In order to get some marketing insights/action, we model sales as a function of different characteristics of the product. The model can be used to predict sales as well as analyze the effect of different economic forces on the sales by perturbing the model.

The dataset consists of :

1. Daily sales data of 10,000 products
2. A mapping of the different products to their parent brands
3. Daily out-of-stock signal of 10,00 product
4. Marketing campaign start dates and binary features for products campaigns
5. Marketing campaign start dates and binary features for 39 brands campaigns

Dealing with the missing values:

The “product sales” data gives us information about the sale each product had every day from 1st March 2012 to 2nd February 2016. The data has some missing sales entries for certain days. The minimum missing entries for a product are 5 and the maximum is 28. So, the percentage of missing entries for the products ranges from 0.35% to 1.95 %.

Since there is no obvious pattern to be seen with the missing data, the data is missing at random. Noticing that the data is non-stationary - mean of the products changes with time, traditional method of dropping the missing values or substituting with the global mean or median wont work well while building a model.
Hence I first choose to substitute missing data with zero but it didn't perform well. Later I chose a specific method where the missing values in the product sales data was substituted by the local mean of the sales in the last 10 days and there was a significant increase in the performance after implementing this method.

Converting Time Series Prediction into a Supervised Learning Problem

To convert this time series prediction problem into a supervised learning problem, we have to construct (x,y) pairs from the data. To predict sales at a given time t, we only consider data from t-T to t. We assume that, something which happens far in the past (more than T days away) does not contribute much to what we are trying to predict. This is a reasonable assumption to make. Given this way of modelling (x,y) pair, now we have to decide on how frequently we're going to create (x,y) pair - are we going to make one for each day in the training data or skip some days (stride). For train data, I extract with a stride of 15 but for test we extract with stride of 1.

Test Train Split

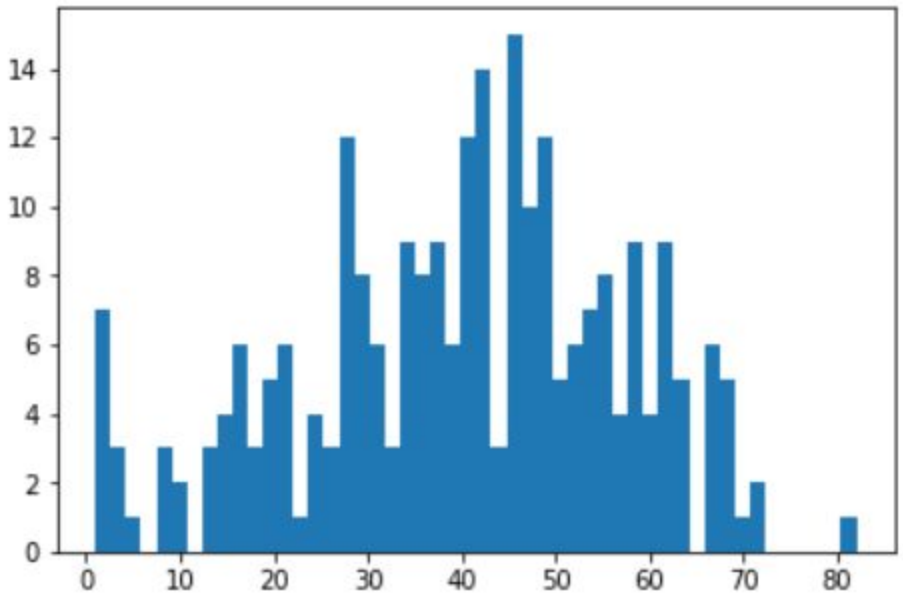
After creating the X matrix it's understood that the sales for the later time periods appear in the later rows of X.

The test data must be the the more recent sales to help us have a better accuracy to predict the future as choosing data towards the end of the time period under consideration would help in emulting the predictions better. Hence, I chose the last 20% rows of both X and Y to be my X_test matrix and Y_test vector. The rest 80% would be used as the training data.

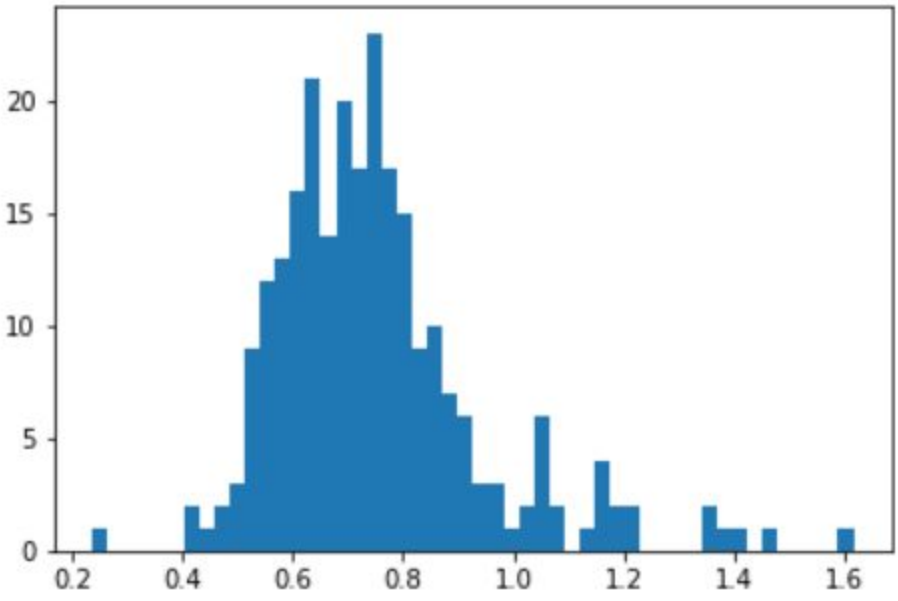
Leveraging Inter product dependencies for effective modelling

Building one model/function per product is not very efficient considering that we don't have a lot of data for one product and that we're leaving out a lot of data without leveraging them - like the dependencies between sales pattern of product. On the other hand, if you build just one model to capture the sales pattern of all the products in the inventory, the model is going to very biased and not give you good performance.

The way out is to model products which show similar sales pattern with the same function. To find which products show similar sales pattern we do the following - we build one linear regression model per product on the (x,y) pairs constructed just out of sales data. Each sales time series was normalized by using the global mean and standard deviation to make sure that the feature vectors were of the same scale and hence the weight vector of linear regression will also be in the same scale. Once weight vectors corresponding to all the products are obtained, they're clustered together using k-means to find which products naturally occur together.
Here I chose k=250 as I noticed that almost all brands have around 250 products associated with it and hence k=250 should give better results.
The MSE of the test of the randomforest is 0.7380 whereas the train value is 0.059829
The number of product in each cluster is given by the histogram :



The MSE of test errors is



Predicting the Campaign End Dates:

None of the campaigns have end date information given to them but we have information on the start date and the daily sales each product had from 2012 to 2016. On some preliminary analysis, I note that the sales of product peak on the day the campaign start and continue for some time after which it eventually drops. I took this observation as the backbone for building the prediction model. Assuming that the sales of the product drops on the day after the

campaign ended, I find out the mean of the sales the a product had in one year between which the campaign was launched. Comparing the daily sales with the mean of the sales, we can get the end date of the campaign if the sales a particular day was below the calculated mean.
Similar approach can be used to get the end date for brand campaigns too.

Possible approaches to do in future:

Further improvement of the model can be done by considering the following assumptions:

1. Predicting the sales by including the features of the campaigns.
   We can add the campaign features to the input X matrix to have a better performing model.
   - The first dimension could be the number of campaigns that was active for a given period for the product
   - The second dimension would be number of days from the start of the most recent promotion to prediction day
   - The third addition to X would be the feature vector of the most recent promotion

   So, an X matrix with these information along with the last 15 days OOS data and last 15 days sales might help us build a better model.

2. Giving more weights for training data points towards the end.

This can be achieved by having different frequency of data points in spectrum. So, choosing a smaller S for points towards the end we have more data points when compared to a larger S in the beginning of time.
This would be good approach as the data we are dealing with is non stationary.