**Comittee of Machines(NN), Boosting, Mixture of Experts , Ensemble Averaging**

**Timeseries Forecasting Optimization**

**Introduction to Raw Data:**

A major financial institution has invested heavily into the CF toolset and methodology and has completed their 'journey' in combining, contextualizing, tuning, and finally aggregating novel KPIs. This data now represents status indicators in key business areas. There are several hundred of these feeds that they care about. Why do we care?

What's shown below are 1-sigma boundaries on daily aggregated data.

iscrete a-bounda 
, Quarterl trainin 

iscrete T-bounda 
, 30-da trainin 
810 
795 
790 

The second is clearly an improvement over the first, but even the 30-day 'refresh' lags the actual signal enough to introduce problematic alerts. This mismatch is minimized with a continuously (daily) updated boundary. The following shows a trailing 30 day window updated daily. The window could be reduced even further to be even more responsive.

ontinuous a-boundary, 30-day window 
Feb 2014 
Mar 2014 
Apr 2014 
May 2014 
Jun 2014 
Jul 2014 
Aug 2014 
sep 2014 
Oct 2014 
Nov 2014 
Dec 2014 

**Low-hanging fruit harvested on 'alerting' topics.**

Many improvements can be achieved with a modest amount of effort here. With curiosity satisfied, we moved on to the idea of 'how good we can get a prediction', and 'what we can use predictions for'.

Q: what can we use predictions for in this context?

**Prediction:**   
This video mocks up what could be done on the 1-feed only side, including a machine learning method of prediction (ARMA) for future values. The up/down arrows for "Today's" values indicate an push/email alert for breaching the boundaries.

The stars shown in the future are predicted values.

The predictions are mocked up--as the training of the model happens only once with data from the period, and not continuously updated for each daily timestep.

<<Alerting\_ARMA\_5\_5\_4\_1fps\_2logo\_yesterday\_today\_tomorrow.mp4>>

**Forecast Data:**

The datasets consist of a number of separate daily data feeds. These are dashboard-level aggregations of customer-journey business metrics spanning 10 months. There are 6 predictions from various ARIMA models 10 months, complete with how well they've done.

Given a set of raw data--consisting of ~10 months of FOX-processed daily feeds (or combinations of daily feeds…) we provide:

6 ARIMA-like predictions in the given data. They are lovingly referred to as:

* data
* cycle\_series
* differenced\_1\_data
* differenced\_1\_cycle
* differenced\_2\_data
* differenced\_2\_data

**ARMA/ARIMA Background**

Timeseries observations fall into a few well-known classes based on the statistical properties exhibited by the dataset's mean, variance, and error terms. Frequently, timeseries data is a result of a process governed by at least random and at most periodic forces with amplitudes which either detract from or reinforce the predictability. These interdependencies can be both gross and subtle. To characterize these, we appeal to the well known Auto-Regressive Moving Average, (ARMA) models--essentially applied *spectral analysis* for econometrics--and generalize the formulation to include preconditioning the input data in various ways to build a predictive framework to forecast future values of the timeseries data.

The starting point for this exposition is an auto-regressive term--two Latin words that when combined indicated that a series is either going back to previous values on its own, or is based on the past values (although English's take on *regress* it is more pejorative, but I digress). So, this component of our formulation is responsible for modeling the future based on previous values and includes periodic component of the timeseries data. To capture these dependencies we seek equations that yield values at a given time (present or future) based on functions of the values at past (lagged) observation times. These are succinctly written in the following form in equation 1:



Where is assumed to be independent and identically distributed random noise (a Gaussian or normally-distributed noise term of zero mean and unit variance). The *pth* term and corresponding coefficient are the order of AR portion of the model and determine the maximum lag taken to be an influencing factor. So for a process which is determined by the previous term alone, p=1 yields an equation where the previous value is equal to the prior--within a scale factor.

It can similarly be assumed that processes exist whereby the current value in a timeseries dataset depends on an integer number of linearly-combinable previous shocks, or stochastic excursions from the mean. This can be seen as a finite implementation of the unit impulse response function or filter. These can be denoted by the following functional form:



While this component of the greater ARMA (and ARIMA) class of predictive models depends in principle in a similar way, its specific action on the future data is markedly different. With the dependency seen in equation 2 of the current value based on previous epsilon terms, it can be seen that these shocks only propagate into the future for q-timesteps, whereas the definition given for the AR term in equation 1 has only the shock at time t factoring into future values of the univariate dataseries. Equation 2 is essentially a linear regression equation, with the same assumptions on epsilon--with the added restriction that all previous unobserved shocks are drawn from the same distribution explained above.

Spoiler alert: equations 1, 2 can be combined (they're both linear models, so a linear combination is still valid) to yield a comprehensive description of the current value of a dataseries explainable by values and shocks at previous times, the ARMA model. The general ARMA(p,q) model is simply written as:



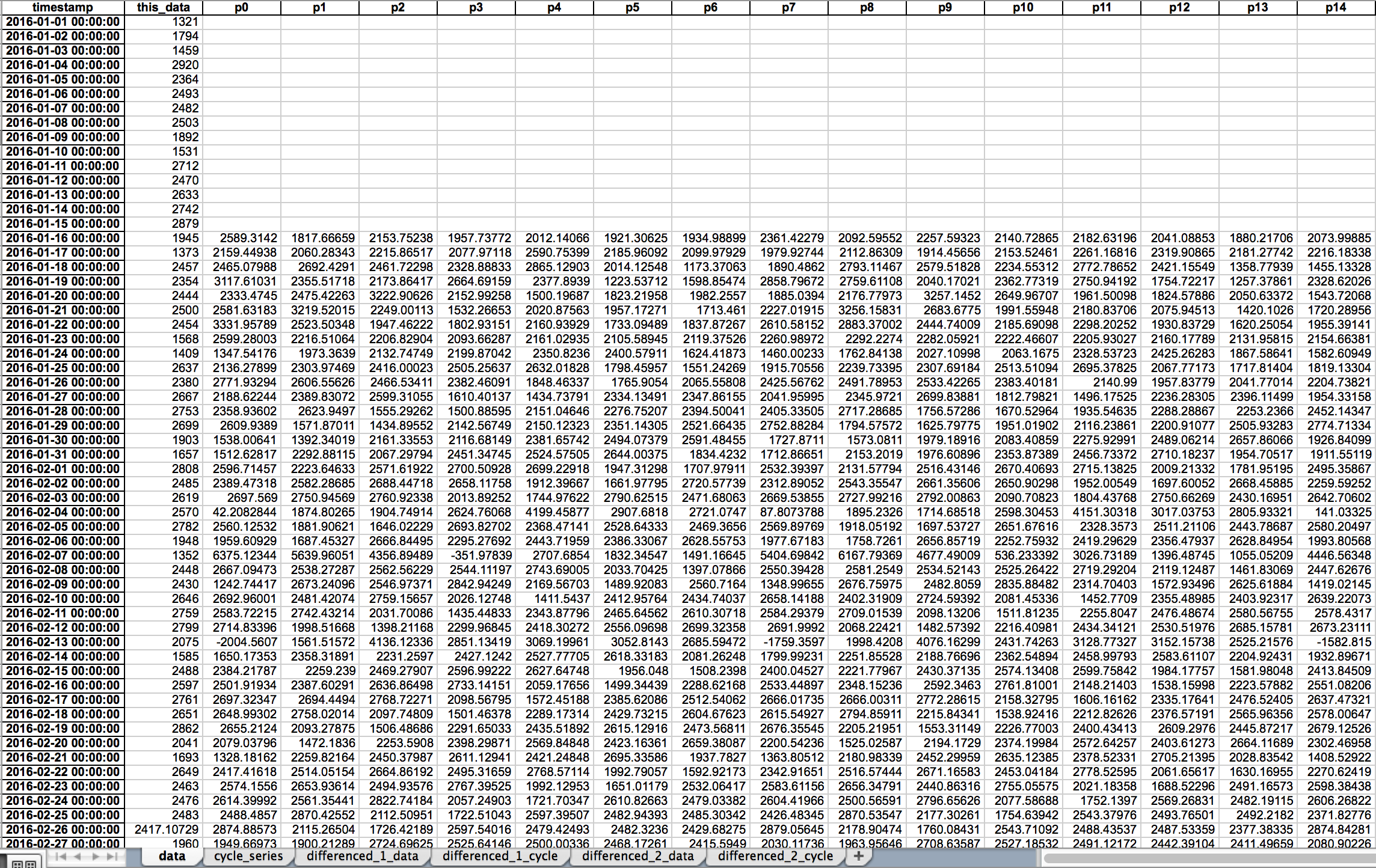
Where it is sometimes taken that the series is zero-meaned (, and (3) is often expressed in terms of lagged operators--a formalism which is omitted here for brevity. Equation 3 imposes restrictions on the solution that comes in two classes: based on invertibility and stationarity of the proposed numeric solution. These limitations on the stability can both introduce unstable predictions for future values as well as prevent the model from presenting a soluble form for a given set of p, q. These will be addressed in the particular formulation of the prediction engine given.

**References:**

* Enders, Walter (2004). "Stationary Time-Series Models". Applied Econometric Time Series (Second ed.). New York: Wiley. pp. 48–107. [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [0-471-45173-8](https://en.wikipedia.org/wiki/Special:BookSources/0-471-45173-8).
* Mills, Terence C. (1990). Time Series Techniques for Economists. Cambridge University Press.
* Percival, Donald B.; Walden, Andrew T. (1993). Spectral Analysis for Physical Applications. Cambridge University Press.
* Pandit, Sudhakar M.; Wu, Shien-Ming (1983). Time Series and System Analysis with Applications. John Wiley & Sons.

**On to the data…**

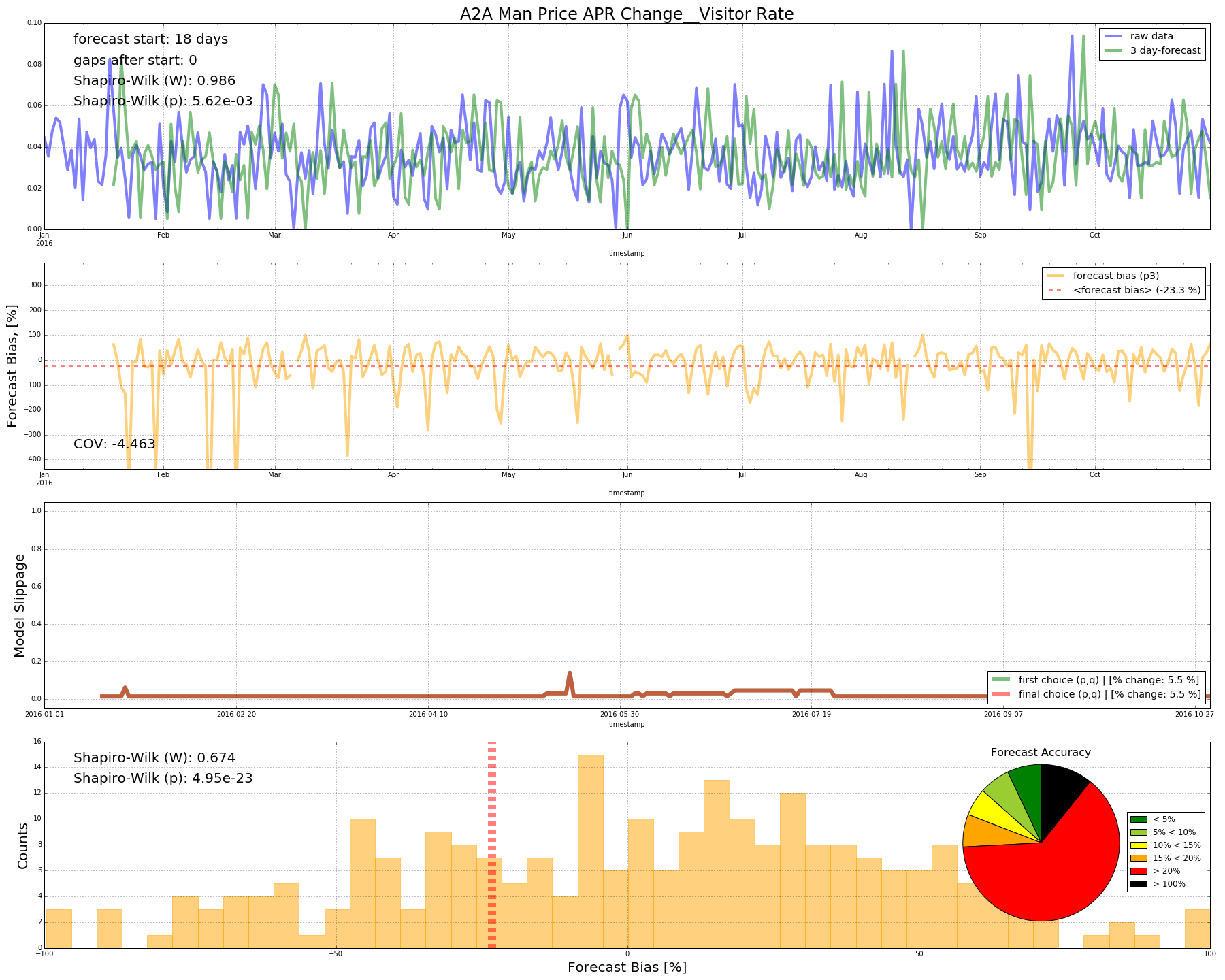
Been DQ'ed, QA'ed and as mentioned, predictions already made (?--or *have* they been? Possible topic here).



Typical single-method predictions when the method works:



Typical single-method predictions when the method doesn't work:



Ok, so just use the method that works…right?

Um, how?



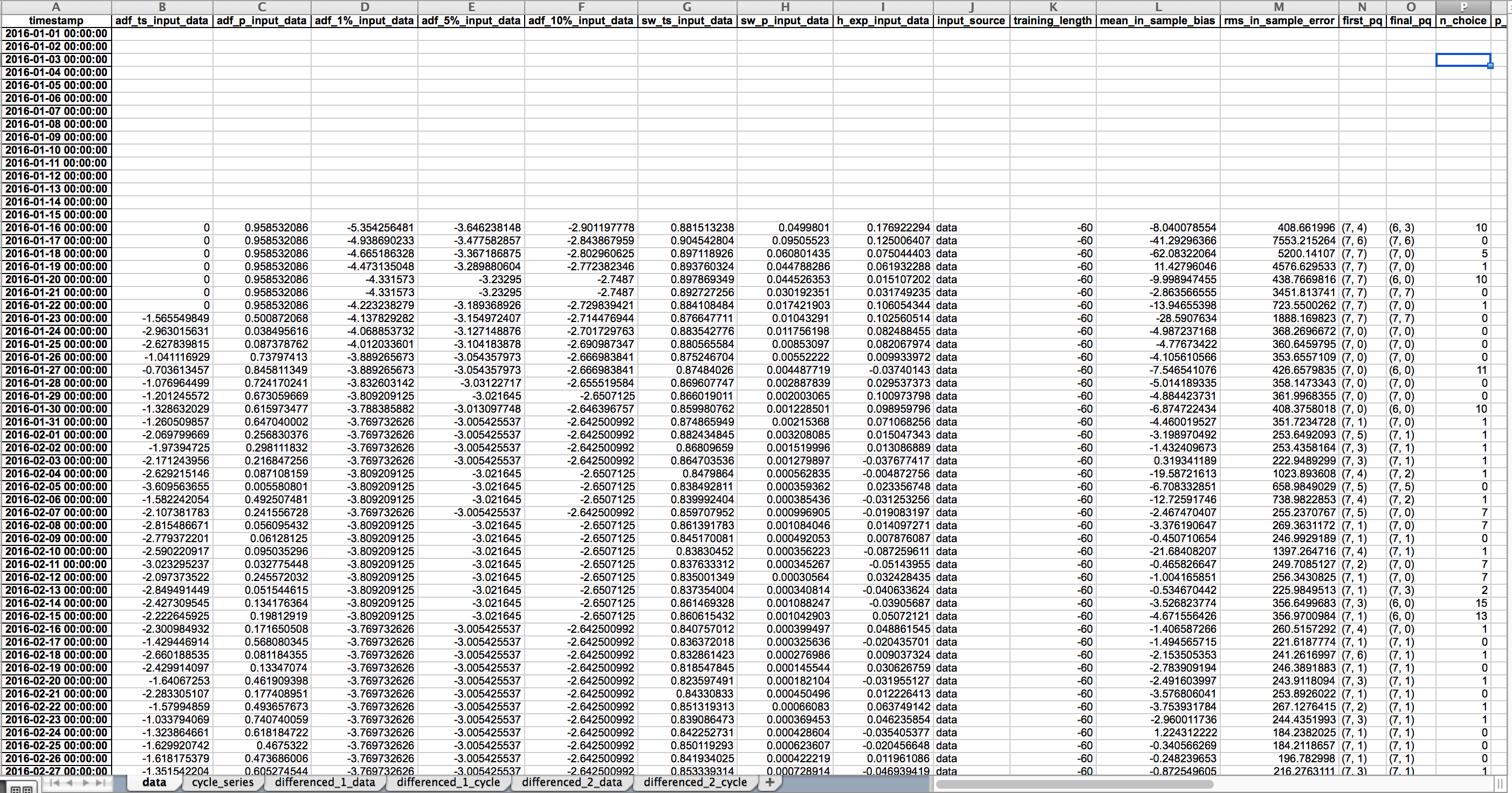
This is a compilation result…we've literally taken the best estimate from each of the 6 and published that for that day. The results are ideal from the ‘best choice to minimize the mean error’ perspective.

*(forecast\_bias\_p6\_XXXXXXXX\_t\_60\_compilation\_results)*

**Ok, but what do we know about the data?**

***(Data provenance)***

For each step (daily), and for the training data (the previous 60 days), various metrics are calculated which are designed to help in understanding how appropriate the given data will be for the model. This is a good start, but is not an exhaustive list of possible indicators…What’s more, these metrics aren’t conclusive. By themselves, they do not indicate which is the best choice as evidenced by the data in light of standard error metrics. This is the problem at hand, in a nutshell.



|  |  |
| --- | --- |
| **NAME** | **DESCRIPTION** |
| **timestamp** | yep, it's a timestamp. This is the date for the given data/prediction. Remember the shift for the future values of the forecast! |
| **adf\_ts\_input\_data** | the TS-component of the Augmented Dicky-Fuller test for timeseries stationarity on the training data. |
| **adf\_p\_input\_data** | the p-component of the Augmented Dicky-Fuller test for timeseries stationarity on the training data. |
| **adf\_1%\_input\_data** | the 1% confidence level of the Augmented Dicky-Fuller test for timeseries stationarity on the training data. |
| **adf\_5%\_input\_data** | the 5% confidence level of the Augmented Dicky-Fuller test for timeseries stationarity on the training data. |
| **adf\_10%\_input\_data** | the 10% confidence level of the Augmented Dicky-Fuller test for timeseries stationarity on the training data. |
| **sw\_ts\_input\_data** | see Shapiro-Wilk Test (this is the TS component) |
| **sw\_p\_input\_data** | see Shapiro-Wilk Test (this is the p-component) |
| **h\_exp\_input\_data** | the Hurst exponent of the training data. |
| **input\_source** | this is the type or category of data that went into the model (raw data, decomposed cycle, differenced level, etc.) |
| **training\_length** | number of days of data that trained the model. Held constant for this batch. |
| **mean\_in\_sample\_bias** | average error as measured by the forecast bias formula when predicting on the training data. |
| **rms\_in\_sample\_error** | average error as measured by the root mean squared formula when predicting on the training data. |
| **first\_pq** | a magic method determined which coefficients should be used. This is its first choice. |
| **final\_pq** | if that failed one or more DQ checks or suffered internal errors, the method moves on to the next one. Here's where it ended up. |
| **n\_choice** | This is how many 'skips' between the first\_pq and final\_pq there were. |
| **p\_zero** | this can be ignored for this purpose. |
| **blank\_count** | the number of NaNs in the forecast. Generally == 0 for these version of prediction engine. |

With the raw data in the prediction file, you're free to create whichever metrics you'd like as well.

**Options from here:**

After satisfying the requirements from Adam (generate your own version of an ARIMA predictor based on the code in the repo), there are some options.

You could:

Create predictor-selector. Come up with an ensemble method or other to decide which model to use.

* Ensemble learning
  + Straight selection
    - Can't beat our 'best'
  + Voting with or without weights based on statistics/ML metrics
* NN/ deep methods.
  + Selection based on this.

Or you could:

Create a prediction method yourself…

* NN/ deep methods
  + Prediction

Within each branch above, comparison to previous BB best as benchmark.