Time Series Forecast in BigApps

Created by Minh Tran, last modified on Feb 24, 2016

Introduction

Several types of real-world data such as stock prices, sales volumes and interest rates are collected over time, forming time series. These are potential data that can be used to drive business decisions. Due to their sequential nature, special statistical techniques need to be developed in order to support customers in mining their business values from time series data. For instance, a good forecast on stock prices can benefit business users in their decisions of selling or buying. Obviously, efficient time series forecasters are essential in any data-driven application that serves as a tool for business users.

BigApps is not an exception in that it needs to integrate different time series forecasting algorithms to serve various demands of its customers. In this technical report, we will present different time series forecasters that are added into BigApps. We select a number of efficient algorithms with high accuracy based on analyzing and exploring special characteristics of time series data. In addition, we also provide a meta algorithm that supports business users in automatically selecting a suitable algorithm for their data.

The rest of this document includes the following sections. We first formulate our problem statement and scope. Then, we present some background concepts that are necessary to understand the technical aspects of this document. Next is our methodology as well as the meta algorithm. We also mention some experimental results before reaching our conclusions.

Problem Statement & Scope

Our problem statement can be formulated as simple as follows. Given a time series at any scale, we need to develop many different forecasting algorithms to predict the values in the next k units, where k is a business parameter and unit can be at hour, day or week, etc., depending on the scale we choose for the time series. Business users can choose k as large as they want for their business plan, remembering that predicting too far in the future may lower the accuracy.

In this version of time series forecasters integrated in BigApps, we provide 4 different forecasting (core) algorithms plusing a meta algorithm that helps select a suitable core algorithm based on analyzing the input time series. We will focus our product on 3 time series characteristics including periodicity, level stationary and trend stationary, which will be defined in the next section.

Backgrounds

We present in this section common characteristics of a time series that can be supported in our product. In addition, the theory of periodicity transform [1] will also be briefly described. Periodicity transform plays an important role in detecting and decomposing periodic components in a time series.

Common Time Series Characteristics

A real world time series usually contains important characteristics. For example, a time series of stock price can have periodicity and/or stationary. These characteristics might impose crucial information for modeling and forecasting. Thus, exploring and utilizing those characteristics in forecast can potentially improve accuracy. Therefore, we will present in this section some backgrounds of common time series characteristics such as periodicity and stationary.

Periodicity

Periodicity is observed because time series usually appear in cycles. The length of a cycle can be in the order of hours, days, months, years or any time length, of which the weekly cycle is a well-known and widely recognized phenomenon in business time series. Detecting the periodicity of a time series process can be done via observing its autocorrelation function (ACF) given by

An example of a time series process with periodicity is given in Figure 1. We can visually observe cycles in its ACF. The autocorrelation at lag 0 is 1 because the process is compared with itself, thus, the ACF results in an exact match. Then when the lag increases, the autocorrelation quickly decays. As we can see, the ACF repeats with a lag of 24 hours since the process tends to match itself whenever the lag is an integral multiple of 24. For a deeper understanding, see [2].

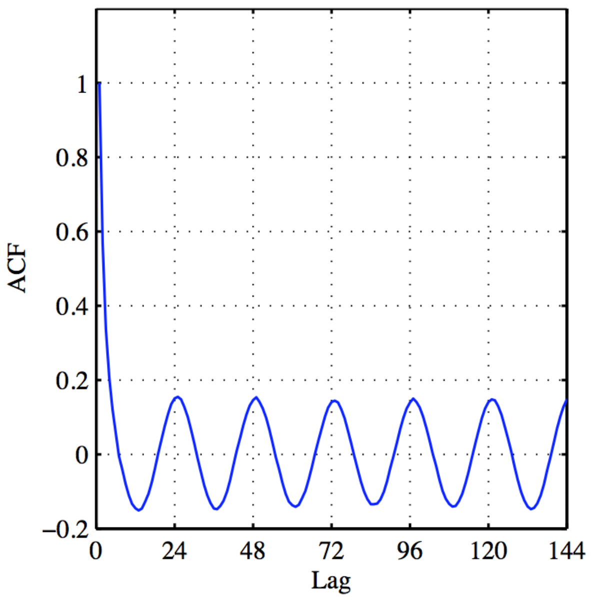
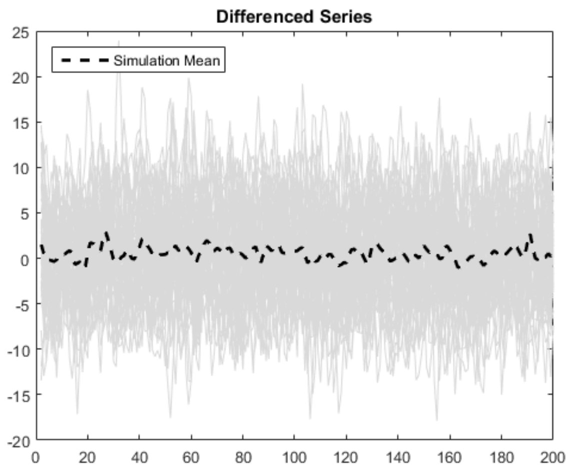


Figure 1: Example of periodicity.

Stationary

A (level) stationary process is a stochastic process whose probability distribution does not change over time. Consequently, parameters such as mean and variance do not change over time and do not follow any trend [3].

A process {*X*} is trend stationary if , where *f* is a function, and {*e*} is a stationary process. The value of the function is the trend value of the process at time *t* [4]. An illustration of level and trend stationary is given in Figure 2.



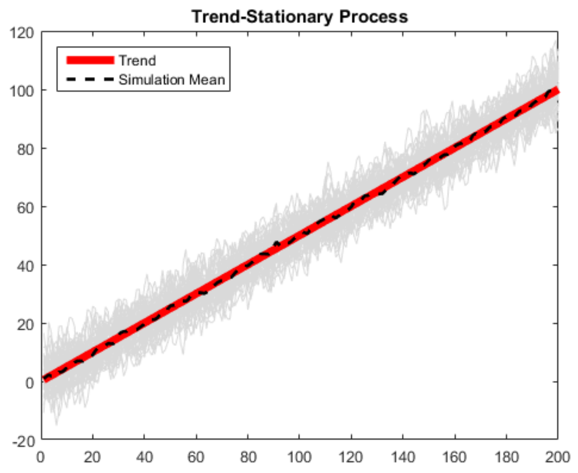


Figure 2: Illustration of level stationary (left) and trend stationary (right). Figure source: www.mathworks.com.

Periodicity Transform

As mentioned earlier, our work focuses on 3 common time series characteristics including periodicity, thus we need an efficient theory to deal with this characteristic and the theory of periodicity transform fits our purpose.

It is well known that signal decomposition techniques are powerful to detect and extract periodic sequences in a time series containing cycles. Introduced by Sethares and Staley [1], the periodicity transforms procedure decomposes a point process into a sum of simple periodic sequences, leaving residuals without periodicity. This decomposition is achieved by using periodic sequences, not by frequency or scale as in the Fourier and wavelet transforms [1, 5].

The M-Best algorithm suggested in the periodicity transforms procedure takes a sequence $\{X_i\}$ containing cycles and a parameter M as its input. The procedure will analyse the sequence $\{X_i\}$ to search M sub-sequences of $\{X_i\}$ that are periodic sequences, denoted by , j = 1 M. The search is done in such a way that is the most important periodic sequence, is the second most important periodic sequence, and so on, until . In other words, the periodicity transforms procedure helps to find M most important periodic sequences of $\{X_i\}$. The term “important” is quantified in the sense of the norm measure which is defined on a concept of inner product. Given two sequences $\{U_i\}$ and $\{V_i\}$ that are u-periodic and v-periodic, respectively. The inner product between $\{U_i\}$ and $\{V_i\}$, denoted by $< \{U_i\}, \{V_i\} >$, is calculated as

The norm of a periodic sequence $\{U_i\}$ is then defined as

The periodicity transforms procedure finishes by calculating a residual sequence $\{R_i\}$, which is obtained by removing , j = 1 M from $\{X_i\}$. The couple , j = 1 M and $\{R_i\}$ form the output of the procedure. The relationship between this couple and $\{X_i\}$ is described by

An illustration of applying the periodicity transform theory to decompose a time series is given in Figure 3.

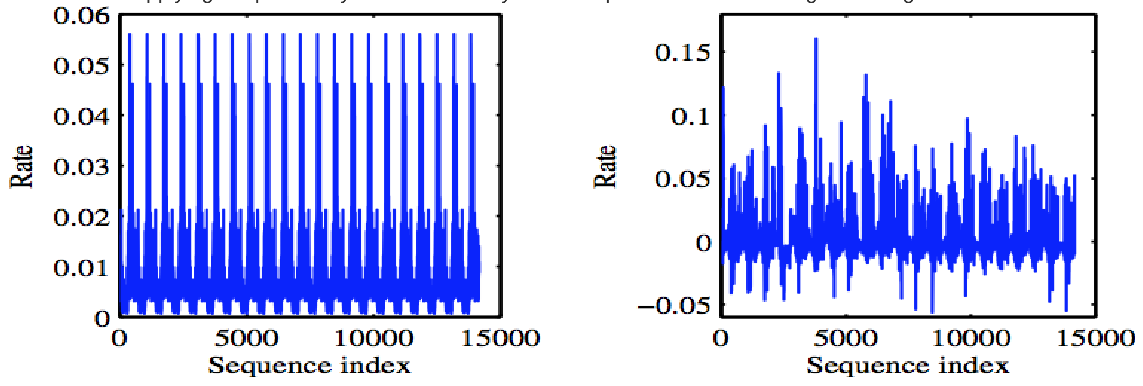


Figure 3: Illustration of applying periodicity transform theory on time series.

Methodology

Our methodology for building time series forecasters is described as follows. Given a time series input, we will decompose it into 2 components including the periodicity component and the residual component using the periodicity transform theory if it is detected that there is seasonal components present in the input data. Then depending on the characteristic of the residual component, a suitable algorithm will be chosen to model the residual time series. The residual model is then used to forecast future values which will be added with properly selected periodic values to form the final forecast.

We provide many choices for customers by the following core algorithms:

1. periodicity transform + arima (season = true) ptarima(season = true): used if data is periodic and the residual component is not stationary, i.e. residuals still contain some periodic pattern that is not detected by the periodicity transforms procedure.
2. periodicity transform + arima (season = false) ptarima(season = false): used if data is periodic and the residual component is trend stationary.
3. periodicity transform + holt ptholt: used if data is periodic and the residual component is level stationary.
4. arima (auto.arima, the advanced version of the arima family): used if data is not periodic but stationary.

Meta Algorithm

As the forecasting functionality that is integrated into BigApps supports several different forecasting algorithms, it could be hard for BigApps users to decide which forecaster they should choose for their time series data. This is also the purpose of the meta algorithm which aims at facilitating BigApps users by giving them recommendation on which forecaster may fit to their data.

The details of the meta algorithm is described in Figure 4. The meta algorithm operates by analyzing the time series characteristics that are present in the data. It uses statistical hypothesis tests such as the fisher.g.test and the kpss test to check whether a time series is periodic, level/trend stationary. The idea of the meta algorithm is as follows. It first checks whether the time series input is periodic. If it is not periodic then using arima if it is stationary. On the other hand, if the time series input is periodic, then using the periodicity transform theory to decompose it and model the residuals. Depending on whether the residual process is level/trend stationary, an appropriate forecasting algorithm will be recommended.

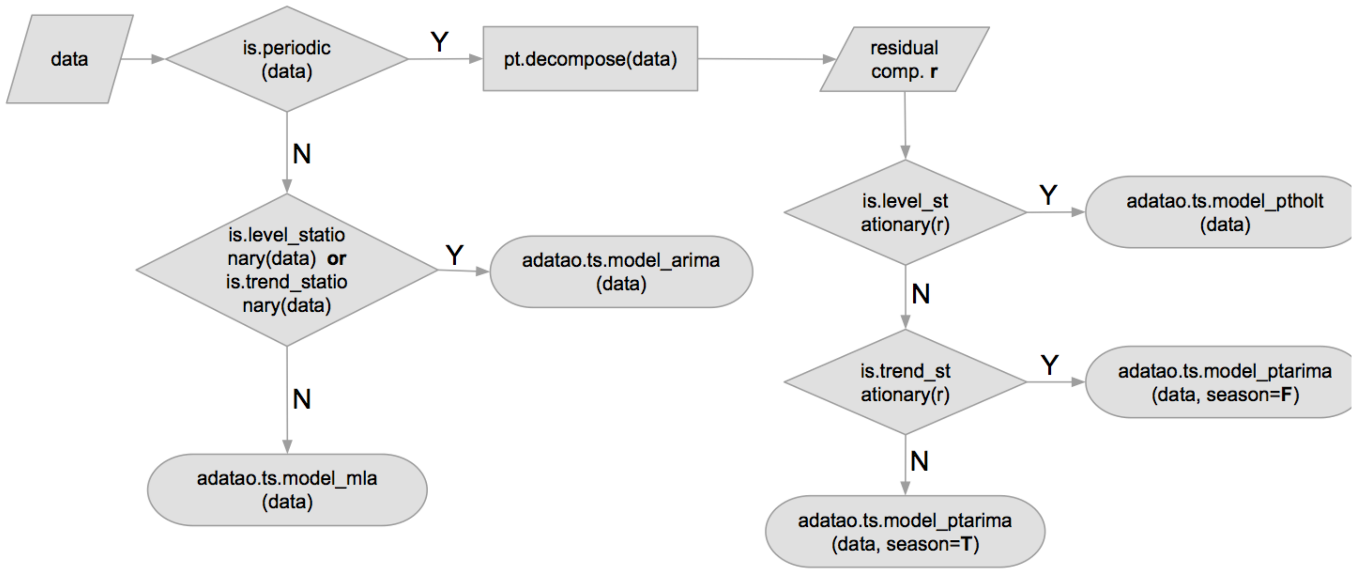

data
is.periodic (data)
Y
pt.decompose(data)
residual comp. r
N
is.level_stationary(data) or is.trend_stationary(data)
Y
adatao.ts.model_arima (data)
N
adatao.ts.model_mla (data)
is.level_stationary(r)
Y
adatao.ts.model_ptholt (data)
N
is.trend_stationary(r)
Y
adatao.ts.model_ptarima (data, season=F)
N
adatao.ts.model_ptarima (data, season=T)

Figure 4: The meta algorithm.

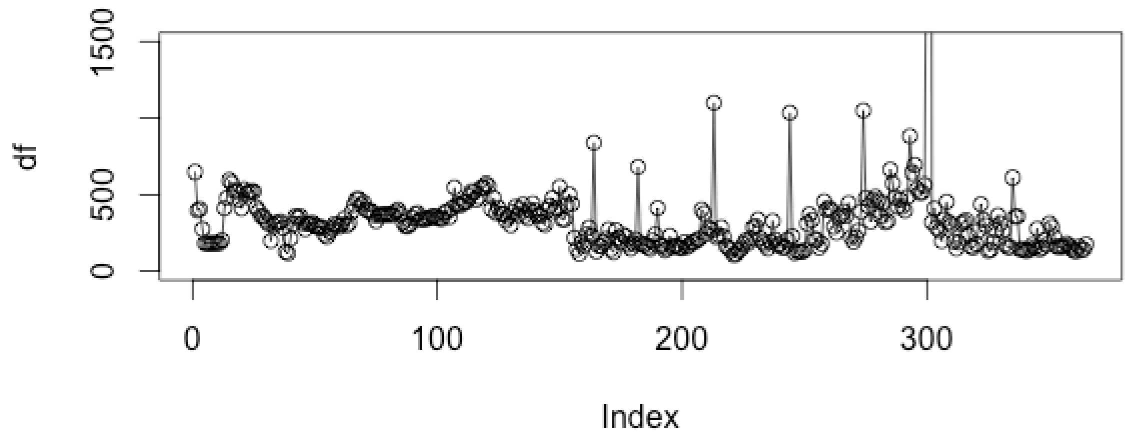
Experimental Results

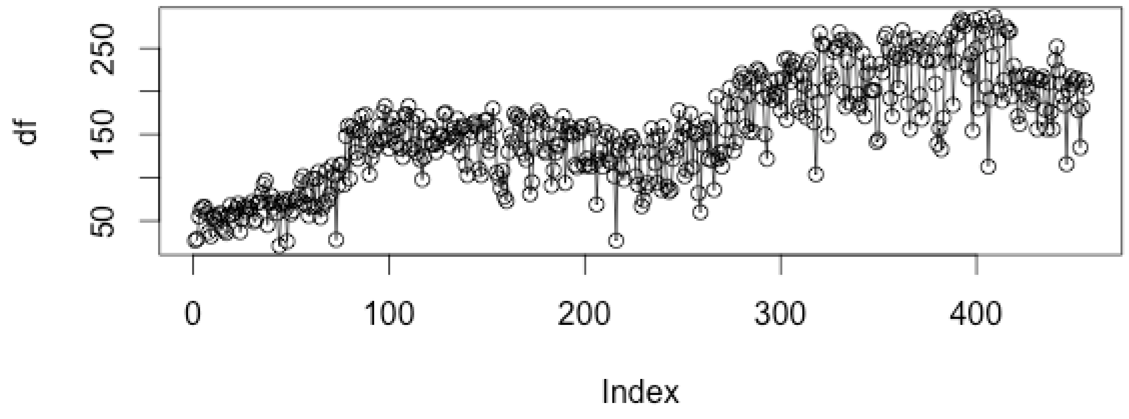
We implemented the meta algorithm as well as its eco-components such as the arima, ptarima and ptholt algorithms, and evaluated those on data sets described in Table 1. The metrics mean absolute error (MAE) and mean absolute percentage error (MAPE) are used in evaluation. Figure 5 draws the time series of some data sets in Table 1.

DataSet	Number of data points	Train/Test 7 days	Train/Test 10 days	Train/Test 30 days
bike_sharing	456	449/7	446/10	426/30
rossman	942	935/7	932/10	912/30

mobivi	205	198/7	195/10	175/30
flightinfo	7763	7756/7	7753/10	7733/30
nyctaxi	730	723/7	720/10	700/30
amazon_bills	365	358/7	355/10	335/30

Table 1: Data sets used in experiments.





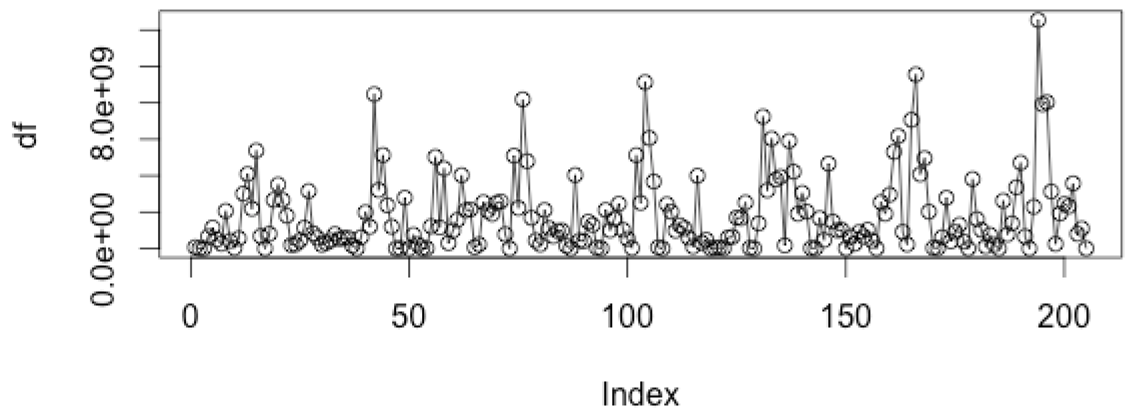


Figure 5: Time series of amazon (top), bike_sharing (middle) and mobivi (bottom).

In Figure 6, we draw the forecasting performance of the core algorithms on bike_sharing and rossman. As we can see, the newly developed forecasting algorithms work pretty well, comparing with HoltWinters, a common time series forecaster implemented in BigApps previously. Similar results are also obtained with other data sets (not shown for brevity).

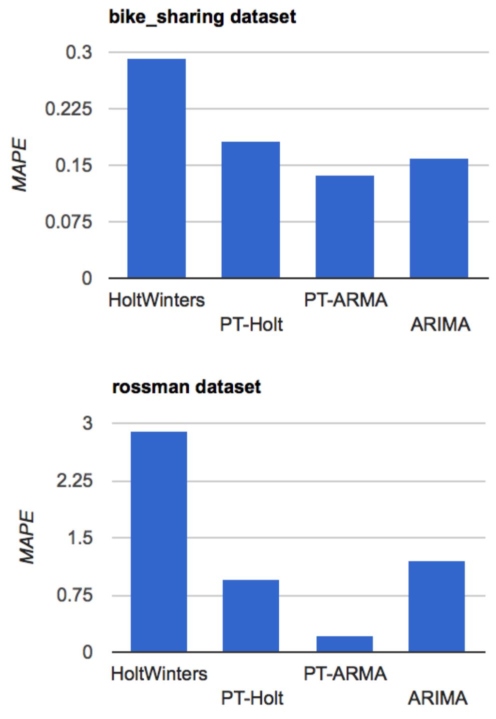
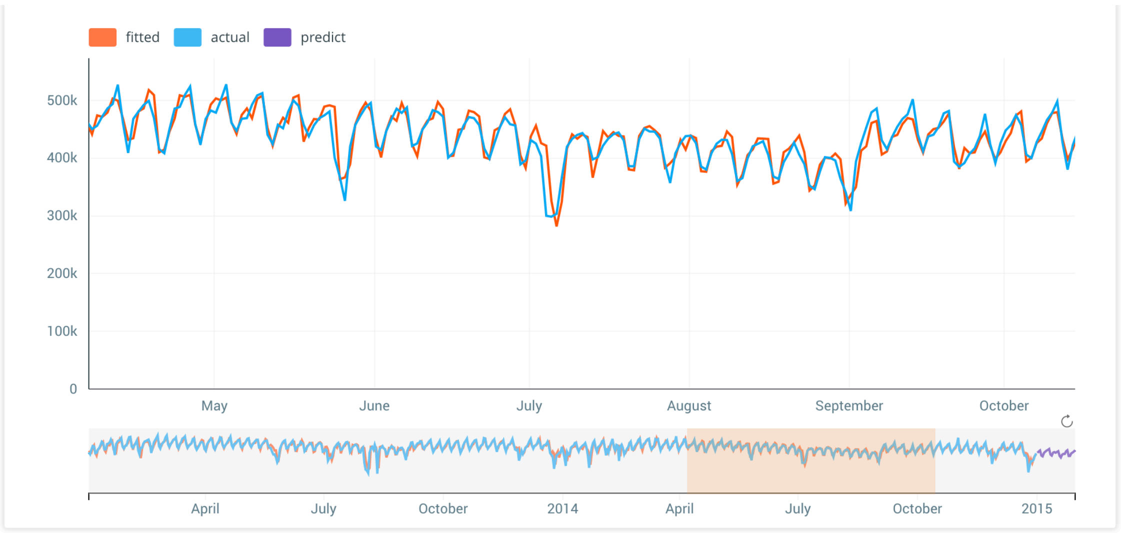
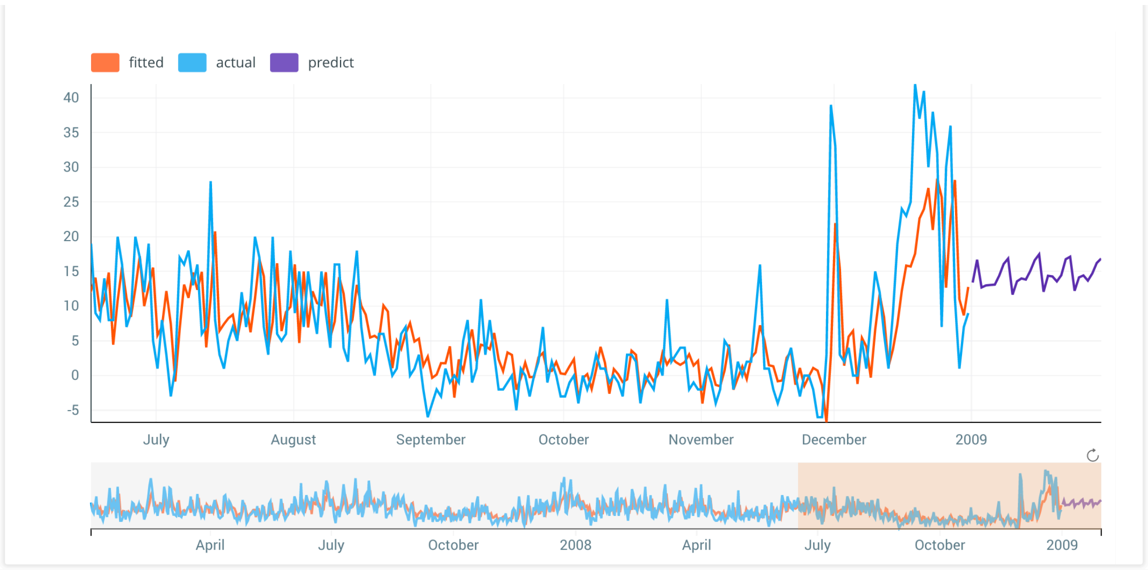


Figure 6: Forecasting performance on bike_sharing and rossman.





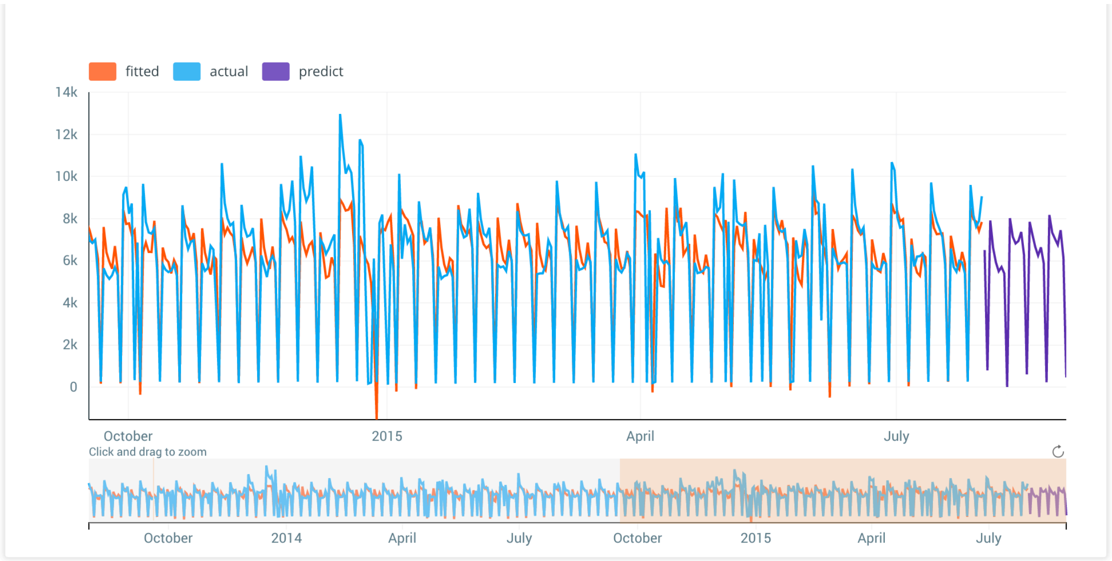


Figure 7: Visualization of forecasting results from BigApps for nyctaxi (top), flightinfo (middle) and rossman (bottom).

In Figure 7, we visualize the fitting and forecasting results from data sets like nyctaxi, flightinfo and rossman. As we can see, the fitting looks pretty nice with highly periodic time series. The visualization is provided as a functionality of BigApps.

Finally, we evaluate the quality of the meta algorithm by comparing its recommendation with the best forecaster (selected manually by observing their performance with metrics like MAE and MAPE). The evaluation is filled in Table 2. It is shown that the meta algorithm works pretty well as it can select exactly as the best algorithm in most of the cases. For data sets like flightinfo and nyctaxi, though it cannot recommend the best ones but the selected algorithms also work quite well (close to the performance of best algorithms).

DataSet	Recommended by the meta algorithm	Best algorithm
amazon_bills	adatao.ts.model_arima	adatao.ts.model_arima
bike_sharing	adatao.ts.model_ptarima	adatao.ts.model_ptarima
rossman	adatao.ts.model_ptarima	adatao.ts.model_ptarima
mobivi	adatao.ts.model_ptarima / adatao.ts.model_arima	adatao.ts.model_ptarima / adatao.ts.model_arima
flightinfo	adatao.ts.model_ptarima	adatao.ts.model_arima
nyctaxi	adatao.ts.model_ptarima	adatao.ts.model_ptholt

Table 2: Performance of the meta algorithm.

Conclusion

We have presented in this document the demands of time series forecasters. Such functionalities are essential in any commercial data science tool including BigApps. The BigApps application developed at Arimo has provided such functionalities with the following features: 1. it has multiple time series forecasting algorithms, and thus it provides customers with several choices, 2. it has a meta algorithm that can systematically support customers with good recommendations on selecting an appropriate algorithm for their input data, and 3. the forecasters implemented in BigApps are with high accuracy.

This is just our first version of time series forecasters that we developed with BigApps, several rooms still can be improved such as taking into account machine learning approaches, exploring other characteristics such as long range dependence, etc., and in particular, any improvement can also be done according to customer's demands.

References

[1] W. A. Sethares, T. W. Staley, “Periodicity Transforms”, IEEE Transactions on Signal Processing, vol. 47, pp. 2953-2964, 1999.
[2] D. G. Feitelson, “Workload Modeling for Computer Systems Performance Evaluation”, Cambridge University Press, ISBN-13: 9781107078239, 2015.
[3] Stationary Process, https://en.wikipedia.org/wiki/Stationary_process.
[4] Trend Stationary, https://en.wikipedia.org/wiki/Trend_stationary.
[5] T. N. Minh, T. Nam, D. H. J. Epema, Parallel Workload Modeling with Realistic Characteristics, IEEE Transactions on Parallel and Distributed Systems, pp. 2138-2148, vol. 25, 2014.