# Using Data Mining Techniques for Detecting Noises and Pre-Processing Financial Time Series

**Carson K.-S. Leung**[1,2] **Ruppa K. Thulasiram**[1,2] **Dmitri A. Bondarenko**[1]

[1]Department of Computer Science, The University of Manitoba, Winnipeg, MB, Canada
[2]Institute of Industrial Mathematical Sciences, The University of Manitoba, Winnipeg, MB, Canada

## Abstract

In this paper, we propose a system to detect noises and to pre-process financial time series. This novel system combines a statistical algorithm with a data mining algorithm. We implemented and tested both algorithms on real-life historical financial time series consisting of security prices with outliers. We observed the strengths and weaknesses of each of the two algorithms, and then developed a hybrid algorithm to overcome the weaknesses of the two algorithms. Consequently, the resulting (processed) datasets can be used as input for models used in forecasting future security prices and in predicting future market behaviour.

**Keywords:** Financial data mining, financial time series forecasting and analysis, data pre-processing, outlier detection, computational intelligence (knowledge discovery) in finance.

## 1. Introduction and Motivation

*Computational finance* is an emerging cross-disciplinary area of research that solves problems in finance/business using advanced scientific computing techniques. Data mining in finance is a new breed of problems within computational finance that serves to solve larger problems such as option pricing and portfolio management.

In general, data mining refers to the search for implicit, previously unknown, and potentially useful information or patterns that might be embedded in data. It is known that a rare event could be an indication of some unusual, suspicious, or criminal activities. So, an important data mining task is *outlier detection* [1, 3, 4, 5, 7, 8], which aims to analyze and find these exceptional activities from datasets like the performance statistics of professional athletes, workers' compensation data, and medical test data. Moreover, outlier detection could be used in other application areas such as finance, electronic commerce, and Web. In this paper, we show how outlier detection can be applicable to computational finance.

An important area in computational finance is quantitative research. Several financial models have been developed to forecast future security prices and to predict future market behaviour. These models usually rely on standardized historical data, and are sensitive to data variations. *Any unusual noises present in the data may lead to incorrect forecast or prediction.* To ensure good prediction of price behaviour, many financial models require large amounts of historical price data. Hence, it is not uncommon to use daily prices (e.g., opening, bid, and/or ask prices) for a period of at least 10 years for hundreds (and often thousands) of securities. This calls for efficient methods to handle large databases.

Similarly, to ensure accuracy in financial applications like the option pricing and risk management, financial models usually require the use of price series that are free from noises. Although the price series are normally obtained from financial data retailers (e.g., Bloomberg) who are generally very reliable, the series might contain some *data polluters*. Data polluters can be caused by (a) missing data, (b)short-lived sudden price changes, and (c)permanent sudden price changes.

Noises/data polluters can significantly influence financial model outputs (i.e., the forecast of future security prices, the prediction of future market behaviour). So, to achieve better results, we need a system for detecting and eliminating noises as well as pre-processing the data. More specifically, algorithms in such a system for this application in finance should satisfy the following three requirements:

(a) It should run efficiently on large datasets.
(b) It should eliminate noises caused by missing data and those short-lived sudden price changes.
(c) It should ignore permanent sudden price changes.

In the past few years, some studies [6, 8] suggested methods to locate noises from financial time series. However, these studies mainly dealt with portfolio selection, but *not* focused on outlier detection. Moreover, these studies stated that the absence of noises/data polluters in the data was often a critical condition for their models to successfully work, but they did not mention how to remove the noises. In addition to these studies, there are some other dispersed studies reported in the finance literature on various specific outlier detection problems. For example, del Hoyo and Llorente [2] presented a

framework for a model specification search and its usage in linear regression models. An important issue identified in their models is that conducting inference without properly considering the sequential selection process, which is commonly referred as data snooping [9], can be *extremely misleading*. Moreover, these forecasting models should accommodate the permanent sudden price changes—say, by *not* considering them as noises.

Therefore, in this paper, we effectively employ data mining as well as statistical algorithms to the problem of noise detection in financial time series. Our ***contribution of this paper*** is the development of a hybrid system for detecting and removing noises from financial time series. More specifically, our *technical contributions* of this paper are as follows:

- We apply a *data mining (outlier detection) algorithm* to detect outliers from time series on security prices.
- We develop a new *statistical algorithm,* which uses distribution properties of data to detect outliers.
- Due to their varying nature and properties, the two algorithms (the data mining and the statistical algorithms) detect different outliers from the time series. However, *they are complementary*. When both algorithms were combined into our hybrid system, almost all noises can be effectively detected and removed.

Consequently, the resulting time series can be used as input to many existing financial models for a more accurate forecast of future security prices, a more accurate prediction of future market behaviour, and more accurate computation of option prices.

To summarize, the hybrid system we proposed in this paper uses both data mining and statistical algorithms to detect noises and to pre-process financial time series. It is interesting to note that, in this paper, we use the outlier detection algorithm as a ***pre-processing step*** for forecasting models. For example, the output of our system (i.e., the "pre-processed" financial data) could be fed as input for other forecasting models (e.g., neural network architecture) to predict future prices. This is quite different from the traditional role of many existing data mining (outlier detection) algorithms, which often serve as *stand-alone tools* for obtaining insight into data distribution!

# 2. Our algorithm: RemoveOutliers

Our proposed system consists of two stages. Stage I identifies the missing prices in a given financial time series, and substitutes them with new prices that are consistent with their neighbouring prices. By so doing, we do not introduce any "artificial" outliers/noises. Stage II uses two techniques (data mining and statistical approaches) to detect those outliers caused by short-lived sudden price changes. We then combine the two techniques into a new and improved algorithm, which nullifies their individual weaknesses.

## 2.1. Stage I: Eliminating Missing Data

The goal of Stage I of our system is to identify gaps (i.e., the missing data/prices in the time series) and to fill them with new prices. With this respect, we develop an algorithm to identify gaps. The algorithm is based on the fact that there are five business days every week, and data for each business day should be available regardless of whether it is a holiday or not. It has a linear complexity with a single scan of the dataset. The algorithm scans the whole dataset once, and divides items on a weekly basis. If a price item for a certain weekday is missing, it is substituted with an item generated by a function, called **NewPrice**, which calculates (a) an approximate value for the missing item or (b) a new value of an outlying item in the time series. The complexity of this function is linear with respect to the size of interval used for calculating the average. Here, we make two realistic assumptions. First, items are normally distributed. Second, items that are close together (based on the date) influence each other to a greater extent than the items that are far apart. Then, in abstract terms, the new price can be computed based on the following equation:

$$\text{NewPrice} = \max\{\textstyle\sum_{i=1}^{m} I_{t-i}\, w_{ui}, 0\} / 2n + \max\{\textstyle\sum_{i=1}^{n} I_{t+i}\, w_{di}, 0\} / 2m$$

where $I$ is an item in the time series, $w_{ui}$ are the weights of the preceding (upstream) items and $w_{di}$ are the weights of the following (downstream) items; $m$ and $n$ are the numbers of neighbouring data items in the upstream and downstream directions.

## 2.2. Stage II: Detecting Short-Lived Sudden Price Changes

Once the missing prices are identified, we can apply Stage II of our system to detect outliers (i.e., the outlying prices in the financial time series). For this stage, we employ a data mining algorithm, develop a new statistical algorithm, and then efficiently combine them into an enhanced hybrid algorithm.

The first algorithm in Stage II is a data mining algorithm, called **RemoveOutliersDM**, which is based on the distance-based outlier detection algorithm Find-AllOutsM [3]. The key idea of our RemoveOutliers-DM algorithm can be described as follows. An item is considered an outlier if it has less than $M$ neighbours within certain distance $D$. Here, the financial time series can be represented in a 2-dimensional space with prices along the $y$-direction and time along the $x$-direction. We divide the space between the maximum

and minimum item price values of the dataset into $K$ equal intervals along the $y$-axis, where $K$ is a user-specified constant and the size of each interval is (MaxPrice – MinPrice)/$K$. The space between the first and the last dates is divided into intervals of size $L$ along the $x$-axis. The distance that defines the neighbourhood for each item is equal to $D$ = (MaxPrice – MinPrice)/$K$ * $2\sqrt{2}$. Then, each item is quantized into this 2-dimensional space. The algorithm has a *linear* complexity with respect to the number of items in the dataset and the number of cells in the 2-dimensional space.

We will show in Section 3 that our data mining algorithm is effective in detecting outliers (especially in detecting those lie away from the normal price range) with experimental results.

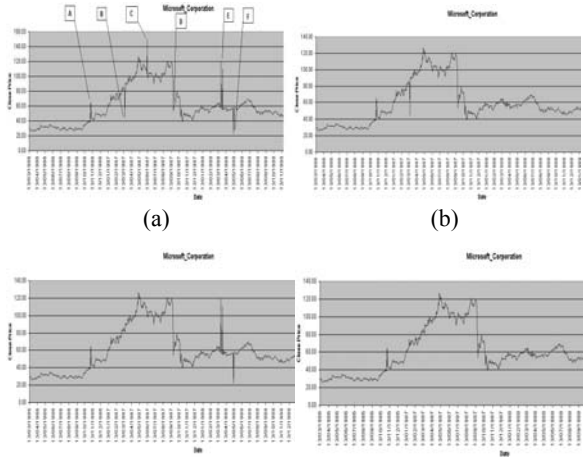
The second algorithm is an in-house developed statistical algorithm, called **RemoveOutliersStat**. Here, we make an assumption that items are normally distributed. This assumption usually holds good for many short continuous sub-groups in the dataset, where our focus of outlier detection lies. The algorithm applies a statistical observation that most items are located within three standard deviations from the mean/average. Thus, if an item is ten standard deviations away from the mean, it is very likely to be an outlier. The distance that measured in standard deviations from the mean is defined as *item ranking*. Each item $I_t$ has an upstream ranking (which is based on the items preceding $I_t$) and a downstream ranking (which is based on the items that following $I_t$). In order to be considered as an outlier, an item $I_t$ needs to have both rankings greater than some specified thresholds. To improve efficiency, we avoid calculating the mean for each item from scratch; instead, we use the moving averages. Since the algorithm works in a sequential manner, the averages (i.e., the means) for the current item can be calculated by adjusting the averages for the upstream items. The complexity of this algorithm is *linear*, and it requires only a single scan of the entire dataset. The items that are marked as outliers are replaced with new items generated by the aforesaid NewPrice function.

# 3. Experimental Results

We ran our system over sets of real (historical) security price time series obtained from the Yahoo! Finance website. To test the performance and effectiveness of our system, we added some "artificial" outliers to these time series.

Our system was implemented using C++ (for the algorithms) and Excel with Visual Basic for Applications (for the interface). We experimented with various datasets. Results on these datasets are consistent. So, for lack of space, we describe in this section the effectiveness of our algorithm through the experimental results of three algorithms—namely, the data mining algorithm called RemoveOutliersDM, the statistical algorithm called RemoveOutliersStat, and their hybrid algorithm called **RemoveOutliers-Hybrid**—only for Microsoft dataset.

In the dataset for Microsoft Corporation shown in Fig. 1(a), several important items are worth special mentioning. For example, $A$ is a set of two outlying items that are very close to each other; the two items are not too far away from the normal items in both $x$- and $y$-directions. $B$ is a single outlier that is also fairly close to those normal items in both $x$- and $y$-directions. The uniqueness of outliers $A$ and $B$ is that their values do not fall outside of the normal range of the dataset. (There are normal items not too far away from these outliers that have similar values.) $C$ is a single outlier that is completely outside of the normal range of the time series. $D$ is *not* an outlier, and it is just a point where the stock price suddenly drops (due to some natural factors such as a change in economic situation or a stock split). $E$ is a set of two outlying items that are close to each other. Similarly, $F$ is also a double-itemed outlier. Although in that sense $E$ and $F$ are similar to $A$, there is a key difference. $E$ is at a sizable distance from any acceptable (normal) item, whereas $F$ is outside of the normal range of the time series. In contrast, $A$ is not too far away from the normal items.



(a) (b) (c) (d)

Fig. 1: Microsoft datasets

Fig. 1(b) shows the results of applying our data mining/outlier detection algorithm (**RemoveOutliers-DM**) to the time series for Microsoft Corporation. The results showed that our algorithm was able to successfully remove outliers $C$, $E$ and $F$, while leaving the non-outlier $D$ intact. An advantage of this algorithm was its effective removal of double-itemed outliers. This stemmed from the fact that the algorithm relied on the number of neighbours of a given data item rather than any other properties to determine an

outlier. Having only one or two neighbours would be a good indication of an outlier. However, the algorithm failed to identify outliers *A* and *B*, because these two outliers were within the normal range of the dataset and they had a large number of items in their surrounding neighbourhood; hence, the algorithm did not see them as outliers.

Fig. 1(c) presents the result of our statistical algorithm (**RemoveOutliersStat**). Here, the algorithm successfully removed outliers *B* and *C*, while leaving the non-outlier *D* intact. However, the algorithm failed to identify all double-itemed outliers *A*, *E* and *F*. It is because the algorithm used statistical methods based on averages and standard deviations. When there were several outlying items that are close to each other, they would influence each other's averages and standard deviations. This would lead to lower rankings of the individual items in the time series, and hence would fail to identify them as outliers.

Finally, we combined the above two algorithms into a hybrid algorithm (**RemoveOutliersHybrid**). As shown in Fig. 1(d), our hybrid algorithm successfully removed outliers *B*, *C*, *E* and *F*, while leaving the non-outlier *D* intact. (It should not be unexpected that *A* was not marked as an outlier because both algorithms failed to identify it. However, with high-order statistics, *A* is expected to be detected as an outlier.) The results showed that, while each algorithm had its strong and weak sides, the combination of both algorithms was the most effective approach for outlier detection in financial time series.

Effectiveness asides, all these algorithms were very efficient. For example, on average, they detected and removed noises from a file consisting of 31,600 price items in about 1.01 ± 0.21 seconds.

# 4. Conclusions

In this paper, we developed a hybrid system to detect and remove noises from financial time series. Our system consists of two stages. Stage I eliminates missing prices (data polluters) by replacing them with new prices. Stage II detects short-lived sudden price changes. For this stage, we implemented two algorithms (one data mining and one statistical).

In general, our data mining algorithm (Remove-OutliersDM) showed good performance and effectiveness. However, a drawback is that it might fail to detect an outlier if the difference between minimum and maximum values in the dataset is large. Another drawback is that its performance decreases when more outliers are introduced. Since financial data generally contains a lot of noises with many outliers, the effectiveness of this data mining algorithm is in need for improvement for large financial time series. To address this problem, we developed a new statistical algorithm.

Our statistical algorithm (RemoveOutliersStat) employed statistical approach that is more natural for time-series data. This approach showed excellent performance and effectiveness. However, a drawback of this algorithm is that it might fail to remove outliers that are next to each other.

Finally, we understood the weaknesses of the both of these algorithms and developed a hybrid algorithm, which was shown to be more effective in detecting outliers than each individual algorithm. *This paper shows a* ***confluence of various disciplines****—namely, data mining, statistics, and finance!*

Moreover, *this paper shows an additional applicability of outlier detection algorithms.* To elaborate, many existing outlier detection algorithms are generally served as stand-alone tools for obtaining insight into data distribution. In contrast, *our proposed outlier detection algorithm served as a pre-processing step for other algorithms* such as models for forecasting future security prices, for predicting future market behaviour, and/or for pricing complex financial instruments such as derivatives.

# 5. Acknowledgement

This project is partially supported by Science and Engineering Research Canada (NSERC) and The University of Manitoba in the form of research grants.

# 6. References

[1] M.M. Breunig et al. "LOF: Identifying Density-Based Local Outliers," *Proc. ACM SIGMOD*, pp. 93-104, 2000.

[2] J. del Hoyo and J.-G. Llorente. "Goodness of Fit, Stability, and Data Mining," *Computational Finance 1999*, The MIT Press, pp. 173-187, 1999.

[3] E.M. Knorr and R.T. Ng. "Algorithms for Mining Distance-Based Outliers in Large Datasets," *Proc. VLDB*, pp. 392-403, 1998.

[4] E.M. Knorr et al. "Robust Space Transformations for Distance-Based Operations," *Proc. ACM SIGKDD*, pp.126-135, 2001.

[5] C.K.-S. Leung. *Evaluation of Data Mining Opportunities at Workers' Compensation Board.* Research Report, Workers' Compensation Board of British Columbia, Canada, Nov. 1998.

[6] J. Park. "Modern Portfolio Theory and Its Application to Hedge Funds: Part II," *MFA Reporter*, pp. 1-2, 4, 12-13, Aug. 2001.

[7] E. Suzuki et al. "Detecting Interesting Exceptions from Medical Test Data with Visual Summarization," *Proc. IEEE ICDM*, pp. 315-322, 2003.

[8] M.-P. Victoria-Feser. *Robust Portfolio Selection*. Working paper 2000.14, University of Geneva, Switzerland, Aug. 2000.

[9] H. White. "Data Snooping with Confidence," *Proc. NNCM*, 1996.