Capstone 2: Inventory Demand Forecasting

By: Kristen Colley

The Problem:

A local food distribution company buys produce from local farms and then distributes to wholesalers, upscale restaurants, grocery stores, and individuals. This company habitually over and under stocks their goods because they do not have an inventory demand forecast model. Buying decisions are mainly made by gut or innate industry knowledge. There is currently no systematic, data-driven stocking approach in place.

The aim of this study will be to create a useful predictive model to forecast the demands for each individual item's PLU number. This represents a huge potential savings for the customer.

Typically, forecasts for grocery items would aim to improve over or understocking issues. However, this particular dataset does not include information about stocked quantities (overstocked), and the data about orders that could not be fulfilled (understocked) may not be reliable.

Thus, the focus will be to create a useful predictive model for each individual item SKU, but once this is accomplished, we can then move onto forecasting the "understocked" quantities specifically.

The Client & Audience:

The most important stakeholder in this problem is the grocery distributor. Being able to help this customer predict their weekly order quantities to some degree of accuracy may result in huge savings to their bottom line. My employer is also a large stakeholder

in this problem. If my company can deliver a good predictive model, this would set a precedence for more data driven decisions from the customer.

Data:

Currently, the invoice data is updated daily on their SQL database living on an AWS server. I started with a CSV file of two and a half years of historical purchasing data for all of the items. For some of the more complicated machine learning methods, adding features will be needed.

I will convert the SQL database to a Pandas Dataframe to apply different forecasting techniques.

The data is fairly clean, although there are some integrity issues that should be considered for a more accurate predictive model if it was to go into production.

Models To Try:

Inventory demand forecasting or optimization has been around for a long time; However, it has been cited as "*one of the hardest problems in machine learning*" to forecast FMCG (fast-moving-consumer-goods). In recent years, there have been some breakthroughs in machine learning techniques that have vastly optimized and increased prediction accuracy in the FMCG space. From very simple models (moving average) to very complicated and computationally expensive models (LSTM, long-term short-term memory deep learning), there is a wide array of approaches to take to solve this problem.

Classical time series and regression techniques normally consider either a single or a few variables such as trend, seasonality and cycle. Whereas, machine learning-based techniques are able to process an unlimited number of predictor variables, determining

the ones that are significant. However, it is very important to exhaust all the different classical methods before moving onto more complicated machine learning models.

Below are the different models I will try:

1. **Naive Forecast**

   A Naive forecast is simply taking the previous periods value for the current periods prediction. This the best a random walk time series can do, and is the most basic/primitive models you can use, and we will use this as our baseline.

   $$\hat{y}_{T+h|T} = y_T$$

2. **Simple Moving Average**

   This will also serve as a great baseline measure to compare the other models against. A moving average simply makes you name a time period (say 7 days), take the average of the values in that time period, and use that predict tomorrow's value. It is up to you to grid search the optimal past period.

   $$\bar{p}_{\mathrm{SM}} = \frac{p_M + p_{M-1} + \cdots + p_{M-(n-1)}}{n}$$

3. **Simple Exponential Smoothing**

   Gives higher weights to recent data, and lower weights to older data and then takes the average of the values with the weights taken into account.

   $$s_t = \alpha x_t + (1-\alpha)s_{t-1} = s_{t-1} + \alpha(x_t - s_{t-1})$$

4. **Holts Linear Trend**

Holts Linear Trend will do the same as the exponential smoothing, but also take into account the trend of the data

$$\begin{aligned}
&\text{Forecast equation} && \hat{y}_{t+h|t} = \ell_t + hb_t \\
&\text{Level equation} && \ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
&\text{Trend equation} && b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1},
\end{aligned}$$

5. **Holts Winter Model**

Holts Winter Method does triple exponential smoothing to try to uncover seasonality and trend. You must gridsearch the correct seasonality parameter.

$$\begin{aligned}
\hat{y}_{t+h|t} &= \ell_t + hb_t + s_{t-m+h_m^+} \\
\ell_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \\
s_t &= \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m},
\end{aligned}$$

6. **Facebook Prophet Model**

Prophet is an additive regression model that is particularly good at data with strong seasonal and linear trends, and also generating forecasts over large amounts of items. You can add in models, and daily/weekly/monthly/quarterly seasonality which might be helpful for grocery items. It also is a LOT quicker than an ARIMA model, and would be a better choice for implementation.

7. **Random Forest Regression**

Random Forest Regression has been known to work well in this space. Trees separate records into more homogeneous subgroups in terms of the outcome variable by creating splits on predictors, thereby creating prediction or classification rules. These splits create logical rules that are transparent and

easily understandable. Trees can be a not so great solution for forecasting because they can't easily forecast trends and seasonality. To combat this I will take the log of the data before running it through the model, and then convert it back to try to predict on a more stationary data set.

8. **Light Gradient Boost Model**

   Another tree-based algorithm, the LGB model is taking Kaggle competitions by storm. It has surprisingly accurate results for a wide variety of problems as well as incredibly fast run times (thus the "light" designation). This model, unlike other boosting models, splits leaf-wise instead of level-wise (tree depth). This has the effects of reducing the loss of each tree, and making it very fast with computation times.

9. **NEURAL NETWORK (LSTM-RNN, CNN)**

   Some of the most cutting edge research done with inventory optimization, is being done with Neural Networks. If the data is large enough I will try this method, but if it is less than 60,000 entries I was advised to move on from this method.

Data Cleaning:

Initially, the data looked surprisingly clean. There wasn't a ton of good information with the other features such as delivery route, promotion code, etc which had too many missing values, but the most important part, the quantities and revenue, looked good.

Each of the 3000 PLU codes did have a description column which was a large string with information about the product size and description of the item itself.

It should be noted that this information is entered on an excel spreadsheet, and there is definitely room for errors with the values themselves.

Steps to clean:

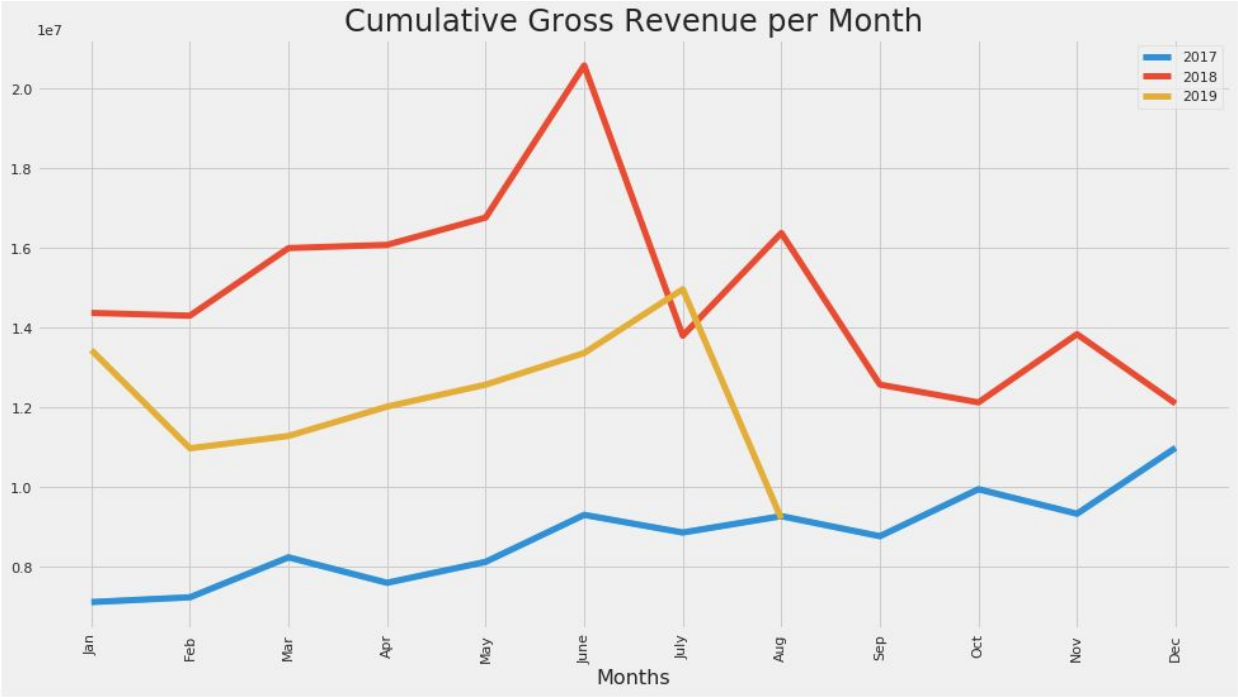
- Querying and joining the data from two tables
- Correcting data types (especially date-time)
    - Filling in a couple missing days
    - Aggregating the data from the past correctly (1 year, 2 year, 2.5 years)
- Getting rid of negative orders (refunds will not be a part of this analysis)
- Taking out any delivery codes from the item list
- Creating columns
    - Normed quantity (total quantity ordered taking in the normalized quantity into account)
    - Total qty (total normed quantity per item per period)
    - Total rev(total revenue per item per period)
    - Label column by slicing off the first word of the description
- There are 528,688 logged purchases for 2.5 years of history with 67,366 unique invoice numbers
- There are 3088 unique item numbers (PLUs)
    - There are 4096 descriptions for those unique item numbers which means some descriptions were modified over the years to correspond to the same item number

EDA:

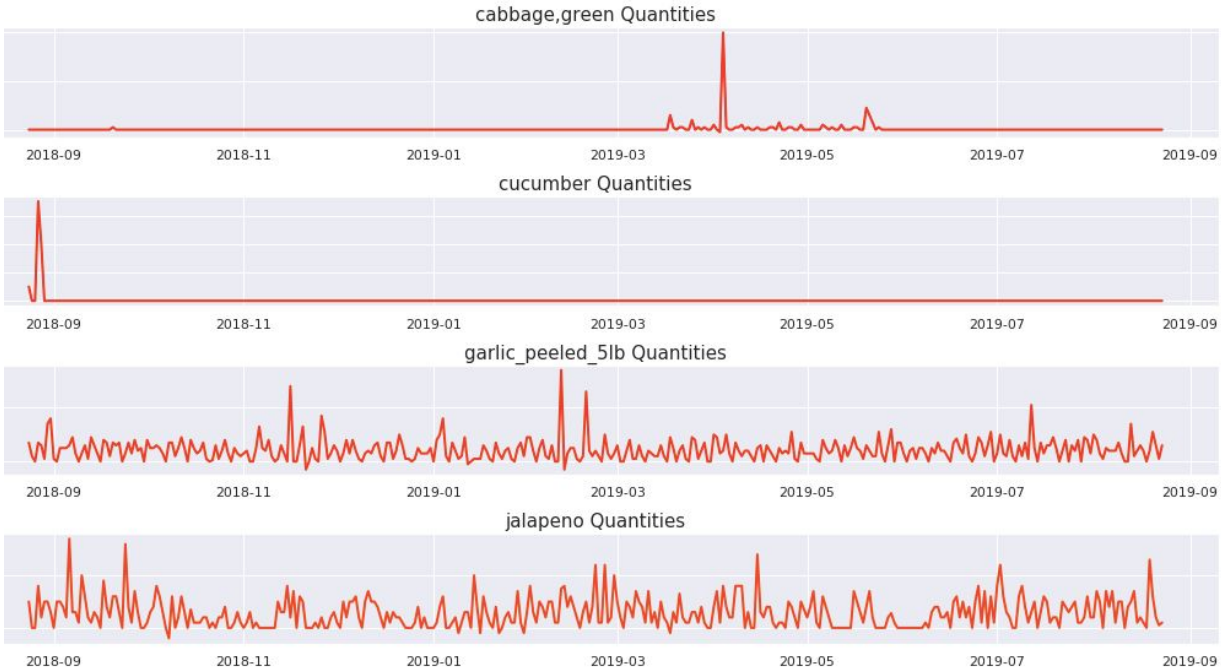
The initial EDA was discovering that every item had a different trend, and seasonality. Even when I grouped the products into their broader 9 categories (fruit, veg, potatoe, tomato, greens, meat, grains, mushrooms, dairy), each product wasn't super representative of the trend as a while.

EDA GRAPHS:

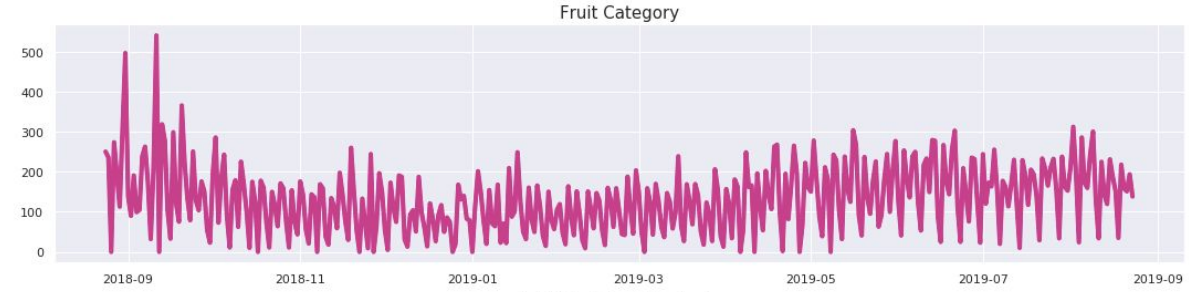
What does the yearly distributions look like?

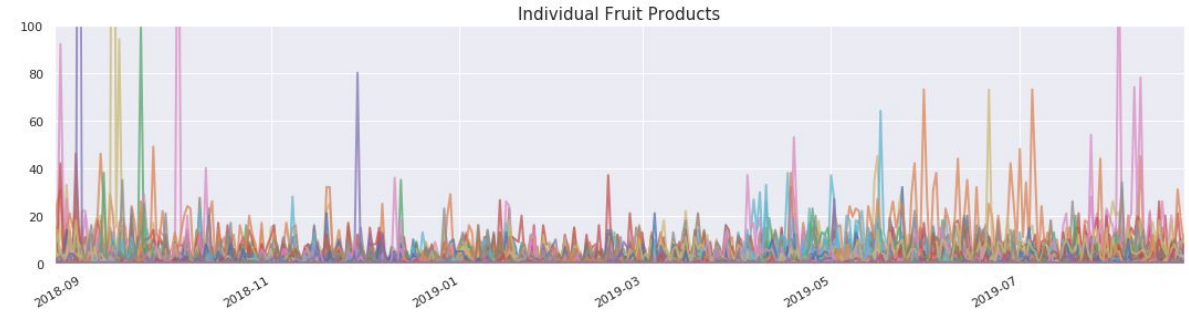


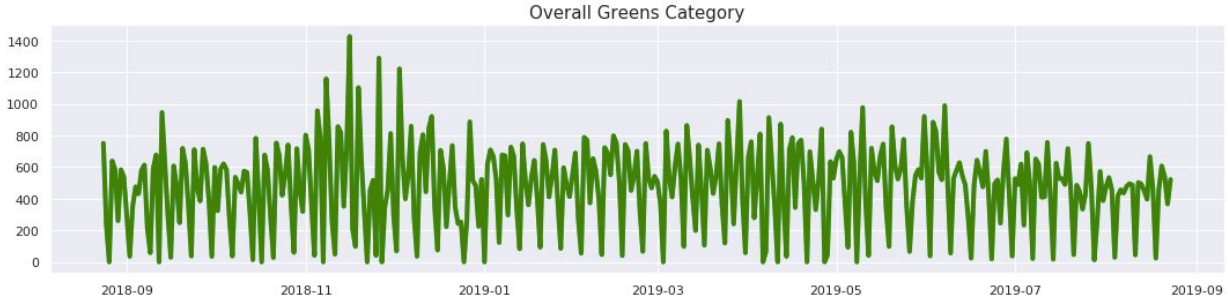
Individual Item distributions:

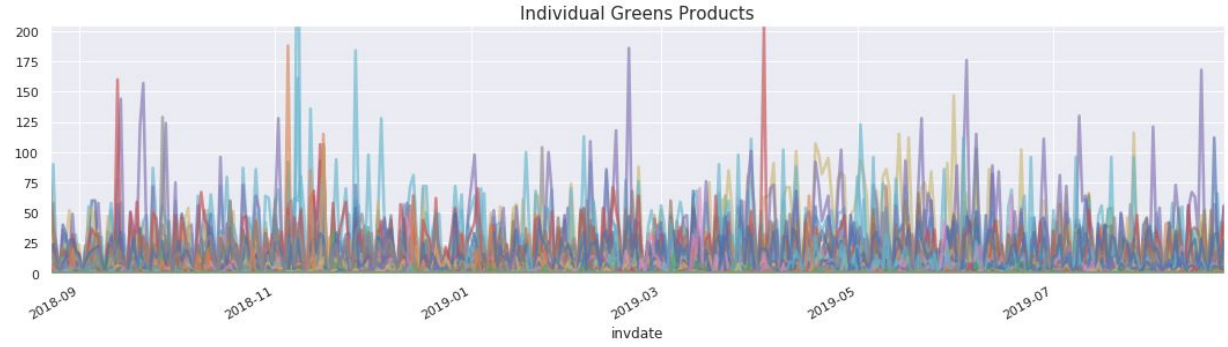


How do individual items look compared to their broader categories?

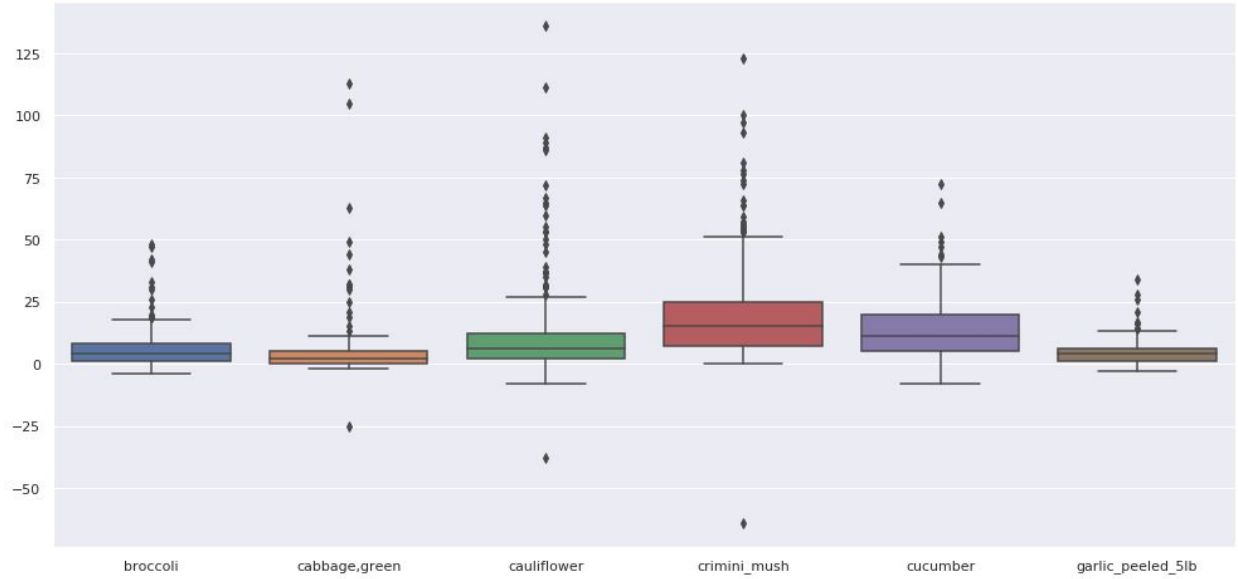








How do the outliers look like on the individual level?



Random Walk Testing:

Random Walks typically happen in financial data a lot. It basically means the data is taking successive steps that are random (think a literal "random walk"). The steps are successive, but not predictable. Within each period the variable takes a random step away from its previous value, and the steps are independently and identically distributed in size (i.i.d.). This concept was popularized in 1973 with the publishing of "A Random Walk Down Wallstreet" book link.

The best prediction for a random walk is to look at the previous "step" or timeframe and use that to predict the future value. This is called a "Naive" or baseline forecast.If we have a random walk process, we will not be able to predict future changes better than a naive model.

In order to determine if I had any random walks, I used five points to test:

1. Augmented Dickey-Fuller hypothesis test. With this test, the null hypothesis is that the time series IS a non-stationary series. Thus, we want all the p-values to be below .05 which means we have a 5% or less chance to see a value this extreme or greater, and we can reject the null hypothesis that they are non-stationary
2. Check the Hurst Exponent (around .5 means your likely to have a random walk)
3. Check correlogram plots of original data. Is there a correlation to past time frames?
4. Check correlogram plots for the converted stationary data. Is there a correlation to past time frames with stationary data?
5. Does the Naive model outperform more complicated models? If so, then we definitely have a random walk!

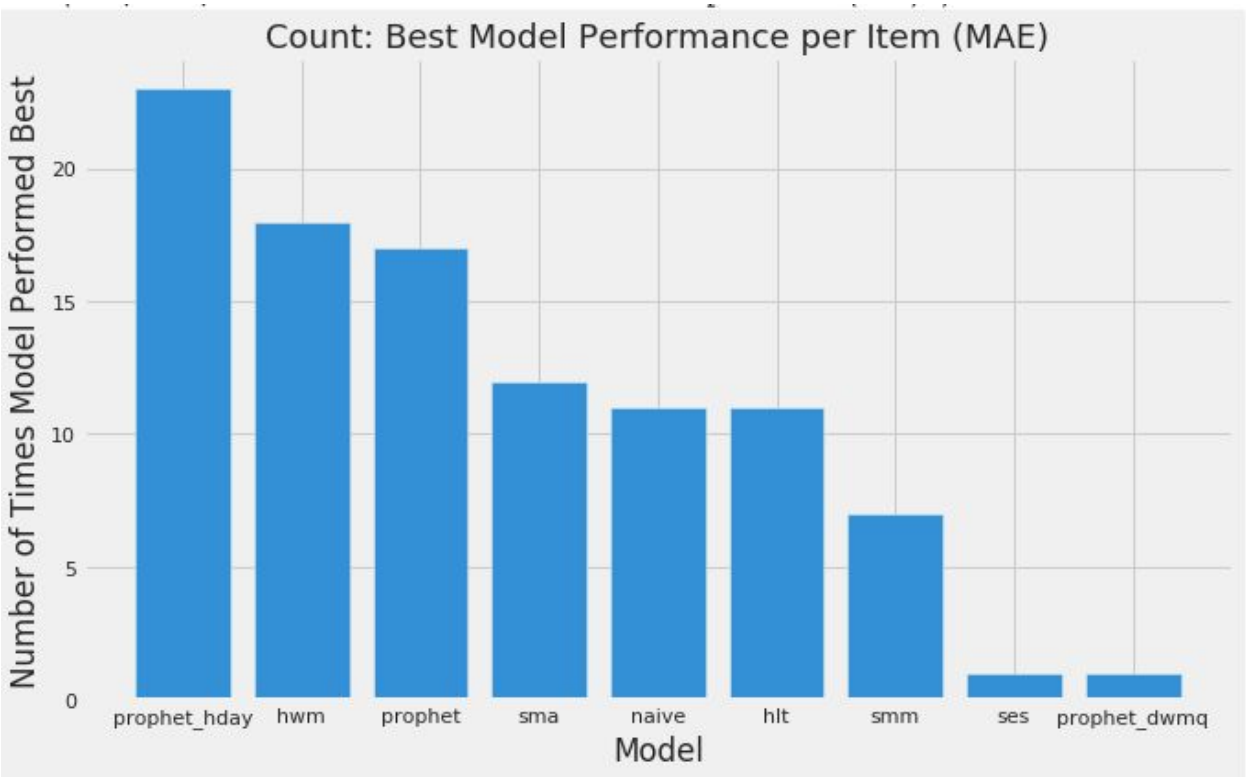
Initially, I tested 2.5 years of data for weekly aggregated quantities, and this returned ALL RANDOM WALK PROCESSES. However, I went back and tried all of the below variables with all their variations:

- Daily, weekly, bi-monthly, monthly aggregated data
- 2.5 years, 2 years, 1.5 years, 1 year of past historical data
- 9 Broader categories
- 100 highest revenue items

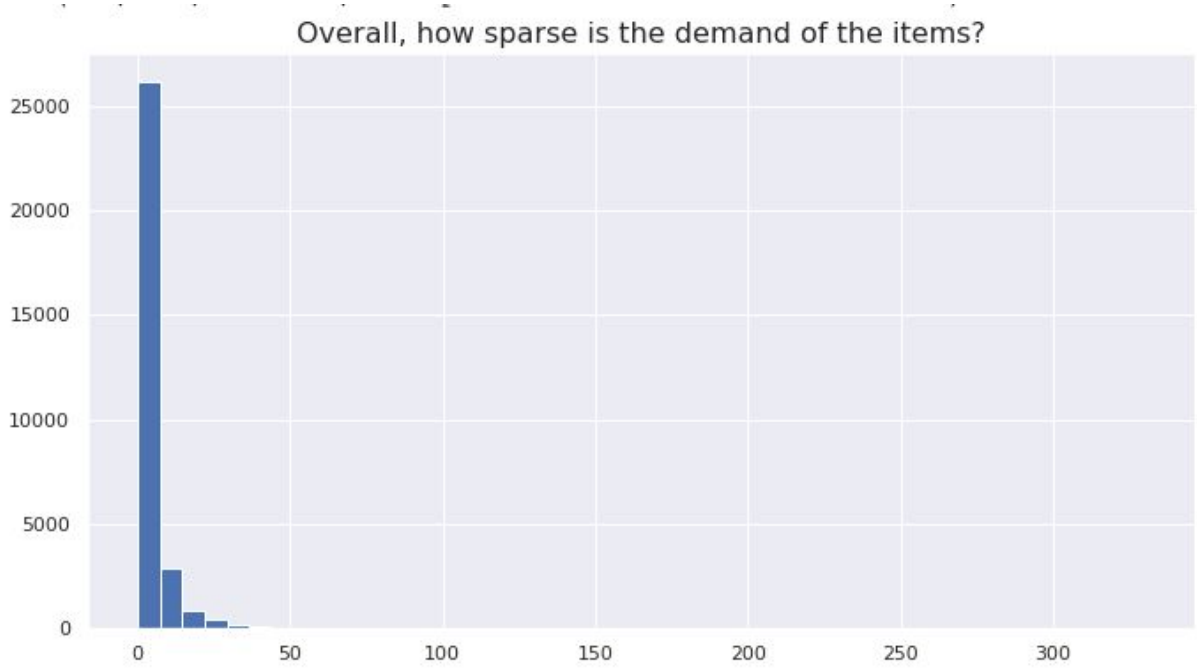
Then I was able to find a winner: **daily data with one year of past history**. Later, I found out that the company lost their biggest client 1.5 years ago which was completely skewing the trend and seasonality of past years. (There was no record of how much this client accounted for so I was unable to experiment with taking this out).

Models:

Tested all the classical models plus Facebook's prophet first on the 100 highest revenue items in the past year, and got the following results:



The prophet model with holidays added was the best performing model (24% of the time), with the holt's winter method (18%) and the out of the box prophet model (17%) coming in a close 2nd and 3rd. Overall the models were increasing performance anywhere from 43%-50%. However, the naive model out performed all the other models 11% of the time. This was not very good, and meant that we were still facing the problem of random walk process' even with the 100 most demanded items! The demand distribution for each of the PLU codes is incredibly sparse and the models above were not doing so well with it.

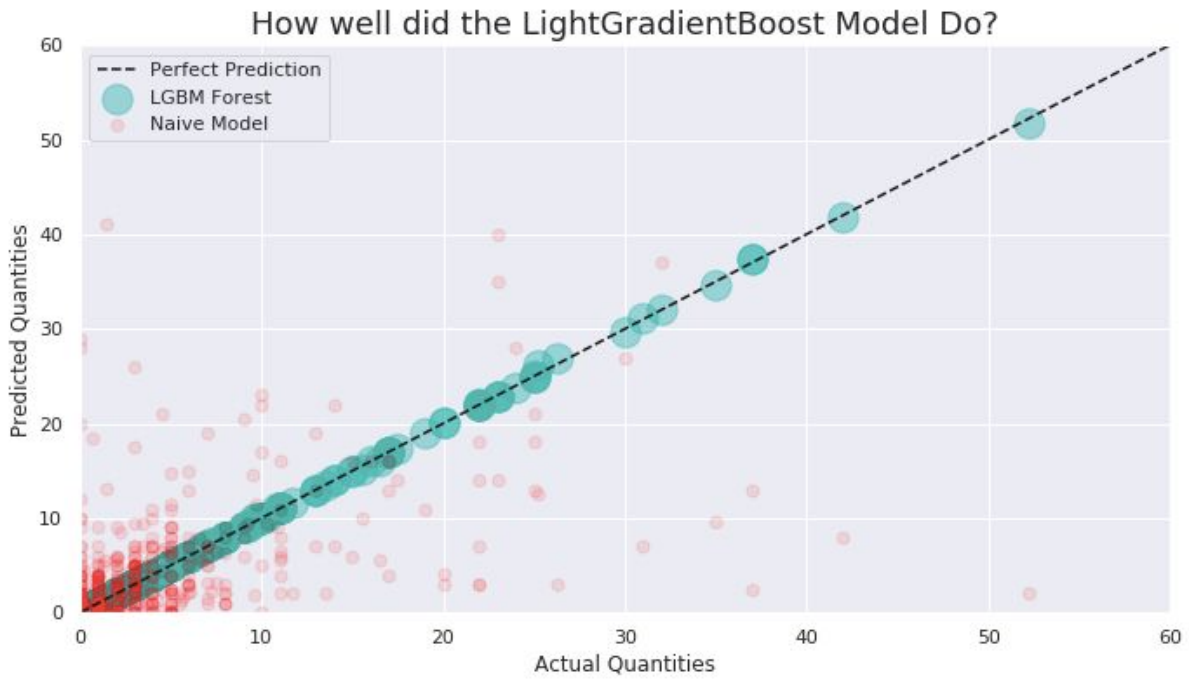


Next, I moved to see how the trees performed. Unlike the classical models, I'm able to aggregate the entire dataset together to feed into the RF model, and thus my thought was that it could be the solution to my very sparse dataset.

On the 100 most demanded items, the Random Forest Regression and the Light Gradient Boost model were able to improve upon the current naive model by 95%-97%! Thus, I moved into hyper tuning and testing the two tree models more in depth.

Light Gradient Boost Model:

Although, my favored algorithm was the LGB model due to its speed and accuracy, it was completely overfitting:



You can see in the graph above, that it was fitting entirely too well, and was not predicting very good. I tried hyper tuning a couple different parameters, but was unsuccessful at fixing this issue. After doing some research, I read that this was a common problem for LGB models, and they don't do well with small data sets like mine.

WINNER = Random Forest Regression:

Since Light Gradient Boost was overfitting, my algorithm of choice was the Random Forest Regression which still had high accuracy rates, but was not suffering from overfitting. It was slower than the LGB model, but not too slow to work into production.

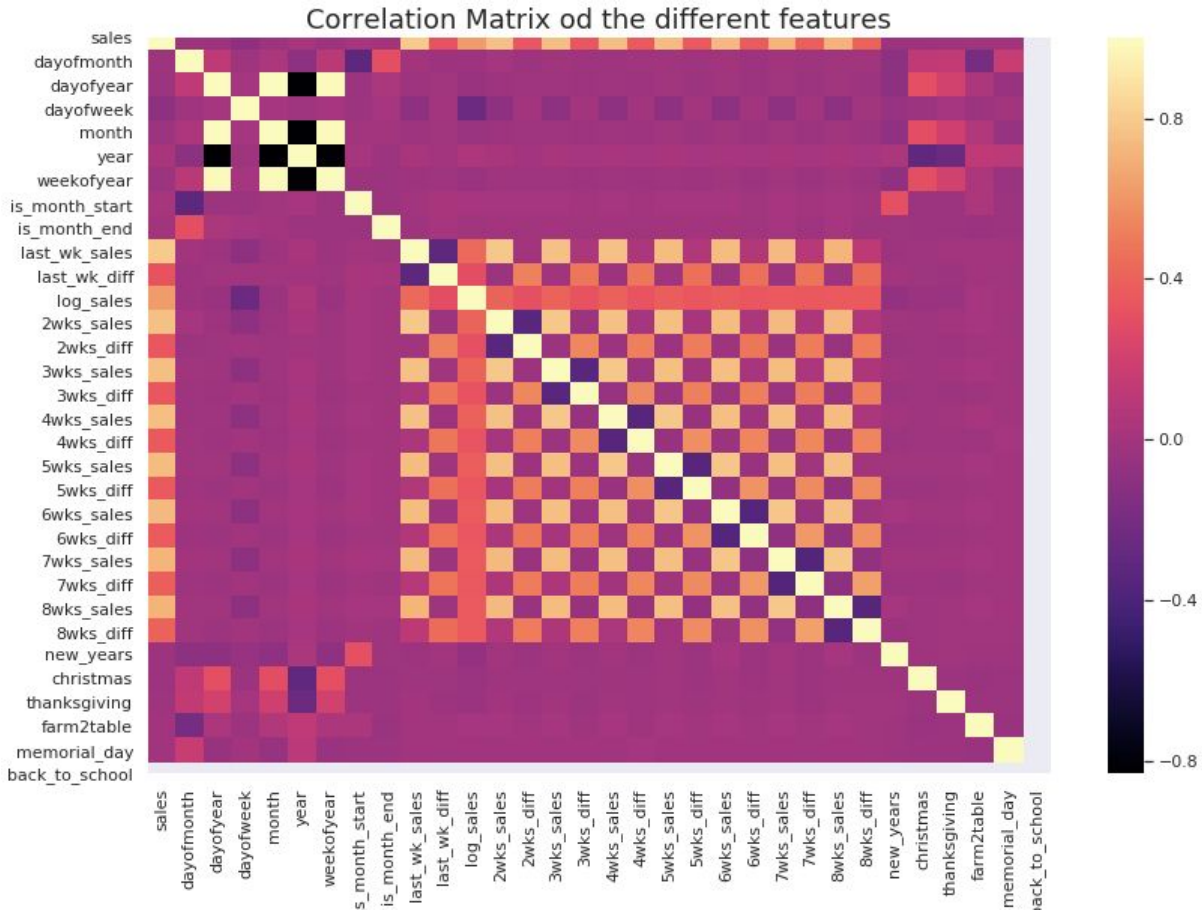
When testing ALL the items (not just the top 100 ordered) the RFR beats the customers current forecast model by 60%-88% (depending on the week).

I did account for the problem with using a Random Forest Regressor to forecast future trends by taking the log of the data first, and then transforming it back after

the prediction which should help transform future data into a more stationary trend.

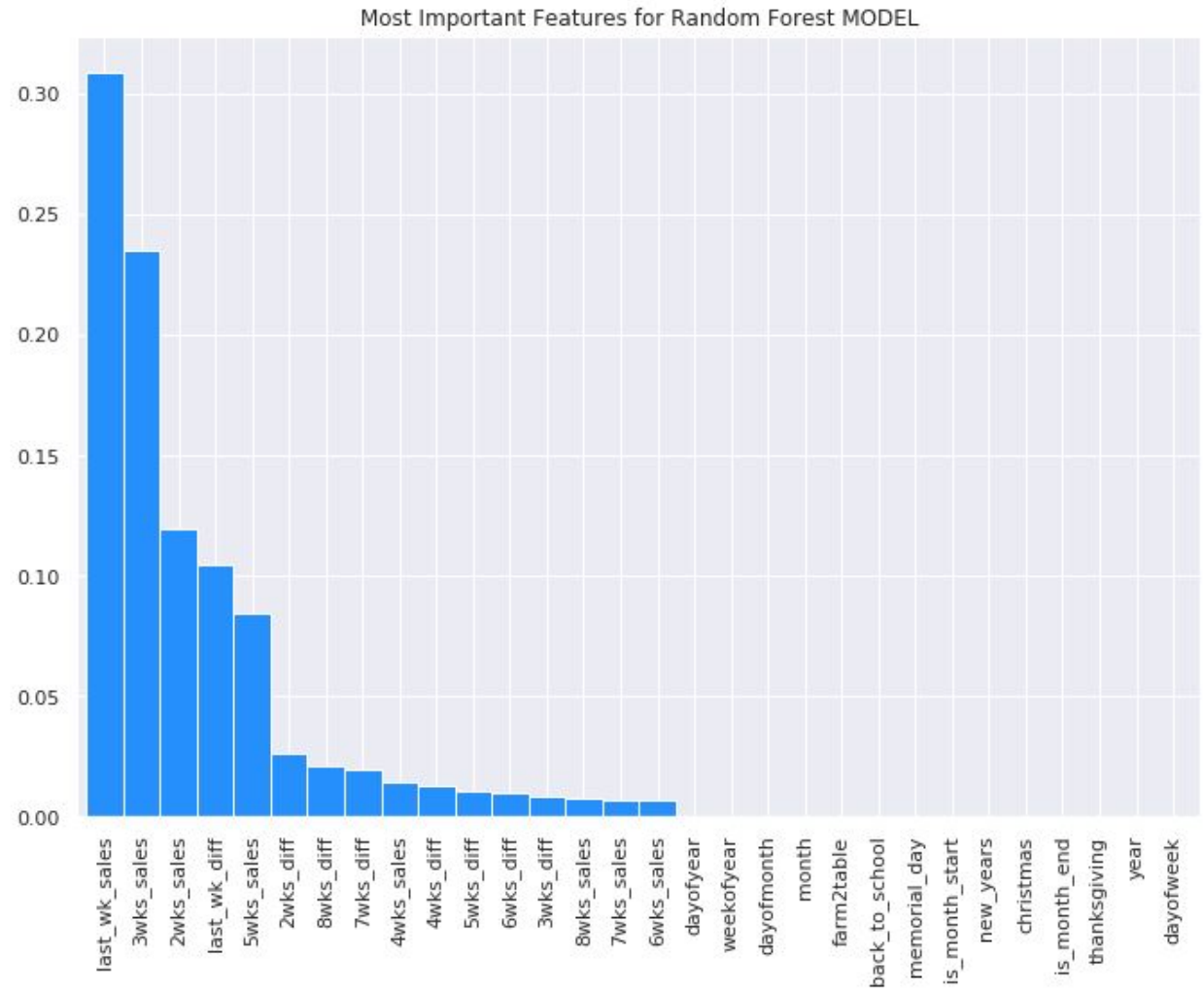
RF Feature Engineering:

I created a bunch of features that were lags or differences of past orders. I also worked in all the holiday time frames the customer and data indicated. You can see with the correlation heat map below that the highest correlations were to past sales data. The holiday or time categories weren't giving me much traction.



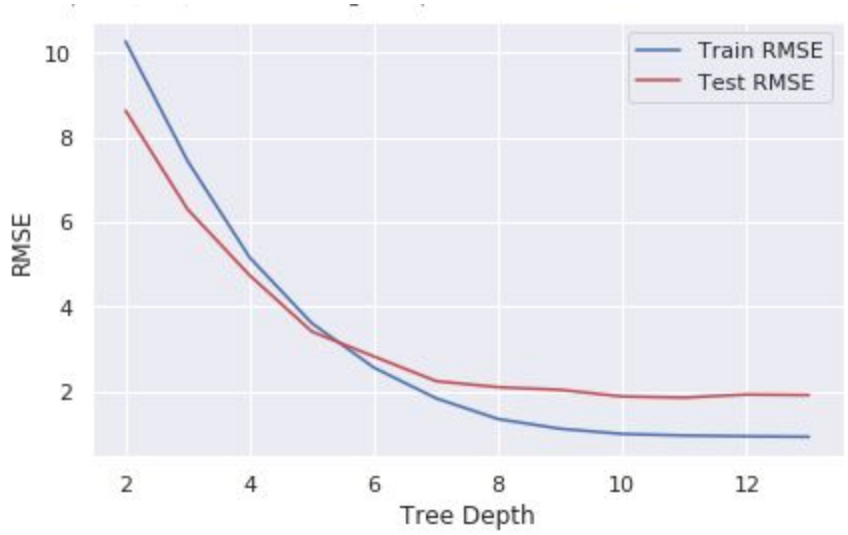
The most important features the RF model are listed below with the previous week

of sales being the biggest indicator with 2-3 weeks of past sales following 2nd and 3rd. This makes sense, and proves what the customer always intuitively knew that the best data was in the most recent history.

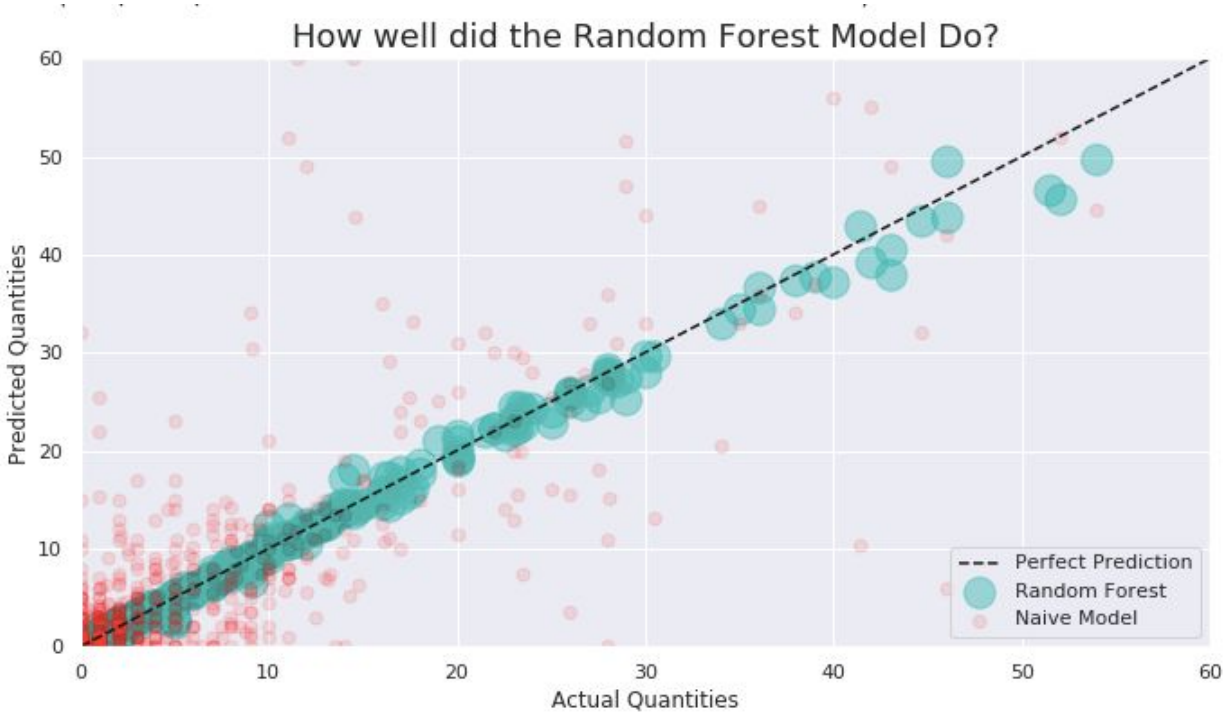


RF Parameters:

There are only two important parameters to hypertune (max trees and max depth of trees). Below you can see the trade off with underfitting and overfitting depending on the tree depth:



Tree depth of 8 was the magic number where the error levelled out with max trees at 5000.



Conclusion:

Overall, this project was successful at beating the customers current forecast model. In the end the Random Forest Regressor performs anywhere from 60-88% better than the naive model which is very good for this type of problem. Unfortunately, forecasting problems don’t see the accuracy scores that classification can.