max of the range and other days appearing lower. IF this was a yearly seasonality, we used 5 year worth of data and looked on the exact same day of previous years.

We scale each of the seasonalities mentioned accordingly. If we are looking at weekly seasonality, then we compute the cosine similarity between each of the 12 weeks and average them. The result is the scaling factor for that seasonality. This was a method that had

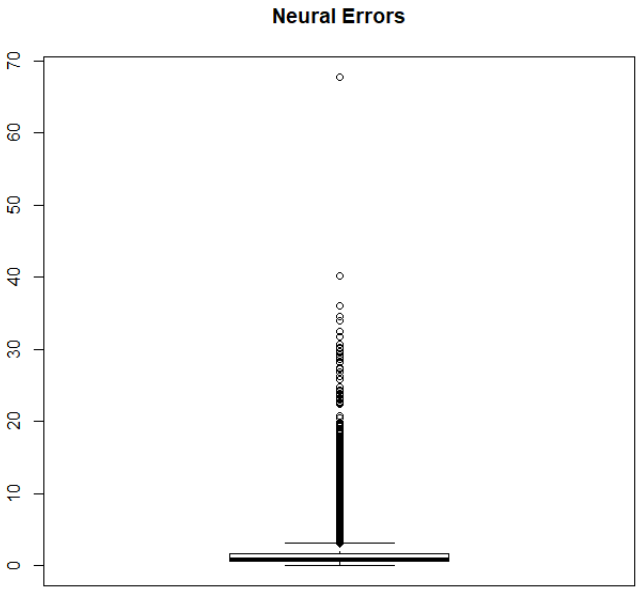
strong results as seen below.



4 Results

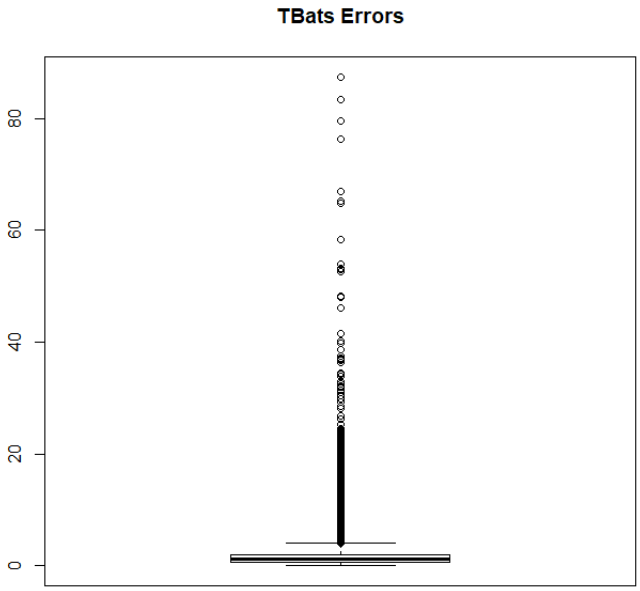
4.1 Neural Network Results

Our neural network results were quite strong. We can take a look at the distribution of errors in the boxplot below. Most of our errors are quite small, but there are still several outliers with very high error. The mean error of our neural network is 1.4604. This is calculated using root mean square error. Of the 30490 different items, the neural network performed better than the TBATS model on 23062 of the items.



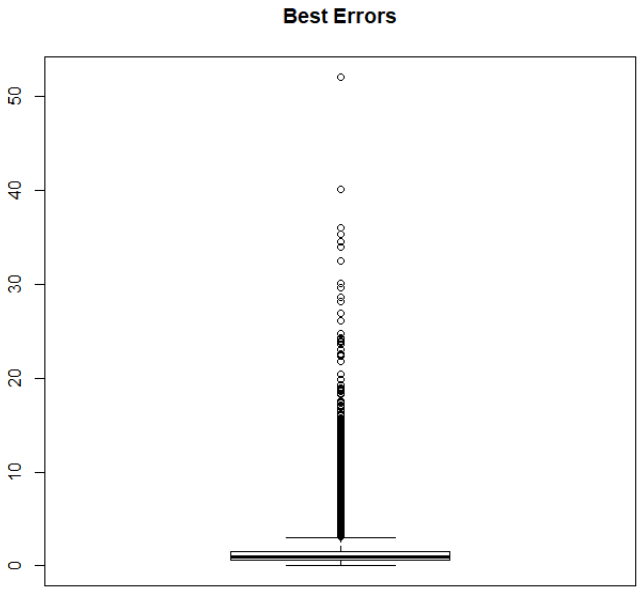
4.2 TBATS Results

Our TBATS method results were also quite good. The distribution of errors can be seen below in the following boxplot. Similar to the neural network errors, most of our errors are quite small with several outliers. The mean error is 1.5544. The TBATS method outperformed the neural network on 7428 of the items.



4.3 Overall Results

For our overall results, we can combine the best of both of our methods. We can see our best errors for each item in the boxplot below. While most of our errors are quite low, we are still left with several outliers. We have 965 different items with a root mean square error above 5. We end up with a mean error of 1.3798.



5 Feedback

Most of our feedback that we received was based on our dataset that we were given by Kaggle. The data that was provided wasn't specific in a lot of aspects including what each product specifically is and where each store is located in its respective state. We agreed that this kind of information could have been helpful because each store could have more or less foot traffic based on its location and because something like an apple is going to have different sales seasonality than something like a canoe. Unfortunately, this is information that we do not have access to.

We also received feedback that we should consider seasonality in regards to specific months. This is because purchases in January can be very different from August. We actually addressed this after the first WIP and applied weights to specific months and even days of the week.

6 Plan

Our plan for this project was quite simple. We both had very little experience dealing with time series analysis data so we spent the first few weeks getting familiar with the distribution of our data and researching different methods to analyze data like this. After finding several models that we believed would work, we tested them to find the best. We worked through the data slowly to understand how things were working and what was working. We took time to analyze our results leading to us creating new ideas and ways to tackle this problem.

7 Conclusion/Future Plans

In total, we believe we have done a good job of analyzing our data and predicting the next 28 days for the items. We have learned a lot about time series analysis data and forecasting. This project was a good deep dive for us into new methods we have never used before. Future plans for this problem are interesting. Trying to reduce the large errors on our few outliers is very important. Finding an effective way to remove the possible "missing" data would also help in making the models better. If we were able to reduce that part of the chaos, then are models would undoubtedly be much better for it. Trying to understand what make one method work over another method is an interesting problem and would certainly lead to a better understanding of the data and stronger models. We attempted to find a way to cluster our items, but were unable to find an effective way to do this. Clustering the items along certain parameters would lead to better analysis.