INFORMATICA, 1998, Vol. 9, No. 4, 425–436

Forecasting Automation: an Emerging Branch of Forecasting Engineering

Andrey KHARCHENKO
Glushkov Institute of Cybernetics
40 Akademik Glushkov ave., 252022 Kiev-22, Ukraine
e-mail: olgal@olgal.pp.kiev.ua

Received: December 1997

Abstract. Principles of the framework called time series *forecasting automation* are presented. It is required in processing massive temporal data sets and creating completely user-oriented forecasting software where manual data analysis and a user's decision-making is either impractical or undesirable. Its distinct features are local extrapolation models, their active training, criterion of model performance assessment used in adding new examples to the model training set and in deciding on which one of a group of competing models consistent with the common training set performs best. A generalized algorithm for local model tuning on massive data series that can be run without human intervention is presented.

Key words: forecasting, forecasting automation, massive time series, model scoring, time series analysis.

1. Introduction

Time series are an important class of data objects. They arise in financial and scientific applications, e.g., stock price indices, volume of product sales, telecommunications data, audio data, environmental measurement sequences, etc.

In a typical forecasting problem, there are several important steps:

1) selecting informative inputs (in case of multivariate data) and preprocessing them if needed (noise reduction, outlier detection, etc.);
2) identifying models applicable to the data;
3) tuning the applicable models;
4) selecting the model which is the best in forecasting test data or a group of best models to produce averaged result.

Until recently tuning models to data was rarely done in a uniform way: after carefully eyeballing data, analysts devised tricky techniques to build good models (to be primarily accurate) by incorporating basic extrapolation models with methods (very specific to the concrete data they dealt with) for treating seasonality present in data, suppressing outliers, cleaning out noise, etc. Due to poor computation resources encouraging the practice of "touching data with fingers", analysts had to train themselves in partly computer-aided, however altogether manual, model preparation. The most important issue in those times

was that all needed reasoning (e.g., which of several "competing" models to choose) was performed perfectly: people did it themselves. The task so far was not difficult: human intuition helped software process short series. Makridakis and Wheelwright (1978) and Makridakis *et al.* (1984) gave excellent examples of this.

The event was of the great importance in the time series community – the so-called M-competition (Makridakis *et al.*, 1984). Its aim was to determine which of 24 popular forecasting models are best in extrapolating 1001 series from the economy and finance domain. The competition revealed no champion(s); its most important conclusions were: (i) there is no model a priori best in forecasting all of the series and (ii) there is no way to know a priori which of two models X and Y will produce better prediction. (Criteria of prediction goodness were still being discussed then.) It goes without saying that model preparation in the competition was performed manually thus involving a great deal of human insight. Later, another group of researchers (Poulos *et al.*, 1987) performed a similar competition to test among others *automatic* procedures of identifying and tuning some models. The experiments showed that they produced almost as accurate predictions as those done by human experts. To summarize, at the dawn of the industry the problem was how to obtain useful data from short historical series and to build models as accurate as possible. However, there has been developed many models and useful statistical estimators.

Emergence of massive historical databases caused new problems. Tuning even a single model became very time-consuming and performing the above four-step forecasting procedure cannot be completed in acceptable time at all. Furthermore, "elder" part of the dataset included in a model's training set worsens forecasting models (despite from the classical statistical viewpoint that larger training sets are preferable well applicable to non-massive temporal data, see Fig. 1 as the dataset's later underlying process is somewhat different from earlier one. But owing to huge size of data, manual analysis of it to tackle these new problems is nearly always impractical. Machines learning also faced massive data and a number of interesting techniques to treat them have been devised by now. Of primary importance to these forecasting problems are *active learning* and *automated learning*. Briefly, contrary to passive learning in which the learner is unable to influence its training set, active learning provides the learning algorithm with some control over which data items are used in training (Cohn *et al.*, 1996). Finally, the notion of *goodness* of time series prediction has now reached its maturity.

In data mining and machine learning communities, a number of activities of automated processing massive data remind the necessity of developing their analogy in forecasting temporal data. To quote only a few, a scientific data set navigation and visualization system being developed at Carnegie Mellon University (Pittsburgh, USA) uses the autopilot principle to automatically seek out "interesting" local subsets and "fly" to them (Mockus *et al.*, 1996). The framework that can be called *forecasting automation* must operate in the highly constructive way peculiar to most of modern massive data analysis software. The latter promptly provides a user with initially rough results being continuously updated as the system processes more data, and makes the results accessible during all the process of their perfection.

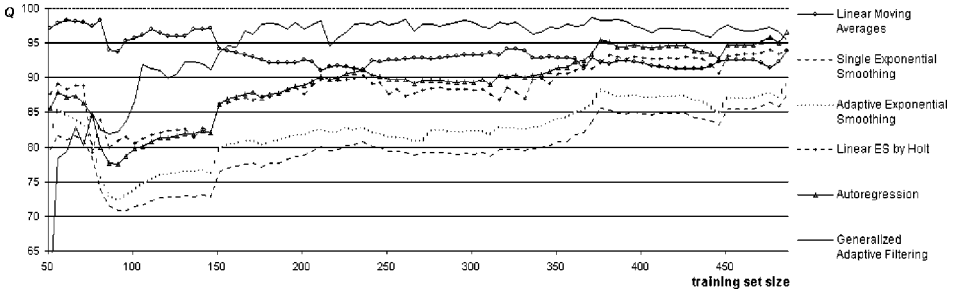


Fig. 1. Performance of various models (axis Q) as functions of their training set size in processing non-massive series.

2. Local Tuning Time Series Models

2.1. *Data and Models*

Time series being primarily considered in this paper are those consisting of thousands observations, such that often fitting even a single model on it can take unacceptable long time. Given a vector of n observations $Y = \{y_1, \ldots, y_n\}$ taken at discrete time intervals, we can apply the following models implemented in a typical forecasting system to extrapolate it:

1. regression models (univariate, multivariate, stepwise);
2. moving averages (Single Moving Averages, Linear Moving Averages);
3. exponential smoothing (Single Exponential Smoothing, Adaptive Exponential Smoothing, Linear ES by Holt, Linear ES by Brown, Quadratic ES by Brown, Seasonal ES by Holt-Winters, Additive Seasonal ES by Winters, Seasonal ES by Brown-Harrison);
4. ARMA and ARIMA models (Autoregression, Generalized Adaptive Filtering, ARARMA);
5. Other types of models, e.g., neural networks.

Some of the models, e.g., neural networks, can treat also *multivariate* data. It demands an extra step before the model tuning for analysis, e.g., search of indicators and determining their lags, but it does not require specific treatment in case of very extensive data. Makridakis and Wheelwright (1978) give a review of multivariate data problems and analysis techniques that can be performed in the automatic way.

Denote the set of implemented (possible) models as M, and the set of ones applicable to the given vector Y as $M_A,\ M_A = M_A(Y),\ M_A \in M$. The set M_A here is analogous to the *version space* in machine learning.

2.2. *Model Scoring and Training Set Enlargement*

To provide applicable models with better performance (particularly to make them more general) their training set should be gradually enlarged with historically earlier observations as they are obviously less valuable according to the active learning concept. The

concept requires determining a rule of the learner's querying new examples that will be added to the training set. Despite there are a number of machine learning criteria to perform the querying, the general scheme of acquiring new examples for the model training set in time series forecasting being proposed is as follows: *examples are taken in the retrospective order until their model performs unsatisfactorily*. To be used, this scheme needs a technique to measure performance of a prediction model. Primarily, apart from the active learning issues, the technique is required in deciding which of two competing models consistent with their common training set performs better.

Denote the training vector of a model from M_A as Y_L, and its test vector as Y_T. All further considerations are based on the following their mapping to Y : $Y_L = \{y_i, \ldots, y_p\}$, $Y_T = y_i, \ldots, y_p, \ldots, y_n$ and $\{y_{p+1}, \ldots, y_n\}$ is the short sub-vector of Y_T comprised by observations which are not used in training a model but testing it only, hence $Y_L \subset Y_T \subset Y$. Denote the vector resulting from modeling the vector Y_T as $\widehat{Y}_T = \{y_i, \ldots, y_n\}$.

A natural treatment of the case when tuning m on $Y_L = Y$ (*global* fit) cannot be completed in acceptable time is using as Y_L a short sub-vector of Y which makes such a tuning possible. In other words, *local* fit of the model is necessary. Having at least two competing models (comprising M_A) we face a problem of selecting the best one further referred to as the *best fitted model* (BFM) to serve as a current model of data in an automated system. Generally, there are several approaches to model scoring and separation:

- Analysis of correlation between Y_T and $\widehat{Y}_T$ produced by each competing model, comparing their accuracy, e.g., the MSE, and the like,
- Comparing the estimate of each model's accuracy and its complexity: Minimum Description Length (Rissanen, 1987) and use of information criteria (Akaike, 1973 and 1974) being widely used in pattern recognition for model analysis.

However, to use the approaches one should take into account the specificity of time series: along with (i) maximizing the generality (as in machine learning) there is another goal – (ii) requirements to residual autocorrelation specific to forecasting. The latter motivates one more approach:

• Analyzing an autocorrelation statistic of residuals each model yields.

Additionally, the pure forecasting view of model performance is that ideally residuals it yields should be a white noise process. However, the straightforward idea of measuring a residual series' proximity to white noise is rather hard to implement.

Insignificant residual autocorrelation indicates that a model covers all the series' underlying processes, and vice versa – high autocorrelation is the indication of a structure in residuals not reflected by the model series (Holden *et al.*, 1990, and Makridakis and Wheelwright, 1978). It is of primary importance to short range forecasts.

Thus, in scoring a prediction model m of a time series tested on the vector Y_T, the *model quality* functional should cover all their three basic aspects (accuracy, complexity, and residual autocorrelation control):

$$Q(m, Y_T) = Q^*(A, IC, AC), \tag{1}$$

where A is a measure of accuracy, $A = A(Y_T, \widehat{Y}_T)$, IC is a value of an information criterion statistic, AC is a value of an autocorrelation statistic.

The Durbin-Watson's statistic (Durbin and Watson, 1950) is approximately normally distributed with mean 2 and has values in the range [0,4]:

$$d = \frac{\sum_{i=2}^{n}(e_i - e_{i-1})^2}{\sum_{i=1}^{n} e_i^2}, \tag{2}$$

where $e_i = y_i - \widehat{y}_i$. Residual covariance grows as d deviates from 2. AC can be calculated as the proximity of d to the center of its distribution: $AC = DW$ where

$$DW = 1 - |d - 2|/2. \tag{3}$$

Motivated by practical considerations only, better separation of competing models can be reached by means of a model quality estimate reinforced with account of the correlation coefficient $R_{Y_T \widehat{Y}_T}$ of the vector Y_T and its prediction $\widehat{Y}_T$:

$$Q(m, Y_T) = Q^*(A, IC, AC, R_{Y_T \widehat{Y}_T}). \tag{4}$$

An information criterion statistic is generally a likelihood function penalized with model complexity, e.g.,:

$$FPE = \frac{n+k}{n-k} - \frac{\sigma^2(n-k)}{n} - \text{the final prognostic error (Akaike, 1970)},$$
$$AIC = n \log \sigma^2 + 2p - \text{Akaike's information criterion (Akaike, 1973)},$$
$$HQ = n \log \sigma^2 + 2k \log \log n - \text{Hannan-Quinn's statistic (Hannan and Quinn, 1979)},$$
$$BIC = n \log \sigma^2 + k \log n - \text{Bayes' information criterion (Geweke and Meese, 1981)},$$

where n is the number of observations in a model's test vector, σ^2 is the standard deviation of residuals, and p is a model's parameter count.

But in an all-purpose forecasting system, M_A can contain models from different "families", e.g., regressions and ARIMA-like models, therefore direct comparison of their quality values calculated as functions of IC is incorrect. The set of models M_A applicable to the series Y can be represented as $\bigcup_{i=1}^{f} M_i$ where f is the number of model families implemented by the system and M_i is the subset of the i-th family applicable to Y. It dictates a two-phase procedure of determining the BFM:

1) determine the BFM among models within one (i-th) family which are applicable to $Y - BFM_i = m_j \in M_i \mid m_j \in M_A \wedge \max_j Q(m_j, Y_T)$;
2) determine the BFM of all the families – $BFM = BFM_i \mid \max_i Q_2(BFM_i, Y_T)$ where

$$Q_2(m, Y_T) = Q_2^*(A, AC, R_{Y_T \widehat{Y}_T}). \tag{5}$$

A shortcoming of this technique is that models can be ranged within M_i but not within M_A.

Local tuning brings about a problem of determining the length of local training vector Y_L. Its right bound y_L as well as the minimum training vector length of each family models $L_{\min} = \min_i L_{\min}(M_i)$ is known.

Similar to neural network models, time series models are the more general the more extensive data they are trained on. Therefore, starting from some initial state, incremental enlargement of their local training vector $Y_L = \{y_{L1}, \ldots, y_{Lp}\}$ with some reasonably small portion of items of very large Y to the left from y_{L1} leads to the whole vector's greater generality until training takes unacceptable time.

3. Algorithms

3.1. *Tentative Algorithm*

The *tentative algorithm* insures that the current model of vector Y represented with its training vector Y_L (being incrementally enlarged as well as the test vector Y_T) is BFM of the latter. It repeatedly performs BFM detection (which remains permanently accessible to the user) and training vector enlargement until an event *time-out* alarms the iteration. Fig. 2 presents a typical fragment of three competing models' quality graph computed with the tentative algorithm.

Tentative Algorithm:

Notation:
i – index of vector item (integer);
s – step – size of training vector enlargement portion (integer)

$i = p - L_{\min}$;
while $(i > 0)$ **do begin**
  if (*time-out*) then
    break loop
  $Y_L = y_i, \ldots, y_p$;
  $Y_T = \{y_i, \ldots, y_n\}$;
  $M_A = M_A(Y_L)$;
  $BFM = BestFittedModel(M_A, Y_L, Y_T)$;
  $i = i - s$
end.

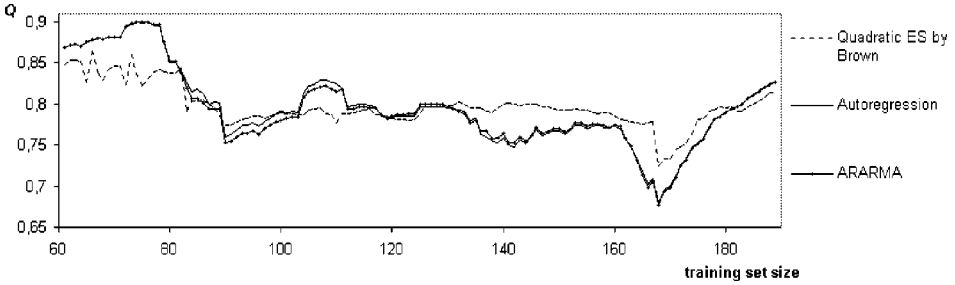


Fig. 2. Tentative algorithm illustrated.

Algorithm behind the function 'BestFittedModel(M_A, Y_L, Y_T)*'*:

Notation:
M_A – parameter, the set of models applicable to the test vector (Y_T);
Y_L – parameter, the training vector;
Y_T – parameter, the test vector;
M_F – set of all possible model families;
M_j – any single family of models $\bigcup M_j = M$;
m – any single model;
BFM_j – best fitted model within any single family M_j;

forall $(M_j \in M_F)$ **do begin**
  forall $(m \in M_j$ **and** $m \in M_A)$ **do begin**
    fit m *on* Y_L;
    calculate $Q = Q(m, Y_T)$;
    store $m,\ Q$
  end
  $BFM_j = m|\max Q$;
  calculate $Q_2 = Q_2(BFM_j, Y_T)$;
  store BFM_j, Q_2
end
result $= BFM_j|Q_2$.

Performing model competition for each state of Y_L insures its timely change. However, if the portion of enlargement (s) is small the whole algorithm proceeds too slow because model training being the most time-consuming operation is being done too frequently. This shortcoming can be easily treated by retaining the BFM detected in i-th iteration during a number of next periods of training vector growth.

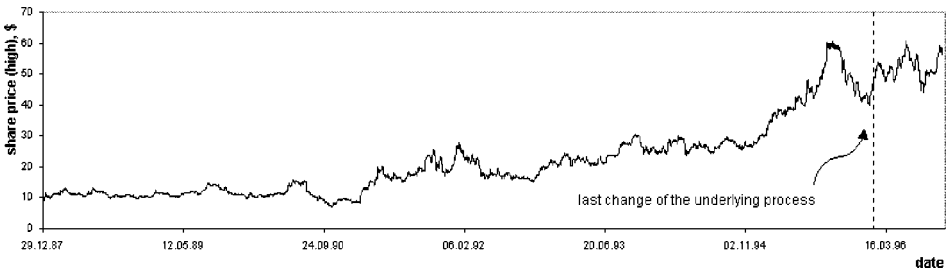


Fig. 3. Real world data: trend part of a share price series followed by stationary part.

3.2. *Refined Algorithm*

Often a very long series have two parts $Y_1 = \{y_h, \dots y_t\}$ and $Y_2 = \{y_{t+1}, \dots, y_n\}$ having different trends, seasonality or so. In its turn, Y_2 sometimes can be similarly subdivided into two parts, anf so forth. After a model was trained on Y_2, extending its training vector with Y_1 having different underlying process will worsen predictive features of the model, especially for short-range forecasting. Fig. 3 shows a graph of maximum daily share price of some stock company taken from 31.12.1987 to 30.9.1996 with clear last sub-series having distinct seasonality and trend. In other words, similarly to the observed value y_i that changes over time, its underlying process (to model) also changes over time. Obviously the latter can be analyzed by examining the model's performance as function of the training set size.

In the *tentative algorithm*, the incremental enlargement of the current BFM m_b's training vector Y_L should be stopped when the value of its quality $Q(m_b, Y_T)$ becomes lower than some critical level (denote it Q_{cr}). The value of Q_{cr} is such that it is likely that after training another model m_a on that Y_L:

$$Q(m_b, Y_T) < Q_{cr} \Leftrightarrow Q(m_a, Y_T) > Q(m_b, Y_T). \tag{6}$$

The *refined algorithm* implements this principle.

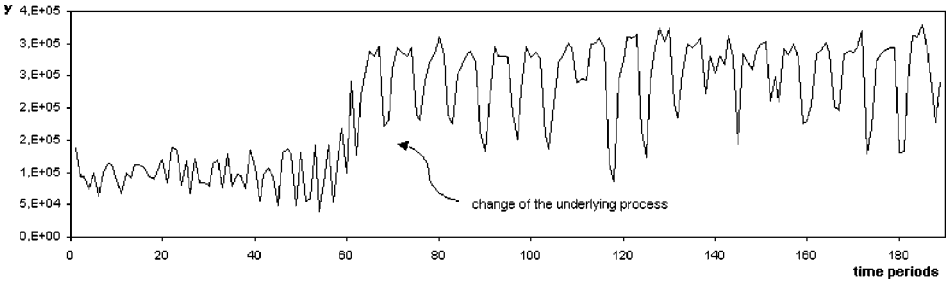


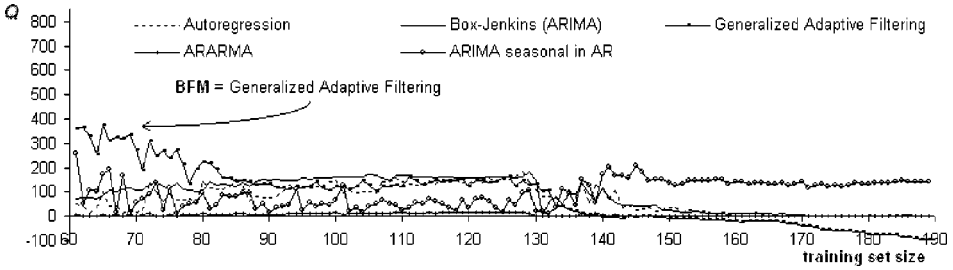
Fig. 4. Part of a series (last 190 observations) containing change of the underlying process to serve as the test series for the refined algorithm.

Refined Algorithm:

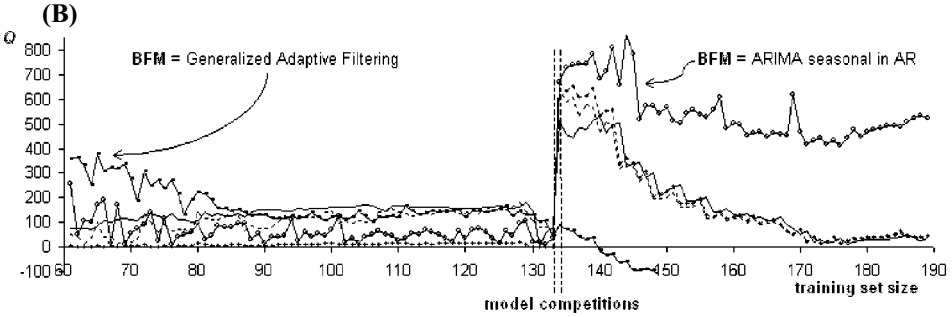
$i = p - L_{\min}$;
$Y_L = \{y_i, \ldots, y_p\}$;
$Y_T = \{y_i, \ldots, y_n\}$;
$M_A = M_A(Y_L)$;
$m = BestFittedModel(M_A, Y_L, Y_T)$;
$Q = Q(m, Y_T)$;
while $(i > 0)$ **do begin**
  if ***(time-out)*** **then**
    break loop
  $i = i - s$;
  $Y_T = \{y_i, \ldots, y_n\}$;
  $Q = Q(m, Y_T)$;
  if $(Q < Q_{cr})$ **then**
  begin
    $Y_L = y_i, \ldots, y_p$;
    $\boldsymbol{m} = BestFittedModel(M_A, Y_L, Y_T)$
  end
end.

The value of Q_{cr} depends on the functional Q^* (3) and the data being processed. For example, with A computed as the cumulated absolute percentage error (*CAPE*), $CAPE = \sum_{i=1}^{n} \frac{(y_i - \widehat{y}_i)}{y_i} \cdot 100\%$, for Q_{cr} to alarm 900% cumulated error it should be set equal to 100, and model performance measured as $Q = DW \cdot R_{Y_T \widehat{Y}_T} \cdot (\alpha - CAPE)$, $\alpha = 1000\%$. Fig. 5 illustrates the refined algorithm processing the test series shown in Fig. 4 particularly changes of BFM each time $Q < Q_{cr}$ (part B) compared with the same models' performance without retraining when it is necessary (part A).

(A)



Performance of competing models once trained on 60 observations and then tested on the vector being repeatedly enlarged with earlier observations. The model initially performing best (BFM) is 'Generalized Adaptive Filtering'.



The same models re-trained and re-evaluated (dashed lines mark points of competitions). Each time performance of the current BFM gets lower than the threshold $Q_{cr}(= 100)$ new BFM trained on larger set is being selected.

Fig. 5. Refined algorithm processing series in Fig. 4 illustrated.

The algorithm works in two modes:

• *express*. In this mode, the algorithm proceeds enlarging the training vector Y_L while quality of current BFM trained on it (denoted as Q) remains at satisfactor level (greater than Q_{cr}). Samples to be attached to the training vector after $Q > Q_{cr}$ can "spoil" the model therefore further enlargement of Y_L is finished. Otherwise the algorithm proceeds until there are no samples in Y to process. Given some Q_{cr} the algorithm at the end advises a user the BFM as the model of the whole vector Y. Current BFM is available to the user all the run time.

• *time-out*. If alarming is enabled working in the previous mode is eventually interrupted also when a system level event of the type time-out stops it.

Further improvement of the algorithm can be made if the portion s of vector Y_L enlargement being constant were a function of Q calculated in the preceding test. Namely, s calculated as a function of quality differentiated, e.g., $s_t = \alpha \exp(\beta q_{t-1})$ so that it could sense the behavior of Q.

4. Conclusion and Future Work

Features of an emerging branch of forecasting software engineering – forecasting automation were presented. The key problems of creating the forecasting automation software are automatic model identification and fitting, processing massive series, and techniques of model estimation to support it. An algorithm for local stepwise model identification and tuning to be applied to large temporal databases was presented.

Sometimes a database can store only a small portion of a very dense data flow therefore historically early data cannot be used to fit extrapolation models. Even if a database were unlimited in size, the task of fitting a single model with desired generality on its contents would be infeasible as it would take a lot of time to be completed. However, little computation still could be performed in the intervals between entering subsequent

data portions. The idea is to store in the database very few specific data only that would make it possible to optimally train models of the whole data flow. Guyon, Matic, and Vapnik (1996) proposed a technique for detecting informative patterns thus cleaning data and showed the point of optimum cleaning that can be presented by the Vapnik-Chervonenkis theory (Vapnik, 1982). An automation technique to detect the above-mentioned specific observations as most surprising ones taking into account that forecasting software never implements a single model but many of them, is a challenge for future research. The fact that procedures of prior data validation usually remove surprising observations as outliers is an additional obstacle.

5. Acknowledgments

I am very grateful for Alexander Stupak for his assistance in preparing this paper and to Vadim Tulchinsky for critical reading it and providing valuable comments. Special thanks to Albertas Caplinskas for his supportive efforts and encouragement.

References

Akaike, H. (1970). Statistical predictor identification. *Annals of the Institute of Statistical Mathematics*, **22**, 203–217.

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In Pertov B.N. and F. Csaki (Eds.) *Second International Symposium on Information Theory*, Budapest.

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Trans. Aut. Control*., **19**, 716–723.

Box, G.E.P., and G.M. Jenkins (1970). *Time Series Analysis Forecasting and Control*. Holden-Day, Inc.: San Francisco.

Cohn, D.A., Z. Ghahramani, M.I. Jordan (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, **4**, 129–145.

Durbin, J., and G.S. Watson (1950). Testing for serial correlation in least squares regression I. *Biometrika*, **37**, 409–428.

Durbin, J., and G.S. Watson (1951). Testing for serial correlation in least squares regression II. *Biometrika*, **38**, 159–178.

Geweke, J., and R. Meese (1981). Estimating regression models of finite but unknown order. *International Economic Review*, **22**, 55–70.

Guyon, I., N. Matic and V. Vapnik (1996). Discovering informative patterns and data cleaning. In U.M. Fayad *et al.* (Eds.), *Advances in Knowledge Discovery and Data Mining*. Cambridge: AAAI Press.

Hannan, E.J., and B.G. Quinn (1979). The determination of the order of an autoregression. *Journal of the Royal Statistical Society, Series B*, **41**, 190–195.

Holden, K., D.A. Peel and J.L. Thompson (1990). *Economic Forecasting: an Introduction*. Cambridge: Cambridge University Press.

Kharchenko, A.V. (1997). Estimating and comparing forecasting models in decision making systems. In *Proceedings of 1997 Knowledge-Dialog-Solution Conference*, Yalta, Ukraine (in Russian).

Makridakis, S., and S.C. Wheelwright (1978). *Interactive Forecasting*, Second Ed. Holden-Day, Inc: San-Francisco.

Makridakis, S., A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, S. Newton, E. Parzen and R. Winkler (1984). The accuracy of extrapolation (time series) methods: results of a forecasting competition. *Journal of Forecasting*, **1**, 111–153.

Mockus, A., W.E. Eddy, S.Y. Chang and K.R. Thulborn (1996). Software for the visualization of fMRI data. In *Proceedings of the International Society for Magnetic Resonance in Medicine Fourth Scientific Meeting and Exhibition*.

Poulos, L., A. Kvanli and R. Pavur (1987). A comparison of the accuracy of the Box-Jenkins method with that of automated forecasting methods. *International Journal of Forecasting*, **3**, 261–268.
Rissanen, J. (1987). Stochastic complexity. *Journal of the Royal Statistical Society, Series B*, **49**(3), 223–239 and 252–265.
Vapnik, V. (1982). *Estimation of Dependencies Based on Empirical Data*. New York: Springer Verlag.
Wasserman, P.D. (1993). *Advanced Methods in Neural Computing*. Van Nostrand Reinhold: New York.

A. Kharchenko graduated from Chernigov Institute of Technology in Ukraine in 1994. He joined Glushkov Institute of Cybernetics where he is currently a post-graduate student. His research interests include databases, knowledge discovery and data mining.

Prognozės automatizavimas – nauja prognozės inžinerijos šaka

Andrey CHARČENKO

Pateikiami laiko eilučių prognozės automatizavimo principai, kurie reikalingi tvarkant didelius laikinų duomenų masyvus, kai vartotojo sprendimai yra arba nepraktiški, arba nepageidautini.

Išskirtinės automatinės prognozės savybės yra lokalūs ekstrapoliacijos modeliai, jų apmokymas ir sprendimas, kurie iš konkuruojančių modelių veikia geriau. Pateikiamas lokalaus modelio derinimo be žmogaus įsikišimo didelės apimties duomenų sekoje apibendrintas modelis.