Online Retail Forecasting using Facebook Prophet

Author: Hussain Abbas, MSc

I leveraged the Facebook Prophet forecasting library since it is designed to overcome issues encountered in the wild such as non-stationary data, sparse data, irregularly spaced data, missing data, outliers, etc.

Prophet automatically detects changes in trends by selecting changepoints from the data. Prophet leverages a Bayesian Generalized Additive Model Approach which uses Stan as the backend. Prophet is designed for the business user, as it provides first-class end-user flexibility to adjust forecasts.

Many time series such as Australia (see below) result in sparse, irregularly spaced order quantities. Facebook Prophet can handle this easily whereas methods such as ARIMA cannot. This is a major advantage that Prophet has over libraries leveraging traditional time series methods such as ARIMA.

```
df[(df['country'] == 'Australia') & (df['stock_code'] == '23084')]
```

	invoice_no	stock_code	desc	quantity	invoice_date	unit_price	customer_id	country
187252	552956	23084	RABBIT NIGHT LIGHT	12	2011-05-12 12:34:00	2.08	12431	Australia
228193	556917	23084	RABBIT NIGHT LIGHT	240	2011-06-15 13:37:00	1.79	12415	Australia
436426	574138	23084	RABBIT NIGHT LIGHT	96	2011-11-03 11:26:00	1.79	12415	Australia
439360	574469	23084	RABBIT NIGHT LIGHT	336	2011-11-04 11:55:00	1.79	12431	Australia
468139	576394	23084	RABBIT NIGHT LIGHT	960	2011-11-15 10:32:00	1.79	12415	Australia
469139	576586	23084	RABBIT NIGHT LIGHT	240	2011-11-15 14:22:00	1.79	12415	Australia

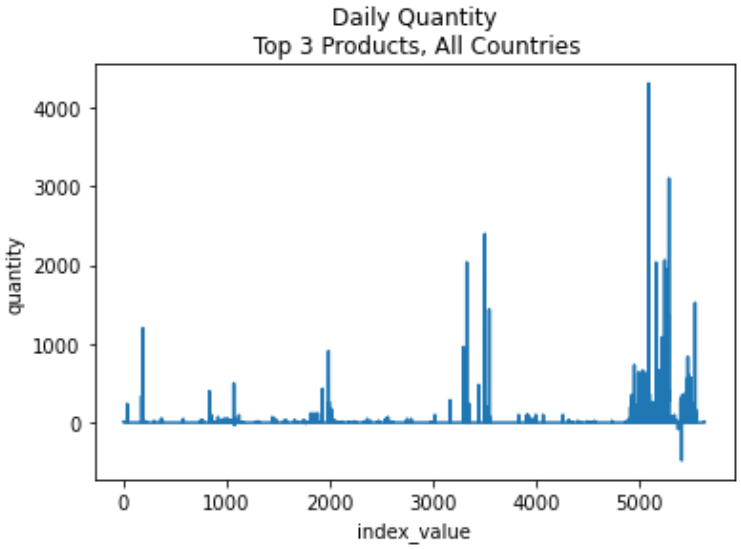
```
#most of the orders come from the UK

train_data_daily.country.value_counts()
```

```
United Kingdom     633
France             581
EIRE               553
Germany            549
Spain              474
Norway             404
Italy              379
Singapore          263
Netherlands        212
Channel Islands    203
Belgium            189
Australia          188
```

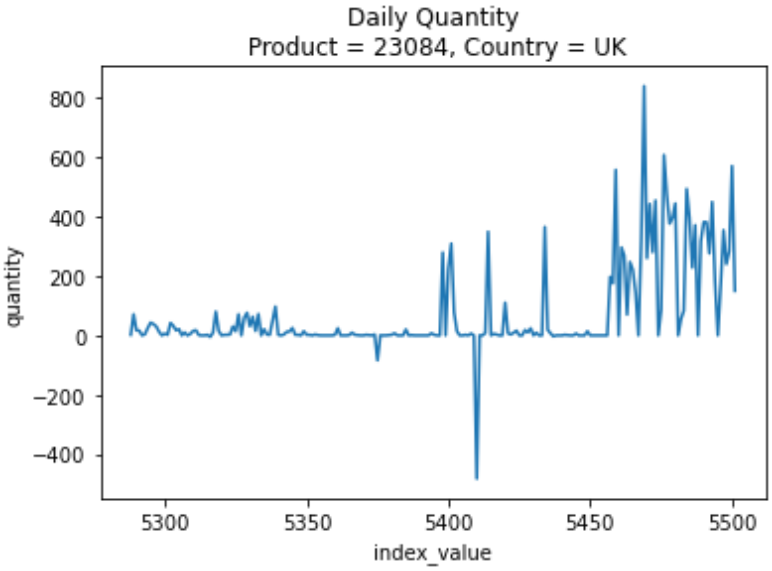
The below chart shows periods of high demand (large quantity ordered on a given day) interspersed with periods of low demand. The spikes could be due to holidays, events, marketing campaigns such as sales and flyers, businesses purchasing in bulk to reduce costs, etc.

In addition, we observe some negative quantities, which may be indicative of product returns. Large negative returns may be indicative of a batch of product defects. The top 3 products are defined as the stock codes with the largest quantities for the period 11/27/2011 - 12/3/2011).

Going forward to make it more realistic, we could shift the time period of interest to the week prior to the week for which we are generating the forecast.



The below chart narrows the scope to only the top product in the country with the most orders, the UK. We see more spikes towards the end of the time series. Diving into the data, we see that these dates occur in December. This is most likely due to holidays. Although Thanksgiving is a US holiday, global companies may experience an uplift from the increase in demand during this period. It would be interesting to investigate further the effect of US holidays on firms in other countries and vice versa.



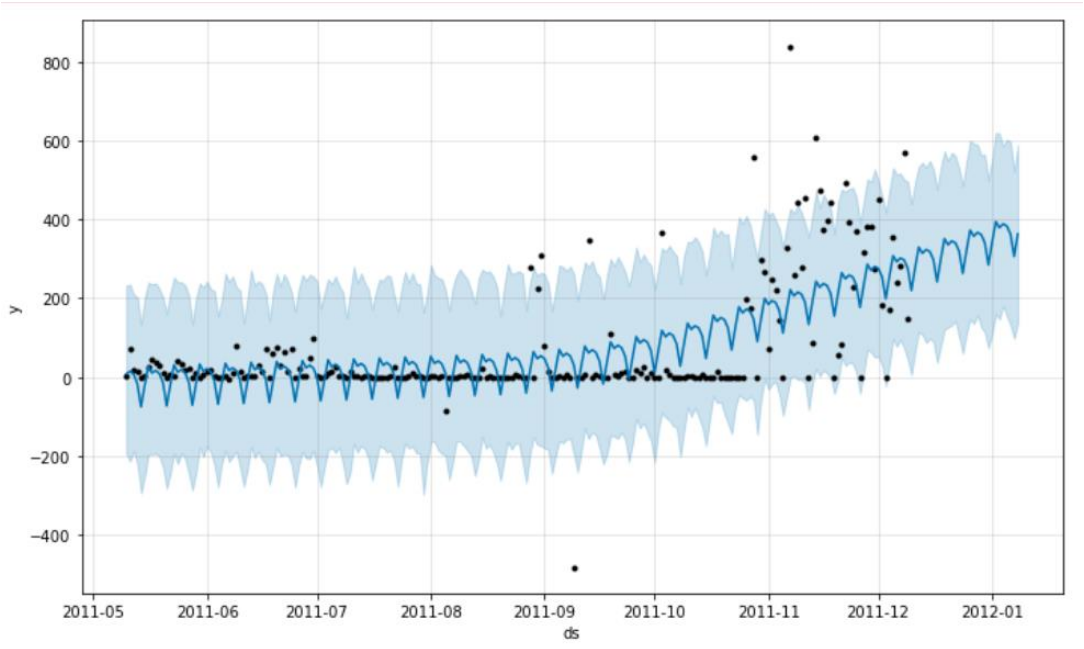
	country	stock_code	ds	y
5288	United Kingdom	23084	2011-05-10	2
5289	United Kingdom	23084	2011-05-11	71
5290	United Kingdom	23084	2011-05-12	17
5291	United Kingdom	23084	2011-05-13	16
5292	United Kingdom	23084	2011-05-14	0
...	...	...	...	...
5497	United Kingdom	23084	2011-12-05	354
5498	United Kingdom	23084	2011-12-06	240
5499	United Kingdom	23084	2011-12-07	283
5500	United Kingdom	23084	2011-12-08	570
5501	United Kingdom	23084	2011-12-09	150

The below chart shows the 30-day forecast for the above dataset.

The 95% confidence intervals capture the actual points most of the time, which is a good sign.

We can observe how Prophet is able to automatically handle weekly and daily seasonality easily. Yearly seasonality was deliberately excluded since we only have a year of data.

Within the forecast period of interest, 11/27/2011 - 12/3/2011, we observe that every actual falls within the 95% forecast confidence intervals, lending further credibility to Prophet's ability to forecast accurately with little to no data pre-processing.



	ds	yhat	yhat_lower	yhat_upper	actual	actual_within_CI
0	2011-11-27	233.635255	-4.194539	425.559387	317	True
1	2011-11-28	286.447956	57.957924	510.679746	381	True
2	2011-11-29	271.222913	53.916497	490.939203	381	True
3	2011-11-30	280.451973	46.691881	511.359367	276	True
4	2011-12-01	275.712136	57.539085	512.338411	449	True
5	2011-12-02	254.992993	40.412287	474.813714	181	True
6	2011-12-03	198.604575	-4.283309	419.190896	0	True

One obvious tradeoff with Prophet is lack of interpretability. That said, for highly chaotic datasets such as the small count SKU level data seen above, our inferences would not matter much anyway since they would be highly unstable. The gains from drawing stable inferences would be worthwhile pursuit but would be a project unto itself.

One drawback with Stan is that it does not leverage the GPU, making calculations slower than necessary. Prophet uses multi-core parallelization for the CPU, but the speedup is small relative to what we would gain using a GPU since a GPU is akin to thousands of mini-CPU's.

Prophet requires at least 2 data points for each time series, so I had to build in some error handling for situations where it tries to fit a model to time series with not enough data.

I used the tqdm library which tells us how long the program will take to run. Furthermore, it gives us an upper bound on gains from code speedups on each section of the loop.

I wrote the code so that it requires minimal code changes and enables the reader to quickly understand what is going on. It can certainly be optimized further, but it is a good first cut.

It would be interesting to see how applying preprocessing techniques such as making the data stationary would impact the models generated by Prophet.

Alternative approaches such as traditional methods such as ARIMA could be leveraged as well. Time series models such as LSTM's, GRU',s and Transformers could also be leveraged via libraries such as TensorFlow. TensorFlow enables us to generate massive speedups since we can leverage the GPU for model training.

It would be interesting to have a bake-off between the traditional statistical forecasting and machine learning methods with the goal of minimizing forecast errors as well as maximizing speed and inferential capabilities. Indeed, this in-line with my PhD research in AutoML, which seeks to find the best model automatically for a given arbitrary set of input data within a fixed time budget.

The following links were used to install the additional non-standard Python libraries used in the analysis.

https://github.com/tqdm/tqdm

https://facebook.github.io/prophet/