Stranger Forecasts

Maxwell Peterson

Abstract

We motivate and demonstrate a method we call lagged-feed-xgboost (LFX).

This is an experiment with timeseries prediction using the boosting algorithm xgboost. We focus on Arizona Yelp reviews-per-day.

Method

The idea is to train an xgboost model where the input features to predict the value of the timeseries at y[t] are N lags of y up to time t. For example, if we used 3 lags, then to predict a future out-of-sample value y[i], we would use the 1-by-3 matrix [y[i - 1], y[i - 2], y[i - 3]].

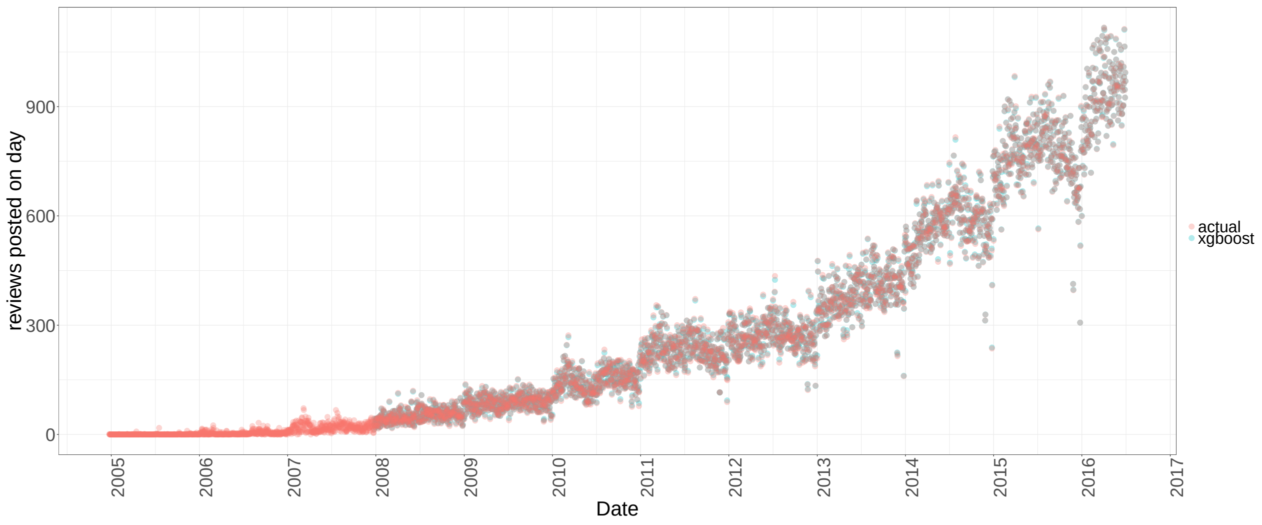
For forecast horizons longer than 1 day, not all lags are known. For example, when forecasting the day k days after the final known date, there are k - 1 unknown lags of y[k]. Thus we use a one-step-forecast-and-feed procedure: starting at the day after the final known day, we predict one value at a time, and use the predictions as if they were known lags of the series. With this scheme, the 1st lag of y[k] is the single value predicted at step y[k - 1].

Data

We train on data from December 24, 2004 to June 30, 2016, and forecast over a horizon of 365 days after June 30, 2016.

In-sample fit

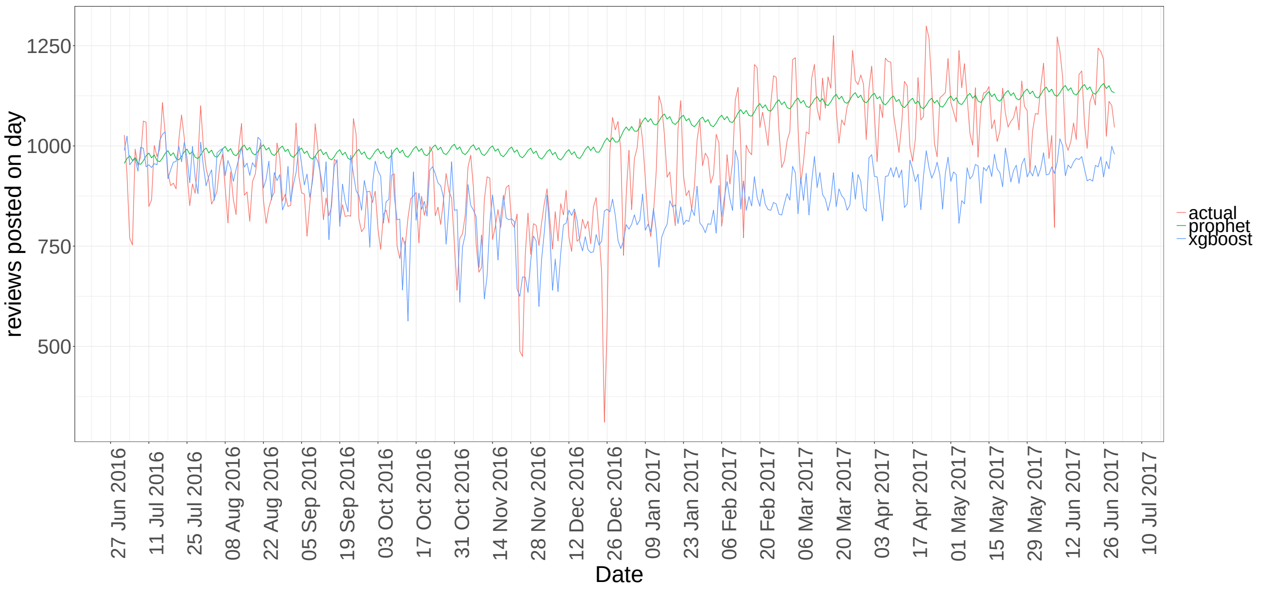
This method has no problem fitting the training data:



But it isn't clear whether this is a good thing.

Out-of-sample forecast

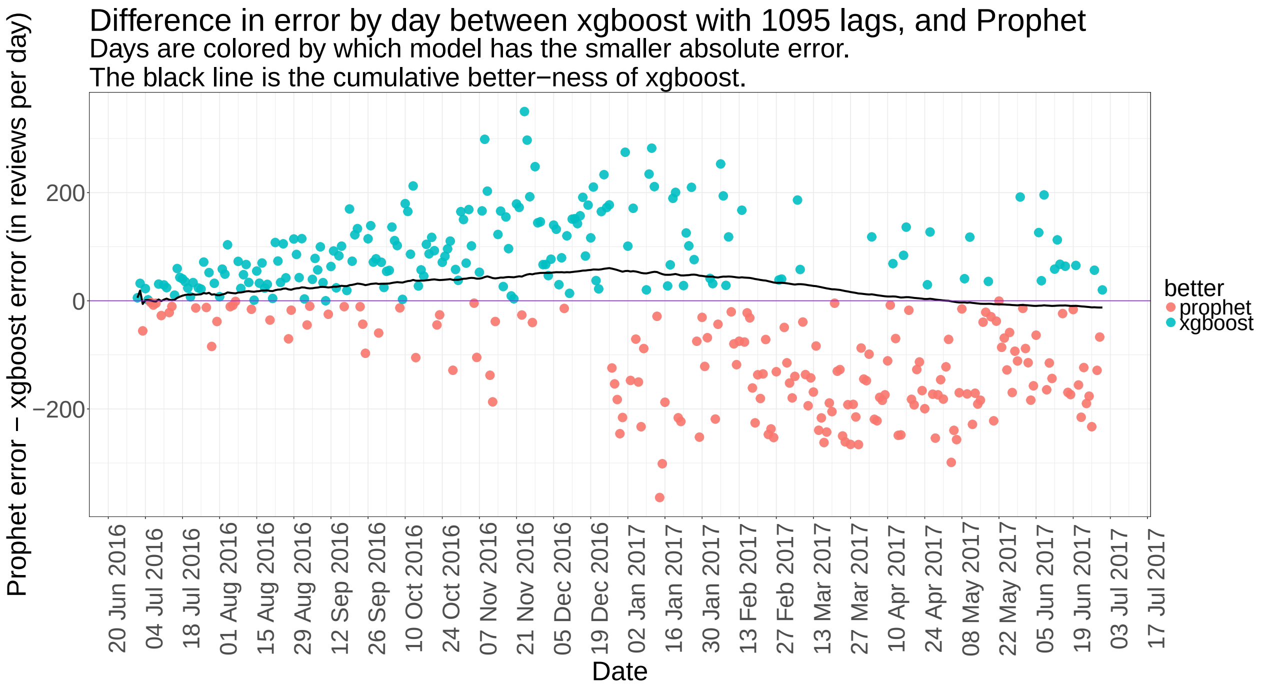
Using 1095 lags as features, the following forecast results. The forecast from a prophet model trained on the same data is also plotted.



What interesting behavior! And what differences between the two methods! The prophet forecast is stodgy and safe, preferring to skip attempting to capture the day-to-day variation in favor of attempting (but not succeeding) to stay around the local mean of the series; the xgboost forecast wants it all, and reaches all around day-to-day. The immediate suggestion is that for long horizons, prophet may be a safer bet; but that for day-to-day variation attempts, this xgboost method has much more promise.

Comparing errors over time

LFX is the better method for this series for the first 6 months or so, then Prophet begins to do better:



The reason Prophet overtakes lagged-feed-xgboost is that the latter fails to increase enough at the beginning of 2017, so is aiming too low.

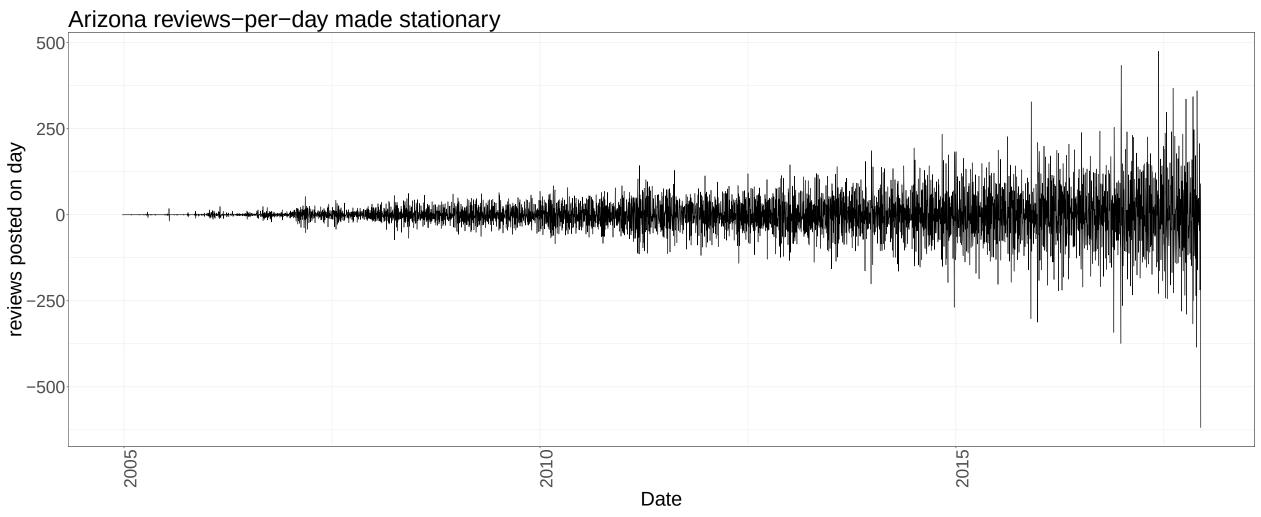
The global variable importance from the xgboost model is interesting:

```r
series_result$importance %>% head(20) %>%
  xtable::xtable(align = "llr") %>%
  print(type = "latex", include.rownames = FALSE, comment = FALSE)
```

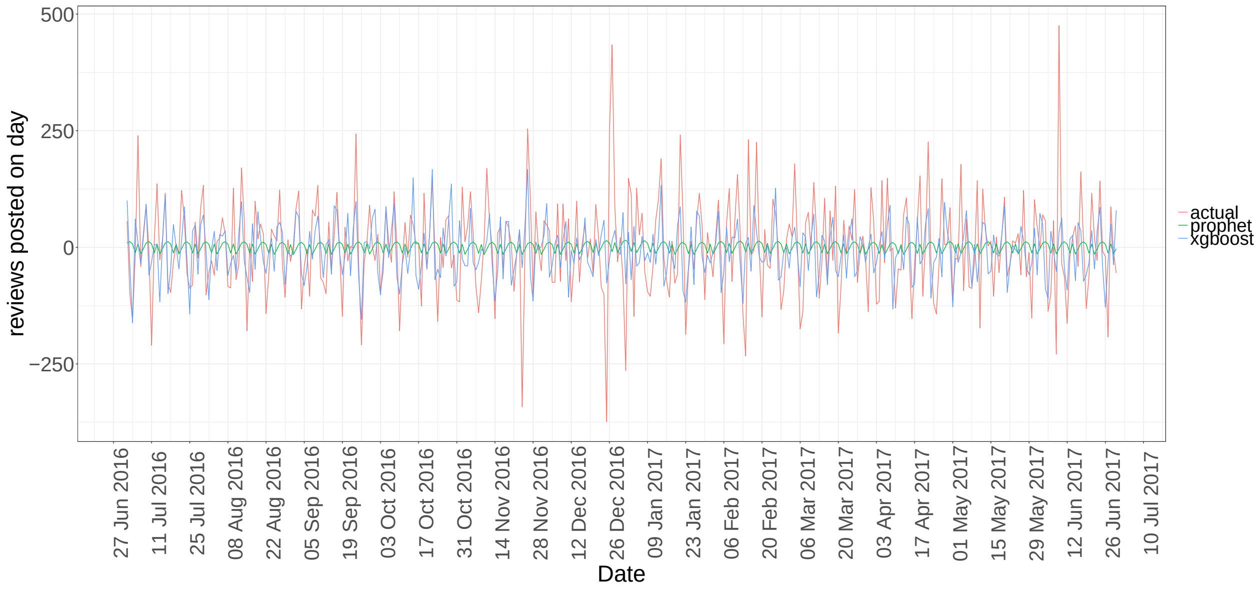
lag_number	Gain
1	0.50
7	0.23
6	0.10
364	0.08
20	0.02
49	0.01
312	0.01
363	0.01
371	0.00
341	0.00
14	0.00
713	0.00
21	0.00
1071	0.00
28	0.00
411	0.00
1092	0.00
8	0.00
356	0.00
740	0.00

On the stationary series

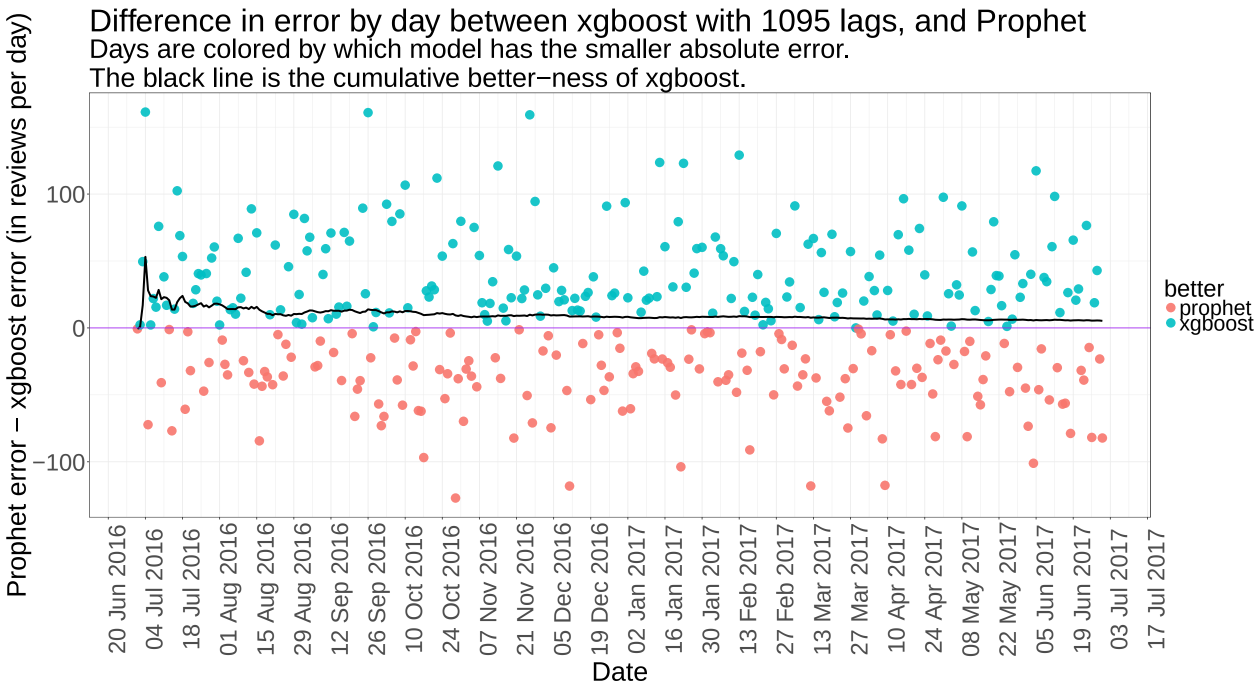
So far, the two methods are mainly doing better as a function on what they predicted the level of the series to be. How do they compare on a stationary series? Removing the upward trend from the series with single-lag differencing, we obtain the following series to model and predict upon:



Training the lagged-features xgboost model and prophet model and forecasting just as before, we obtain the following forecasts:



And daily errors:



Lagged xgboost outperforms prophet on the stationary series!

Implications; Avenues of Future Investigation

- xgboost is a general enough framework that adding external regressors is straightforward For example, if temperature is known to correlate with review counts and temperature forecasts are good enough, temperature (and/or lags of temperature) could easily be included as a reviews-per-day predictor.
- Day-to-day forecasts from LFX could be combined with a separate estimate of longer-term trend

If LFX turns out to forecast day-to-day variation well but longer-term trend less well (like in this document), LFX predictions could be added to trend estimates (from e.g. exponential smoothing) to create a better forecast.

- Hyper-parameter tuning is reasonably well-developed for xgboost and models like it. e.g. the hyperopt package in Python
- Variable-importance techniques like LIME, and in general many meta-modeling techniques for classifiers, apply.