Expert Systems With Applications 97 (2018) 60–69

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A novel data-driven stock price trend prediction system☆

Jing Zhang[a,*], Shicheng Cui[a], Yan Xu[a], Qianmu Li[a], Tao Li[b]

[a] School of Computer Science and Engineering, Nanjing University of Science and Technology, 200 Xiaolingwei Street, Nanjing 210094, China
[b] School of Computer Science, Florida International University, 11200 SW 8th Street, Miami, FL 33199, USA

ARTICLE INFO

Article history:
Received 8 September 2017
Revised 12 December 2017
Accepted 13 December 2017
Available online 13 December 2017

Keywords:
Feature selection
Morphological pattern recognition
Random forest
Stock price prediction

ABSTRACT

This paper proposes a novel stock price trend prediction system that can predict both stock price movement and its interval of growth (or decline) rate within the predefined prediction durations. It utilizes an unsupervised heuristic algorithm to cut raw transaction data of each stock into multiple clips with the predefined fixed length and classifies them into four main classes (*Up, Down, Flat*, and *Unknown*) according to the shapes of their close prices. The clips in *Up* and *Down* can be further classified into different levels reflecting the extents of their growth (or decline) rates with respect to both close price and relative return rate. The features of clips include their prices and technical indices. The prediction models are trained from these clips by a combination of random forests, imbalance learning and feature selection. Evaluations on the seven-year Shenzhen Growth Enterprise Market (China) transaction data show that the proposed system can make effective predictions, is robust to the market volatility, and outperforms some existing methods in terms of accuracy and return per trade.

1. Introduction

Stock price trend prediction is a classic and interesting topic that has attracted many researchers and participants in multiple disciplines such as economics, financial engineering, statistics, operations research, and machine learning. Although a lot of efforts have been paid during the past several decades (Abarbanell & Bernard, 1992; Adam, Marcet, & Nicolini, 2016; Adebiyi, Adewumi, & Ayo, 2014; Blume, Easley, & O'hara, 1994; Göçken, Özçalıcı, Boru, & Dosdoğru, 2016), accurate forecast of the stock price, even its movements, is still not easy to achieve hitherto, though some advanced machine learning techniques have been utilized. For instance, Kim (2003) used support vector machines to predict the direction of the daily socket price movements in Korea, obtaining a hit rate 56%. Schumaker and Chen (2009) included the text mining technique into socket price forecast, achieving a hit rate 57%. Tsai and Wang (2009) combined the decision tree and neural networks to make prediction to Taiwan stock market. The accuracy of the hybrid model achieves around 70%. However, their test data sets were relatively small, only including dozens of stocks. According to a recent empirical study (Gerlein, McGinnity, Belatreche, & Coleman, 2016), the prediction accuracies of several machine learning models (such as C4.5, K*, logistic model tree, etc.) are in the range of 48%∼54%.

Traditional technical analysts have developed many indices and sequential analytical methods that may reflect the trends in the movements of the stock price. However, technical analysis contradicts with the efficient-market hypothesis but they cannot make generalised inferences regarding the accuracy. For example, the efficient-market hypothesis states that as long as the market is weak-form efficient, the price of a stock follows the random walk model (Fama, 1995) and cannot be predicted by analyzing prices from the past. Meanwhile, the prices are affected by many macroeconomical factors, fundamental factors of companies and the involvement of public investors. Therefore, some criticism of technical analysis is that it only considers transactional data of stocks and completely ignores the fundamental factors of companies (Nassirtoussi, Aghabozorgi, Wah, & Ngo, 2014; Patel, Shah, Thakkar, & Kotecha, 2015) which might be helpful, if the market is in weak-form efficiency.

The fundamental factors of a company cover many aspects such as basic financial status, marketing and development strategies, political events, general economic conditions, commodity price indices, interest rate changes, movements of other stock markets, expectations and psychology of investors, and so on. Comprehensively figuring out the impact of these compound factors on the movement of the stock price is obviously out of the capability of human analysts. Researchers have begun to develop some text-mining based methods that can automatically analyze some

☆ The source code of the system is available at https://sourceforge.net/p/xuanwu/svn/.

* Corresponding author.
E-mail addresses: jzhang@njust.edu.cn (J. Zhang), qianmu@njust.edu.cn (Q. Li), taoli@cs.fiu.edu (T. Li).

https://doi.org/10.1016/j.eswa.2017.12.026
0957-4174/

of these fundamental factors (Nassirtoussi et al., 2014). For example, Schumaker and Chen (2009) extracted information from the breaking financial news to increase the accuracy of prediction. Bollen, Mao, and Zeng (2011) analyzed the mood of investors from twitter to reveal the sentiments of investors to some stocks. Ruiz, Hristidis, Castillo, Gionis, and Jaimes (2012) analyzed the correlations of financial time series from micro-blogging activities. Si et al. (2013) proposed a technique to leverage topic based sentiments from Twitter to facilitate the prediction of the stock market. Even the up-to-date deep learning techniques have been introduced to conduct event-driven stock market prediction, where events are extracted from news (Ding, Zhang, Liu, & Duan, 2015). However, the automatic fundamental factor analysis may be of some weakness. First, even though the messages or reports are released by the companies, public media or some third-party institutes, it still cannot be guaranteed that there is no misleading information. Second, it is not very clear how strong the correlation is between the released information and the stock price movement. Third, when the market is in semi-strong-form and strong-form efficiencies, the fundamental factor analysis even cannot bring excess returns (Timmermann & Granger, 2004).

Fortunately, in today's big data age, above issues could be bypassed, as a new train of thought, saying "let the data speak for themselves", has been proposed and drawn more attention. Unlike the information obtained from newspapers, micro-blogging and twitter, the everyday transaction data taking place in trade systems are absolutely realistic. The rapid development of machine learning provides a lot of new opportunities to utilize these transaction data to predict the trend of the stock price movement. In fact, applying machine learning to stock prediction has been studied for over thirty years. The early studies in 1990s mainly focused on using Neutral Networks to make prediction (Schöneburg, 1990; Zhang, Patuwo, & Hu, 1998), which partially refuted the validity of efficient market hypothesis (Lawrence, 1997). For example, Tsibouris and Zeidenberg (1995) utilized neural networks to predict stock price only based on past stock prices. The performance of these early methods usually was not good because of the size limitation of the neural networks. To address this issue, some recent studies resort to fusion or combination of models (Hadavandi, Shavandi, & Ghanbari, 2010; Tsai & Hsiao, 2010) and ensemble learning (Ballings, Van den Poel, Hespeels, & Gryp, 2015; Barak, Arjmand, & Ortobelli, 2017; Tsai, Lin, Yen, & Chen, 2011). All above studies have a common weak point that their practical availability is still questionable. In their studies, a small amount of carefully selected and labeled stock data were used to train and test models. Since the data do not cover all stocks and their movements in a stock market, the generalisation capabilities of the models are reduced in the real applications.

A real stock market carries out huge amount of transactions every day. We cannot expect that a real-world computer-aided decision system heavily relies on humans selecting and labeling the data used for model training. Unsupervised pattern recognition becomes more and more important in today's big data age (Wu, Zhu, Wu, & Ding, 2014). If the problem of automatic data preprocessing cannot be solved, the system will hardly to be pushed into a real usage, even if the learning algorithms inside are advanced. In this paper, we propose a novel data-driven stock price trend prediction system *Xuanwu*[1] The contribution of *Xuanwu* is three-fold. (1) it introduces unsupervised pattern recognition methods to generate training samples from raw transaction data without any human intervene; (2) it is a system for a real usage, in which multiple learning models are trained to meet the prediction goals derived from actual user requirements, and its application interface is of the maximum availability that is suitable for any stock and any prediction duration; (3) it provides a simple and easy-to-test framework in which different supervised learning models and feature selection methods can be easily integrated. Experimental results show that the proposed system outperforms some state-of-the-art methods in stock movement prediction even though the models of the compared methods are trained with carefully human-labeled samples.

The remainder of the paper is organized as follows. Section 2 describes the requirements from the aspect of users. Section 3 illustrates the architecture of the proposed system. Section 4 describes our unsupervised method to generate training samples. Section 5 presents our learning method in details. Section 6 addresses experimental results and discussions on these results. Section 7 concludes the paper and points out some future work.

2. Requirements

Xuanwu follows an assumption that the actual investment activities which are carried out based on the prediction results of the system do not have a far-reaching impact on the movements of the stock prices in the future. Therefore, it is specially suitable for the small startup investment companies that collect money from a small population and return the profits in fixed contract periods, whose investment volumes usually do not cause obvious market fluctuation. Otherwise, the prediction will be inaccurate. Moreover, these companies are unlikely to trade a stock very frequently (e.g., daily), nor do they hold a stock for a long time (e.g., more than three months) without make a deal. We outline the key points of user requirements in this section.

2.1. Prediction granularity

Nowadays, stock trades can take place in very high frequency when the market is open. The prediction granularity can be various such as second, minute, hour, day and even a fix investment period. *Xuanwu* chooses *trade day* as the prediction granularity. That is, it predicts the trend of a stock in a predefined period measured by *trade day*. Short-term stock prediction is also interesting (Lin, Yang, & Song, 2009) but not suitable for startup investment companies because of the constraints in the capital volumes and transaction costs. The standard prediction durations of *Xuanwu* (refer to Section 4.1) are 10, 15, 20, 30, 40, 50, and 60 trade days, which spans two weeks to three months.

2.2. Automatic pattern discovery

In the era of big data, continuous growth of generated data requires that the learning models update accordingly within short productive periods (Chen & Zhang, 2014; Sakurai, Matsubara, & Faloutsos, 2015). Obviously, it is no longer possible that training samples are still selected and labeled by humans. *Xuanwu* aims to get through all machine learning processes from generating training samples from the raw transaction data to building the prediction models without any human intervene. All that users need to do is to prepare a copy of original transaction data and then click to start the learning progress. All patterns that we are interesting in are extracted from every stock in the market. Then, the pieces of interested patterns are transformed into training samples.

2.3. Subdivision of classes

Since *Xuanwu* aims to predict the trend of a stock price movement by the end of a predefined period, it defines four main

[1] Xuanwu (Black Tortoise in English) is one of the Four Symbols of the Chinese constellations, usually depicted as a tortoise entwined together with a snake. The creature was thought to have spiritual power to predict the future.

classes based on the shapes of the close prices of a stock, i.e., the price will (1) rise up (class *Up*), (2) go down (class *Down*), (3) approximate the same (class *Flat*) and (4) vibrate with large amplitudes (class *Unknown*). Besides these four main classes, for class *Up* (*Down*), we are also interested in the extents of the growth (decline). Thus, for class *Up* (*Down*), it defines two sub-classes, i.e, the growth (decline) rate is (1) within the range [10%, 30%] (class *UA1* (*DA1*)) and (2) greater than 30% (class *UA2* (*DA2*)). In addition to the changes of close prices, we are also interested in the *relative return* which is the return achieved by an asset over a specific time period contrasted to a benchmark. For a period of n days, the *log relative return* can be calculated as follows:

$$rt = \sum_{i=1}^{n} (\ln(1 + f_i) - \ln(1 + b_i)), \quad (1)$$

where f_i and b_i are the asset return and benchmark return in the i-th day, respectively. According to the *relative return* rate, for class *Up* (*Down*), it defines two sub-classes, i.e., the increase (decrease) of relative return rate is (1) less than 10% (class *UR1* (*DR1*)), (2) within the range [10%, 20%] (class *UR2* (*DR2*)), and (3) greater than 20% (class *UR3* (*DR3*)). According to these two kinds of measures, *Xuanwu* runs at two prediction modes *AbsoluteMode* and *RelativeMode*.

In the *RelativeMode*, the patterns (shapes) of stocks are recognized and the classes are assigned based on the relative returns rather than the close prices, because investors sometimes are more interested in whether they can win over the "average", for example, when the investments are made in a typical bull (or bear) market. In a bull (or bear) market, most predictions are likely to be *Up* (or *Down*), if the close prices are used.

2.4. Prediction interface

The system provides an easy-to-use prediction API as follows:

```
XwResult predict(stockId, duration, modelName, mode),
```

where *XwResult* is an object holding the prediction results, *stockId* is an identifier for a stock, *duration* is a continuous sequence of dates (i.e., prediction duration), *modelName* specifies a learning model, and *mode* specifies a prediction mode (*AbsoluteMode* or *RelativeMode*).

When making prediction, we must specify a continuous sequence of dates (*duration*) which serves as a time series for prediction. For example, when we input the date sequence from Feb. 1, 2017 to Feb. 10, 2017, the price trend by the end of Feb. 15, 2017 will be returned if parameter *modelName* is not specified. There are several models in the system (refer to Section 4.1). If we do not explicitly specify one of them, the most suitable model will be automatically chosen for prediction. The prediction results are stored in an object *XwResult* defined as follows:

```
class XwResult {
 public Double probClassUp;
 public Double probClassDown;
 public Double probClassFlat;
 public Double probClassUnknown;
 public String modeType;
 public Double probFirstClass;
 public Double probSecondClass;
 public Double probThirdClass;
}.
```

Here, the first four elements are the probabilities of the four main classes *Up, Down, Flat* and *Unknown*. The sum of these first four values equals 1. *modeType* can be *AbsoluteMode* or *RelativeMode*. If *probClassUp* (*probClassDown*) is the largest one among the first four elements, when parameter *mode* in prediction API is set to *RelativeMode*, the last three elements provide the probabilities that the increase (decrease) of the price will be classified as *UR1* (*DR1*), *DR2* (*DR2*) and *UR3* (*DR3*), respectively. When parameter *mode* in prediction API is set to *AbsoluteMode*, *probThirdClass* is meaningless and the other two provide the probabilities of classes *UA1* (*DA1*) and *UA2* (*DA2*).

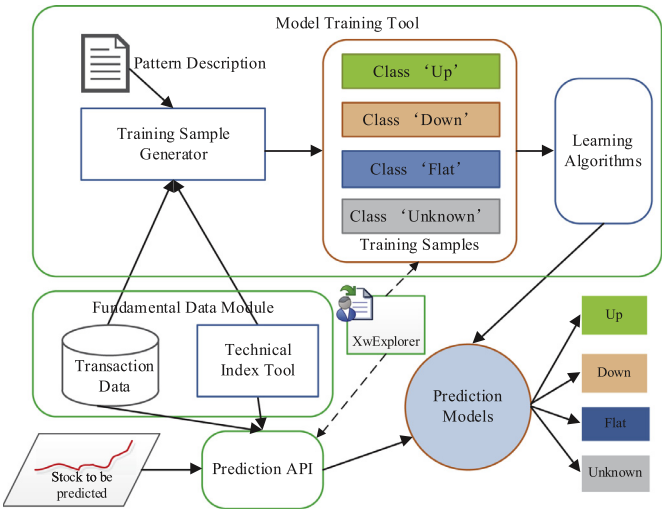

Fig. 1. The system architecture of *Xuanwu*.

2.5. Auxiliary functions

The system has some auxiliary functions to facilitate its usage. For example, we implemented a visualization tool to display the training samples. We can use this tool to check whether the shapes of the samples follow the desired patterns. We also implemented a set of applications that can be easily used for sample generation, model training, prediction and result validation.

2.6. A typical use case

We describe a typical use case to help understand the requirements of the system. Every day, the analyst in the company queries the prediction results by the end of the next five trade days of each stock via specifying the most recent 10 trade days. Then, by ranking all prediction results, s/he can quickly find those stocks which will rise up with the largest probabilities and the largest extent in the next five trade days. Analysts can focus on a small group of stocks with a higher probability of rising, which makes their choices more efficient and, to some extent, avoids selecting the stocks with a high probability of decline. Every three months, the analyst runs the model training tool again to update the all prediction models. The updated models will cover all transaction data of all stocks in the recent three months.

3. System architecture

The system architecture of *Xuanwu* is illustrated in Fig. 1. The main components of the system include a model training tool, a fundamental data module, a prediction API and a visualization tool (*XwExplorer*).

3.1. Fundamental data module (FDM)

FDM provides the original transaction data and a technical index tool, which serves as a basic database for generating training samples and the instances to be predicted. The technical index tool is implemented with an open source library Ta-Lib[2]. Traditional technical analysis indices can serve as features for model training (Ni, Ni, & Gao, 2011).

3.2. Model training tool (MTT)

MTT is the most complicated component in *Xuanwu*. It first reads the raw transaction data from FDM. Then, the Training Sample Generator (TSG) divides the raw transaction data into pieces (we call them *Clips*) and analyzes their shapes to match the patterns defined in a pattern description document. The training samples are generated according to the recognized patterns, combining with some technical indices as features. Finally, machine learning algorithms are utilized to build learning models. We directly implemented the learning algorithms using a well-known open source machine learning tool WEKA (Hall et al., 2009). The details of the training sample generation and learning model building will be further discussed in Section 4 and Section 5, respectively.

3.3. Prediction API and XwExplorer

The function signature of the prediction API has been discussed in Section 2.4. Another main function of the prediction API is to generate the instance to be predicted given the continuous sequence of dates through parameter *duration*. This function will be further discussed in Section 4.4 after the details of the training sample generation are introduced. *XwExplorer* is an easy-to-use visualization tool for users to build the learning models and check the results of the training sample generation and prediction.

4. Training sample generation

In this section, we will present our training sample generation scheme in *Xuanwu* in detail.

4.1. Morphological patterns for classes

Our training sample generation scheme is based on the shape of the close prices of a stock in predefined fixed trade durations. Different from some studies that focus on the traditional technical shapes defined by stock analysts (Jeon, Hong, & Chang, 2017), we only focus on several simple shapes. Traditional prediction by technical shapes is to make prediction when the predefined shapes appear, which usually does not work well because of the weak correlations between the trend of the price movement and these technical shapes (Nassirtoussi et al., 2014; Patel et al., 2015). Our method is to predict the probability of forming a predefined shape in a fixed duration when we only see some of data in early trade days. We define the close prices in a fixed duration which form a specific shape as a pattern. In our system, there are three kinds of durations as follows.

Definition 1. Pattern (Prediction) Duration (*PD*) is a time span within which we expect the trend of close prices exhibits a specific pattern.

Definition 2. Model (Training) Duration (*MD*) is a time span within which all data points are used as a training sample.

[2] http://ta-lib.org/.

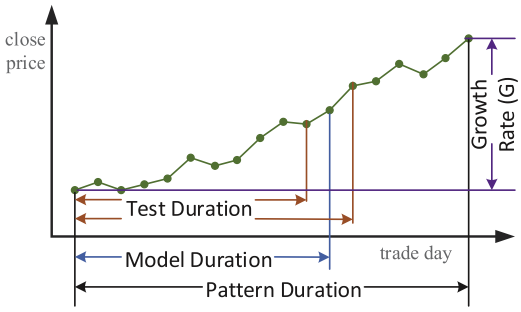

Fig. 2. Definitions of three durations and the growth rate when preprocessing the raw transaction data for a stock.

Table 1
Predefined durations in *Xuanwu*.

PD	MD	Models
10	6	M106, UA106, DA106, UR106, DR106
15	10	M1510, UA1510, DA1510, UR1510, DR1510
20	13	M2013, UA2013, DA2013, UR2013, DR2013
30	20	M3020, UA3020, DA3020, UR3020, DR3020
40	26	M4026, UA4026, DA4026, UR4026, DR4026
50	33	M5033, UA5033, DA5033, UR5033, DR5033
60	40	M6040, UA6040, DA6040, UR6040, DR6040

Definition 3. Test (Input) Duration (*TD*) is a time span within which all data points form an input instance whose labels and their probabilities will be predicted.

Fig. 2 shows the relationship among the three durations. *PD* is always greater than both *MD* and *TD*, because it embraces a pattern (i.e., the shape of the line connecting all close prices) that is expected to be eventually observed. The set of all data points within *PD* is named a *Clip*. Obviously, we cannot use a *Clip* as a training sample, because a meaningful forecast is to make prediction as soon as possible. The earlier we make a correct prediction, the greater the benefit will be. In our system, we use all data points within *MD* as a training sample. The time span of *MD* is two-thirds that of *PD*, which not only results in good prediction accuracy but also leaves enough room for price movement (increase or decrease) and enough opportunities for decision making. Simply speaking, the shape of a *Clip* determines the class label of its corresponding training sample. Usually, an unlabeled instance to be predicted should have the same dimensions as the training sample. However, in a real-world environment, we hope that the system can exhibit some robustness. For example, if the system can make prediction given 10 days of data, it should be able to work if 9 (or 11) days of data are given, even though it might not be so good. Thus, the length of our input duration (i.e., *TD*) is not necessarily the same as that of *MD*.

In our system, we have seven predefined *PD* and *MD* pairs, which reflect the actual requirement of the small investment companies. For each pair, we can train five different models: (1) *M*: a model for four main classes, (2) *UA*: a model for classes *UA1* and *UA2*, (3) *DA*: a model for classes *DA1* and *DA2*, (4) *UR*: a model for classes *UR1, UR2*, and *UR3*, and (5) *DR*: a model for classes *DR1, DR2*, and *DR3*. (The meanings of these classes can be found in Section 2.3). Table 1 lists all predefined durations and their corresponding models.

The patterns predefined in the current version of *Xuanwu* are illustrated in Fig. 3. It is well known that two typical trends - *Continuous Up* and *Sideways Up* - usually arouse investors' interests. These two kinds of trends form the first main class *Up*. Accordingly, the mirror image of these two trends, namely *Continuous Down* and *Sideways Down*, form the second main class *Down*. If the

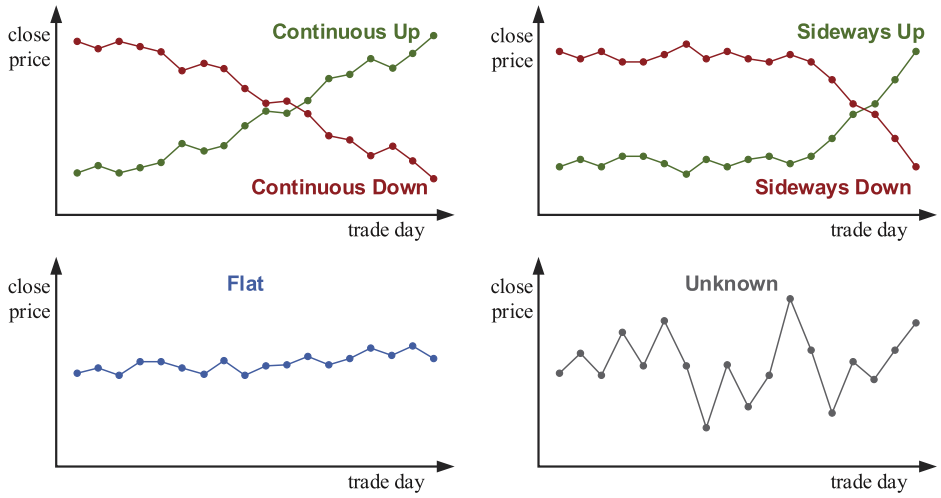

Fig. 3. Patterns predefined in *Xuanwu*.

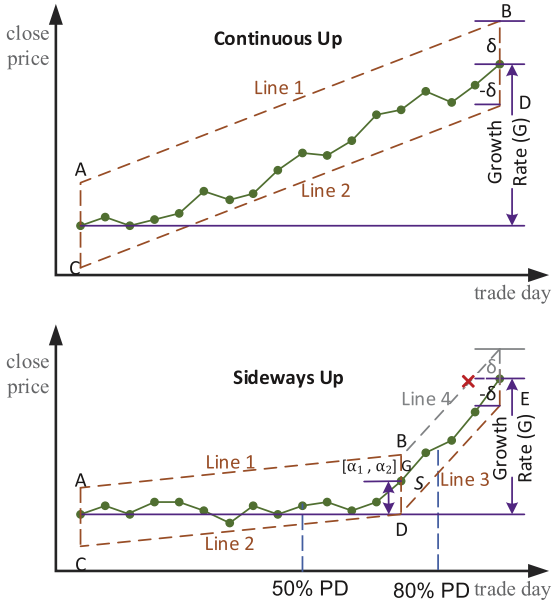

Fig. 4. Heuristic algorithms for recognition of patterns *Continuous Up* and *Sideways Up*.

close prices of the stock do not change considerably in a fixed duration, it will be the third class *Flat*. Those shapes that cannot be classified into the above three classes form the last class *Unknown*, which usually has large vibrations. The ten sub-classes, *UA1, UA2*, ... , etc., can be simply derived from classes *Up* and *Down* by calculating and ranking the growth and decline rates of absolute prices and relative returns.

4.2. Pattern recognition algorithms

To avoid labeling samples by humans, *Xuanwu* introduces unsupervised heuristic algorithms to recognize patterns. Fig. 4 illustrates the basic idea of our algorithms for recognition of patterns *Continuous Up* and *Sideways Up*.

The recognition of pattern *Continuous Up* is relatively simple. First, we calculate the growth rate (G) by comparing the close prices of the first and the last trade days. We allow the close prices of the first and last trade days to vary in the range $[-\delta G, \delta G]$ (e.g., $\delta = 20\%$). Then, we can draw two lines AB and CD. Finally, we check all points between the first and last trade days. If a certain proportion η (e.g., $\eta = 95\%$) of points are in the area $ABCD$, this *Clip* can be classified as *Continuous Up*.

The recognition of pattern *Sideways Up* is more complicated. We outline the main steps in Algorithm 1. The algorithm has more parameters but the principle is the same as the recognition of pattern *Continuous Up*. When we draw an area that the points should lie in, we have a larger search space, because point S can move in the range [50%PD, 80%PD]. All parameters in Algorithm 1 can be set as follows: $\delta = 20\%$, $\alpha_1 = 5\%$, $\alpha_2 = 10\%$ and $\eta = 95\%$. Note that when recognizing *Sideways Up* we do not need line 4 in Fig. 4.

Algorithm 1 Recognition for pattern *Sideways Up*.

Input: A *Clip*, α_1, α_2, δ and η
Output: Whether the *Clip* is *Sideways Up*

```
1:  Flag = FALSE
2:  Calculate the growth rate (G)
3:  for each point S between 50% and 80% PD do
4:      if the close price of S in the range [α1G, α2G] then
5:          Draw an area ABCD and a line DE (points A, B, C, and D
            are determined by α1 and α2; point E is determined by
            δ)
6:          if a proportion η of points between the first point and S
            are in area ABCD and a proportion η of points between S
            the last point are above line DE then
7:              Flag = TRUE
8:              return Flag
9:          end if
10:     end if
11: end for
12: return Flag
```

For patterns *Flat, Continuous Down* and *Sideways Down*, the algorithms are very similar.

4.3. Training samples

Xuanwu is designed with the intention of covering as many as shapes that a stock appear in its trade history. For each stock, we use a sliding window method to cut its historical transaction data into multiple *Clips* as Fig. 5 shows. The window size equals to a predefined prediction duration (PD). From the first trade day to the last trade day, the step of window sliding is one day. As we can see in Fig. 5, *Clip1* and *Clip2* follow pattern *Continuous Up*. They

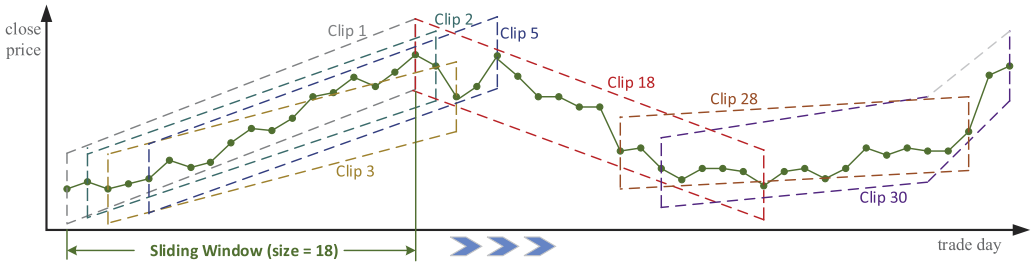

Fig. 5. Using sliding window method to cut transaction data of a stock into *Clips*.

will be labeled as class *Up*. As the window moves, *Clip3* does not follow any pattern, so it will be labeled as class *Unknown*. *Clip5* again follows *Continuous Up*, *Clip18* follows *Continuous Down* and *Clip30* follows *Sideways Up*. Thus, our method generates as many samples as we can, which covers all situations we may encounter in the future.

Note that *Clips* just represent the patterns that we are interested in, while they are not training samples. As mentioned in Section 4.1, the training sample should be the first *MD* points of a *Clip* whose duration is *PD*. Formally, for a *Clip* $C = \{\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_{PD}, y\}$, the corresponding training sample will be $C = \{\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_{MD}, y\}$, where $\mathbf{F}_i (i \in [1, PD])$ is the data in trade day i and y is the class label of this sample. $\mathbf{F}_i$) serves as a part of features of the training sample. Because our pattern recognition algorithm is based on the close prices of a stock, $\mathbf{F}_i$ at least contains one element close price. We can extend $\mathbf{F}_i$ by adding more information as follows:

$$\mathbf{F}_i = \{closeprice_i, openprice_i, highprice_i, lowprice_i, volume_i, t_i^{(1)}, t_i^{(2)}, \ldots, t_i^{(n)}\}, \quad (2)$$

where $t_i^{(k)} (k \in [1, n])$ is the k-th technical index of trade day i. Finally, the features of the training sample can be extended to high dimensions, because it includes information of multiple trade days.

4.4. Test instance

As defined in Section 4.1, test duration (*TD*) does not necessarily equal to model duration (*MD*), because the perdition API in Section 2.4 tells us that users can provide the duration with any number of continuous dates as the input. The system will find the most suitable model for prediction, if parameter *modelName* is not specified. For example, if the duration includes 9 days, the system will select model *M1510* in Table 1 for prediction. However, this model requires the features of an test instance have exactly 10 days' information. To address this issue, we introduced an interpolation method with linear scaling which has been widely used in image processing (Lehmann, Gonner, & Spitzer, 1999) to extend or shrink the test duration.

5. Model training

The current version of *Xuanwu* partially utilizes the off-the-self classification algorithms implemented in WEKA (Hall et al., 2009) to build learning models, among which we find that the models trained by the random forests (Breiman, 2001) perform well. To further improve the performance of the learned classifiers, we focus on two issues in model training: imbalanced class distribution and feature selection.

5.1. Learning with random forests

As described in Section 2.4, the prediction interface outputs the probability of a *Clip* being a certain pattern (class). The prediction model is built by the random forests (Breiman, 2001), which operate by constructing a multitude of decision trees at training time. The decision tree divides the features through its internal nodes to establish the classification model. When creating partitions to a feature, the goodness of a partition is measured by purity. If a partition is pure, for each sub-branch of this node, its instances belong to the same class. For node m, let N_m be the number of training samples arrived. For the root node, we have $N_m = N$. Suppose N_m^k out of N_m instances belong to class k and $\sum_k N_m^k = N_m$. If a test instance (*Clips*) x arrives at node m, the probability that it belongs to class k will be estimated as:

$$\hat{P}(k|\mathbf{x}, m) \equiv p_m^k = \frac{N_m^i}{N_m}. \quad (3)$$

For a node m presenting feature f, the best partition maximizes the purity that is measured by information gain. First, we define the entropy of node m as:

$$Entropy(m) = -\sum_{k=1}^{K} \left(p_m^k \log_2 p_m^k\right), \quad (4)$$

where K is the total number of classes. Then, for the v values of feature f, instances on node m are split into v partitions $\{m_1, \ldots, m_v\}$, where $m_j (1 \le j \le v)$ contains those instances in m that have outcome the vth value of f. We define the information gain on node m as follows:

$$Gain(m) = Entropy(m) - \sum_{j=1}^{v} \frac{|m_j|}{|m|} \times Entropy(m_j). \quad (5)$$

To overcome the over-fitting problem when using a single classification decision tree, we utilize the ensemble of multiple decision trees (i.e., random forests) to make prediction. Suppose we should construct a random forest with B decision trees. Each time, we perform a uniform random sampling with replacement on the training set to get a sub training set with N samples. Then, we use it to build a decision tree h_b whose output vector $\mathbf{p}_b = [p_b^1, \ldots, p_b^K]$ provides the probabilities of an instance being all classes. Finally, the probability of a test instance $\mathbf{x}$ being class k is calculated as:

$$\hat{P}(k|\mathbf{x}, H) = \frac{1}{B} \sum_{b=1}^{B} p_b^k. \quad (6)$$

Finally, the *hard* class label of instance $\mathbf{x}$ can be obtained through a plurality voting:

$$L(\mathbf{x}) = \underset{k \in 1,\ldots,K}{\operatorname{argmax}} \hat{P}(k|\mathbf{x}, H), \quad (7)$$

where function $L()$ returns the *hard* class label.

5.2. Imbalanced class distribution

Intuitively, the numbers of *Clips* that follow the patterns *Continuous Up, Sideways Up, Continuous Down, Sideways Up, Flat* are far less than the number of *Clips* that belongs to *Unknown*. Thus,

it will appear imbalanced learning problem (He & Garcia, 2009). Usually, the differences of the numbers of samples belonging to classes *Up, Down* and *Flat Sideways Up* are not too large. To form a better training set, we should balance the number of samples for all classes using the *undersampling* technique.

In fact, our imbalanced-class treatment is embedded in the construction of random forests. Algorithm 2 shows the skeleton of the

Algorithm 2 Skeleton of model training.

Input: Training set E
Output: Random forests H
1: Count the numbers of instances in four main classes i.e., n_1, n_2, n_3 and n_4
2: $n := \min\{n_1, n_2, n_3, n_4\}$
3: **for** each class k **do**
4: Sample n instances based on the descendent order of D_i in Eq.(8)
5: **end for**
6: The selected instances form a new training set E' sized $N = 4n$
7: **for** b :=1 to B **do**
8: Randomly sample (with replacement) N instances from training set E'
9: Conduct feature selection [optional]
10: Build a decision tree h_b
11: **end for**
12: Random forest H is the ensemble of $\{h_1, h_2, \ldots, h_B\}$ defined by Eq.(6).
13: **return** H

model training for the four main classes. We first count the numbers of *Clips* (n_1, n_2, n_3, n_4) belonging to four main classes *Up, Down, Flat*, and *Unknown*, respectively. The number of instances n for each class will be the minimum of them. Then, we sample n instances for each class (expect the class with n instances) based on the descendent order of D_i in Eq. (8). The new train set contains $4n$ instances selected. Finally, we build a random forest H which is an ensemble of B decision trees. Note that before building decision trees, we may select a subset of features to further optimize the performance of models. The feature selection procedure will be discussed in the next sub-section. The similar processes demonstrated in Algorithm 2 also can be applied to build sub-class models.

As for our *undersampling* in step 4, it is based on the measure of dissimilarity of instances, which is different from the traditional random sampling. Suppose that we will sample n instances from a pool totally containing n_4 instances belonging to class *Unknown*, for each instance $\mathbf{x}_i$ in the pool, we calculate the measure of dissimilarity as follows:

$$D_i = 1/\sum_{j=1}^{n_4} dist(\mathbf{x}_i, \mathbf{x}_j), \tag{8}$$

where function *dist*() returns the Euclidean distance between two instances $\mathbf{x}_i$ and $\mathbf{x}_j$. Then, we select the first n instances after all instances are sorted in descendent order of D_i. Our *undersampling* method guarantees that the selected n instances have the largest diversity, which increases the quality of the models learned from them.

5.3. Feature selection

As mentioned in Section 4.3, the features of a training sample can be extended by adding multiple technical indices of each trade day, which results in very high dimensional features. Usually, for each trade day in a training sample, we only use four price information, close price, open price, high price and low price, to train learning models. If the *MD* of a sample is 6, its feature dimension will be 24. Since we do not know whether other information (e.g., volume and technical indices) has positive impact on the performance of learned classifiers, feature selection techniques can be used for tuning their performance, which has been widely used by some stock prediction systems (Huang, Chang, Cheng, & Chang, 2012; Ni et al., 2011; Tsai & Hsiao, 2010).

Although there are a lot of feature selection methods can be used, finding an optimal combination of features is still an NP-hard problem (Dash & Liu, 1997). In our system, we use the *Forward Sequential Search* method, which selects one among all the candidates to the current state. It works in an iterative manner and once a candidate is selected it is not possible to go back. It does not guarantee an optimal result but has fast search speed. If the length of total sequence is n, the number of search steps must be limited by $O(n)$ and the complexity is determined taking into account the number t of evaluated sub-sets, which gives $O(n^{t+1})$. We need to modify this method a bit, because our feature selection works on each trade day in a sample, that is, once a candidate is selected, *MD* features will be added. For the features $\mathbf{F}_i$ of trade day i in a training sample, it starts with $\mathbf{F}_i = \{closeprice_i, openprice_i, highprice_i, lowprice_i\}$, the forward step consists of:

$$\mathbf{F}'_i := \{\mathbf{F}_i \cup f_i^{(k)} \in \mathcal{F}_i \backslash \mathbf{F}_i | J(\mathbf{F}_i \cup f_i^{(k)}) \text{ is bigger}\}, \tag{9}$$

where $\mathcal{F}_i$ is the complete set of features of trade day i and J is an evaluation measure. *Xuanwu* uses the empirical risk of the learned model as the evaluation measure J which is defined as:

$$J \equiv R_{emp}(H) = \sum_{i=1}^{N} \mathbb{I}(H(\tilde{\mathbf{x}}_i), y_i), \tag{10}$$

where $\mathbb{I}()$ is an indicator function. The stopping criterion can be: $|\mathbf{F}'_i| = n'$ (if n' has been fixed in advance), the value of J has not increased in the last j steps, or it surpasses a prefixed value J_0.

6. Evaluation

We evaluate our proposed system *Xuanwu* on 495 stocks in Shenzhen Growth Enterprise Market in China. The time span of the transaction data of these stocks is within the range from January 25, 2010 to October 1, 2016.

6.1. Experimental setup

We generated the data sets that can be used for model training from the raw transaction data of all stocks, using the methods described in Section 4 and Section 5.2. Since our prediction scheme is based on several fixed prediction durations, Table 1 shows that we need to build 70 learning models with the feature selection in consideration. Thus, we created 70 data sets for our evaluation. For each data set, we randomly held out 30% of its instances with respect to each class for testing and the remaining 70% of instances are used to train models. The models were trained using the random forests implemented in WEKA (Hall et al., 2009) with the default parameter settings. Because we tuned the class distributions to a relatively balanced status, we use the simple *accuracy* as our performance measure. Furthermore, since we randomly conducted a 30/70 splits to the data sets, we repeated the experiments ten times and the average values of *accuracy* and their standard deviations are reported.

6.2. Results on four main classes

We first evaluated the performance on the four main classes (i.e., *Up, Down, Flat* and *Unknown*) with respect to seven $PD - MD$

Table 2
Classification accuracy on four main classes (percent).

B/F\PD-MD	10-6	15-10	20-13	30-20
B	62.1 ± 3.2	65.7 ± 2.7	66.4 ± 3.1	70.8 ± 1.9
F	65.2 ± 2.9	67.2 ± 3.1	69.9 ± 2.8	74.2 ± 2.6
B/F\PD-MD	40-26	50-33	60-40	avg.
B	70.2 ± 2.1	69.8 ± 3.8	67.8 ± 2.5	67.5 ± 2.7
F	75.1 ± 3.7	72.2 ± 2.8	70.3 ± 2.2	70.6 ± 2.9

pairs (i.e., "10 – 6", "15 – 10", "20 – 13", "30 – 20", "40 – 26", "50 – 33", and "60 – 40"). The experimental results are listed in Table 2. The baseline of the performance of the learned models is marked as "B" in Table 2, where the features of each trade day only include four values (i.e., close price, open price, high price and low price). Comparatively, the performance optimized with feature selection is marked as "F". When we conducted the selection selection, we searched the candidates which consist of "volume" and other 74 technical indices calculated by Ta-Lib[3].

From Table 2, we can draw the conclusions as follows. (1) When features of each trade day in each training instance only include four values, the average accuracy of all seven models is 67.5%. Among these models, we find the performance of "30 – 20" and "40 – 26" is significantly better than that of the other. That is, our system is more suitable for predicting the trend of stock price in a relatively long term with the trade duration in the range of 30 to 40 days. (2) When more information such as "volume" and some technical indices are added into the features, the performance of all models has been improved. The average of the increment is greater than 3% (in absolute value). (3) Although we randomly choose 70% samples for model training, the standard deviations of all models are sound. The maximum absolute value of the standard deviations is only 3.7%. Small standard deviations suggest that our unsupervised algorithms for stock movement shape identification are accurate, which increases the robustness of the learned models.

6.3. Results on ten sub-classes

Our system adopts a *tow-stage* prediction scheme to obtain more refined results. For example, when a test instance is predicted as class *Up*, it will be re-predicted using the model *UA* to further determine the level of the increment of the price, i.e., the increment will be in the range [10%, 30%] (class *UA1*) or greater than 30% (class *UA2*). In this experiment, we still follow this *tow-stage* prediction scheme: a test instance is first predicted by the main model that includes four main classes and then re-predicted by a specific sub model according to the results of the main model. That is, if the test instance $\mathbf{x}$ belongs to class *UA1*, our experiment evaluates the joint probability distribution $Pr((\mathbf{x} = UA1) \cup (\mathbf{x} = UP))$. Since our models are trained for different predefined fixed prediction durations, there are totally 56 learned models if the feature selection are taken into account. All experimental results are listed in Table 3.

From Table 3, we can draw the conclusions as follows. (1) Compared with Table 2, the performance of all models in Table 3 decreases. This is because that the joint probability distribution cannot be greater than the marginal probability distribution. For example, we always have $Pr((\mathbf{x} = UA1) \cup (\mathbf{x} = UP)) \leq Pr(\mathbf{x} = UP)$. However, the average performance of these models is greater than 60%, which means that they are still available in a real usage. (2) The performance of the models for absolute price movements (i.e., *UA* and *DA*) outperforms that of the models for relative return

[3] http://ta-lib.org.

Table 3
Classification accuracy on ten sub-classes (percent) .

PD-MD	B/F	UA	DA	UR	DR
10-6	B	56.9 ± 3.1	57.2 ± 2.8	52.5 ± 3.7	51.8 ± 2.9
	F	59.2 ± 2.4	61.1 ± 3.3	58.5 ± 4.1	57.2 ± 3.1
15-10	B	60.4 ± 2.7	62.3 ± 2.2	55.6 ± 3.1	56.2 ± 3.5
	F	62.7 ± 3.1	65.7 ± 2.7	60.7 ± 3.3	59.7 ± 2.5
20-13	B	62.8 ± 2.4	63.5 ± 2.6	59.8 ± 2.1	57.5 ± 3.3
	F	65.3 ± 2.6	68.2 ± 2.1	63.9 ± 3.1	61.2 ± 4.1
30-20	B	67.3 ± 1.9	66.8 ± 2.1	64.3 ± 2.4	65.3 ± 2.7
	F	71.2 ± 2.2	72.6 ± 2.5	69.1 ± 2.4	68.5 ± 2.9
40-26	B	66.9 ± 2.4	65.4 ± 1.9	65.4 ± 2.3	65.8 ± 2.5
	F	70.1 ± 2.1	69.1 ± 3.1	68.3 ± 2.8	69.6 ± 3.2
50-33	B	65.4 ± 3.3	66.3 ± 2.5	64.7 ± 2.7	63.6 ± 2.2
	F	68.4 ± 2.9	70.0 ± 2.7	66.1 ± 2.2	66.6 ± 2.8
60-40	B	63.0 ± 2.2	64.5 ± 3.1	64.8 ± 2.8	64.3 ± 2.4
	F	66.7 ± 2.5	68.1 ± 2.8	67.3 ± 2.5	68.2 ± 2.7
avg.	B	63.2 ± 2.6	63.7 ± 2.5	61.0 ± 2.7	60.6 ± 2.8
	F	66.2 ± 2.5	67.8 ± 2.7	64.8 ± 2.9	64.4 ± 3.0

changes (i.e., *UR* and *DR*). Relative return is more related to the market status (e.g., in bull or bear). We found that using relative return while not the close price is more difficult to identify typical shapes, with results in lower accuracies. The models built with the relative return are especially useful when the market is in a typical bull or bear, where the shapes of most stocks are *Continuous Up* (*Continuous Down*). (3) For the models *UA* and *DA*, the performance of "30 – 20" and "40 – 26" is greater than that of the others, which is consistent with the results of the main model. For the models *UR* and *DR*, besides the "30 – 20" and "40 – 26" models, the "50 – 33", and "60 – 40" models also have good performance. That is, if we consider the relative return, our system is suitable for long-term prediction with the trade duration in the range 30 to 60 days. (4) Consistent with the results of the main model, when more information such as "volume" and some technical indices are added into the features, the performance of all models has been improved. The average of the increment is greater than 4% (in absolute value). (5) The robustness still maintains for these more subtle model according to the standard deviations. Using relative return only slightly increases the absolute values of standard deviations.

6.4. Results under the market volatility

In this experiment, we investigate the performance (in terms of accuracy) of our system under the market volatility. We extracted the data between August, 2014 and May, 2015 to generate test set T_{bull}, in which the market is a bull market with the composite index increasing from 1331 to 3542, the data between June, 2015 and January, 2016 to generate test set T_{bear}, in which the market is a bear market with the composite index decreasing from 3718 to 1994, and the data between February, 2016 and September, 2016 to generate test set $T_{shocking}$, in which the composite index fluctuates between 1192 and 2149 with the maximum value 2324 and the minimum value 1880.

Table 4 shows the classification accuracies for ten sub-classes of fourteen predefined models under three typical market patterns (bull, bear and shocking), where "A" presents that the models (*A-Model*) are trained using the samples whose classes are identified through their absolute close prices and "R" represents that the models (*R-Model*) are trained using the samples whose classes are identified through their relative returns. From this table, we can draw the conclusions as follows. (1) For the bull and bear market, the accuracies of the *A-Models* are significantly improved, compared with the values in Table 3. The reason is that under a bull or bear market, stocks usually have typical ascending or declining forms that are easier to be correctly classified. (2) For the shock-

Table 4
Classification accuracy under three typical market patterns (percent).

PD-MD	A/R	$\mathbf{T}_{bull}$	$\mathbf{T}_{bear}$	$\mathbf{T}_{shocking}$
10-6	A	76.8 ± 2.1	74.9 ± 2.2	58.3 ± 3.1
	R	54.1 ± 2.4	53.3 ± 2.8	53.7 ± 2.4
15-10	A	78.6 ± 2.2	79.9 ± 3.1	63.6 ± 3.2
	R	64.7 ± 2.8	65.2 ± 2.7	59.8 ± 2.3
20-13	A	80.1 ± 1.9	78.2 ± 2.6	66.7 ± 3.1
	R	63.7 ± 2.4	64.8 ± 2.7	63.3 ± 4.1
30-20	A	83.1 ± 2.0	80.3 ± 1.9	72.1 ± 2.7
	R	68.4 ± 1.9	67.3 ± 2.2	65.8 ± 3.1
40-26	A	82.9 ± 2.1	83.5 ± 1.6	71.7 ± 2.7
	R	67.7 ± 2.2	68.3 ± 1.9	66.7 ± 3.1
50-33	A	79.4 ± 1.8	82.2 ± 2.1	70.2 ± 2.3
	R	67.1 ± 2.6	66.3 ± 2.3	65.9 ± 3.2
60-40	A	77.8 ± 1.9	78.2 ± 2.7	68.2 ± 2.6
	R	68.1 ± 3.1	66.9 ± 2.3	66.6 ± 2.8
avg.	A	79.8 ± 2.0	79.6 ± 2.3	67.3 ± 2.8
	R	64.8 ± 2.5	64.6 ± 2.4	63.1 ± 3.0

Table 5
Comparisons with Shynkevich et al. (2017) in accuracy (percent).

PD-MD	SVM	ANN	k-NN	Ours
10-6 (10-7)	58.9 ± 3.6	55.2 ± 9.3	43.9 ± 6.6	**65.2 ± 2.9**
15-10 (15-10)	60.1 ± 4.6	55.9 ± 10.5	41.4 ± 5.4	**67.2 ± 3.1**
20-13 (20-15)	61.5 ± 3.6	57.0 ± 8.1	42.1 ± 6.4	**69.9 ± 2.8**
30-20 (30-20)	59.6 ± 3.7	52.6 ± 10.0	40.8 ± 5.1	**74.2 ± 2.6**
40-26 (30-25)	60.4 ± 5.1	54.4 ± 9.5	41.4 ± 5.2	**75.1 ± 3.7**
50-33 (30-30)	60.9 ± 5.3	55.4 ± 8.8	41.8 ± 5.4	**72.2 ± 2.8**

ing market, the accuracies of the *A-Model* are similar with their values in Table 3 but obviously worse than their values under a bull or bear market, because the shapes of stocks under a shocking market are more complicated to be correctly classified. (3) For all market patterns, the accuracies of the *R-Model*s are similar with their values in Table 3. The models trained using relative returns are more consistent in different market patterns. Overall, Because our training data cover sufficient historical information, their generated models are robust to the market volatility.

6.5. Comparisons with existing work in accuracy

Although direct comparisons among existing work are not easy because of different data preprocessing, model training methods and learning goals, we tried to select some recent work on the prediction of stock price movement and make relatively fair comparisons. Shynkevich, McGinnity, Coleman, Belatreche, and Li (2017) investigated the impact of varying input window length on the prediction accuracy. In their work, each training example consists of a sequence of technical indicators which are calculated from transaction data on trade days. They investigated the prediction performance under different learning algorithms (i.e., support vector machines, neural networks and k-nearest neighbors) by setting the numbers of the trade days (i.e., the length of the sequence) to 1, 3, 5, 7, 10, 15, 20, 25 and 30. Table 5 lists the comparison results in accuracy between the existing work (Shynkevich et al., 2017) and ours.

The first column of Table 5 represents the output-input models of both methods. The *PD-MD* pairs in the parentheses are the closest values to their counterparts of our method. The columns SVM, ANN and k-NN shows the performance of their methods in term of accuracy and the last column shows the performance of our method. Obviously, the performance of our method outperforms that of their method in both mean values and standard deviations under all *PD-MD* models. Furthermore, compared with their work, our method wins at two points: (1) our method provide more subtle classes, compared with theirs which only has three classes *Up, Down* and *NotMove*; and (2) the performance of our method will increase as the prediction duration increases. In their original study, they found some output-input (*PD-MD*) models, such as "7-5", "10-7", "15-10", "20-15", and "30-20", can archive higher performance compared with the other pairs. Their results have shown some consistency with our selections of *PD-MD* pairs, which is that the performance could be better if the length of model-duration is around two-thirds that of prediction-duration.

Table 6
Comparisons with Shynkevich et al. (2017) in return per trade (percent).

PD-MD	SVM	ANN	k-NN	Ours
10-6 (10-7)	4.23 ± 0.68	3.43 ± 1.45	1.78 ± 1.14	**4.58 ± 1.45**
15-10 (15-10)	5.34 ± 0.99	4.52 ± 2.01	2.12 ± 1.52	**6.02 ± 1.73**
20-13 (20-15)	6.41 ± 1.18	5.43 ± 2.28	2.36 ± 1.89	**7.25 ± 2.28**
30-20 (30-20)	7.72 ± 1.68	6.02 ± 3.35	2.94 ± 2.46	**9.65 ± 2.64**

6.6. Comparisons with existing work in return per trade

Return per trade is a commonly used metric when evaluating the performance of a trading system. An actual trading system may have complicated trading rules which may be generated dynamically according to the movement of stocks (Arévalo, García, Guijarro, & Peris, 2017; Cervelló-Royo, Guijarro, & Michniuk, 2015; Wang & Chan, 2007). Although the trading rules are out of the scope of this study, we still introduce very simple rules for evaluate the returns using our prediction system. When the prediction of a stock is *Up*, we buy the stock at the moment of the prediction and then sell it on the last day of a prediction duration (*PD*). The return of this trade can be calculated as:

$$R_{p,m} = (C_p - C_m)/C_m, \tag{11}$$

where C_p is the close price on the last day of prediction duration (*PD*), C_m is the close price on the day that the prediction is made, $R_{p,m}$ is the return from a trade. When the prediction of a stock is *Down*, we sell the stock at the moment of the prediction and then buy it back on the last day of a prediction duration. The return from of trade can be calculated as:

$$R_{m,p} = (C_m - C_p)/C_m. \tag{12}$$

We must point out that here we made a very large simplification to real-world systems. We presume that return per trade, as defined above, neglect transaction costs. Additionally, actual trades will hardly be negotiated in the same level as closing prices.

To compare with the existing study (Shynkevich et al., 2017), we randomly choose 50 stocks and the returns are calculated for the trades made during the testing phase. Return of single stock is averaged over the total number of trades made for this stock. Finally, the return per trade is averaged over 50 stocks for each *PD-MD* pair. Table 6 lists the comparison results in return per trade.

The return per trade value will increase as the investment period increases. Because the longest investment period in the compared study is 30 days, we only list the comparison results under four *PD-MD* pairs. Consistent with the previous comparison results in Section 6.5, the values of return per trade obtained by our method are obviously greater than those of the existing method under all *PD-MD* pairs, since the prediction accuracies of our method are always better. The comparison results reveal that even under such simple trade rules, our system can bring extra returns.

7. Conclusion and future work

For small startup investment companies, due to limited funds, it is impossible to trade in the stock market frequently. Instead,

they are interested in moderate investment periods that last a week to three months. To address the prediction of the stock price trend in such periods, this paper proposes a novel data-driven system *Xuanwu*. The system gets through all machine learning processes from generating training samples from the original transaction data to building the prediction models without any human intervene. It first uses a sliding window method to cut the historical transaction data of each stock into multiple *Clips* whose length equals to a predefined prediction duration. Then, according the shapes that the close prices of these *Clips* appear, it utilizes an unsupervised heuristic algorithm to classify them into four main classes: *Up, Down, Flat*, and *Unknown*. For the *Clips* belonging to classes *Up* and *Down*, they are further classified into different levels which can reflect the extents of their growth and decline rates with respect to both absolute close price and relative return rate. The training sets are derived from these *Clips* by sampling different classes of samples for imbalanced class distribution. Finally, learning models are trained from these training sets with or without feature selection.

The real-world evaluations on seven-year Shenzhen Growth Enterprise Market (China) transaction data show the advantages of the proposed systems as follows. First, the unsupervised training sample generation is effective and efficient, accelerating the model reproduction. Second, the performance of our learning models outperform some existing methods in terms of accuracy and return per trade, because our learning method integrates random forests, imbalance learning and feature selection in a uniform process. Finally, our prediction models are robust to the market volatility.

There is a large room for performance improvement in the future. First, we will study the prediction performance when different learning algorithms are applied to the training sets. Second, more complicated feature selection methods will be examined to select better combinations of features. Finally, our unsurprised heuristic algorithms for pattern recognition can be further improved for more different shapes.

Acknowledgements

This research has been supported by the National Natural Science Foundation of China under Grant No. 61603186, the Natural Science Foundation of Jiangsu Province, China, under Grant No. BK20160843, the China Postdoctoral Science Foundation under Grant No. 2016M590457 and 2017T100370, and the Science Foundation of the Science and Technology Commission of the Central Military Commission (Youth Project), China.

References

Abarbanell, J. S., & Bernard, V. L. (1992). Tests of analysts' overreaction/underreaction to earnings information as an explanation for anomalous stock price behavior. *The Journal of Finance, 47*(3), 1181–1207.
Adam, K., Marcet, A., & Nicolini, J. P. (2016). Stock market volatility and learning. *The Journal of Finance, 71*(1), 33–82.
Adebiyi, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Comparison of ARIMA and artificial neural networks models for stock price prediction. *Journal of Applied Mathematics, 2014*.
Arévalo, R., García, J., Guijarro, F., & Peris, A. (2017). A dynamic trading rule based on filtered flag pattern recognition for stock market price forecasting. *Expert Systems with Applications, 81*, 177–192.
Ballings, M., Van den Poel, D., Hespeels, N., & Gryp, R. (2015). Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications, 42*(20), 7046–7056.
Barak, S., Arjmand, A., & Ortobelli, S. (2017). Fusion of multiple diverse predictors in stock market. *Information Fusion, 36*, 90–102.
Blume, L., Easley, D., & O'hara, M. (1994). Market statistics and technical analysis: The role of volume. *The Journal of Finance, 49*(1), 153–181.
Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science, 2*(1), 1–8.
Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.
Cervelló-Royo, R., Guijarro, F., & Michniuk, K. (2015). Stock market trading rule based on pattern recognition and technical analysis: Forecasting the DJIA index with intraday data. *Expert Systems with Applications, 42*(14), 5963–5975.
Chen, C. P., & Zhang, C.-Y. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences, 275*, 314–347.
Dash, M., & Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis, 1*(1–4), 131–156.
Ding, X., Zhang, Y., Liu, T., & Duan, J. (2015). Deep learning for event-driven stock prediction. In *Ijcai* (pp. 2327–2333).
Fama, E. F. (1995). Random walks in stock market prices. *Financial Analysts Journal, 51*(1), 75–80.
Gerlein, E. A., McGinnity, M., Belatreche, A., & Coleman, S. (2016). Evaluating machine learning classification for financial trading: An empirical approach. *Expert Systems with Applications, 54*, 193–207.
Göçken, M., Özçalıcı, M., Boru, A., & Dosdoğru, A. T. (2016). Integrating metaheuristics and artificial neural networks for improved stock price prediction. *Expert Systems with Applications, 44*, 320–331.
Hadavandi, E., Shavandi, H., & Ghanbari, A. (2010). Integration of genetic fuzzy systems and artificial neural networks for stock price forecasting. *Knowledge-Based Systems, 23*(8), 800–808.
Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The weka data mining software: An update. *ACM SIGKDD Explorations Newsletter, 11*(1), 10–18.
He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering, 21*(9), 1263–1284.
Huang, C.-F., Chang, B. R., Cheng, D.-W., & Chang, C.-H. (2012). Feature selection and parameter optimization of a fuzzy-based stock selection model using genetic algorithms. *International Journal of Fuzzy Systems, 14*(1), 65–75.
Jeon, S., Hong, B., & Chang, V. (2017). Pattern graph tracking-based stock price prediction using big data. *Future Generation Computer Systems*.
Kim, K.-j. (2003). Financial time series forecasting using support vector machines. *Neurocomputing, 55*(1), 307–319.
Lawrence, R. (1997). *Using neural networks to forecast stock market prices*: 333. University of Manitoba.
Lehmann, T. M., Gonner, C., & Spitzer, K. (1999). Survey: Interpolation methods in medical image processing. *IEEE Transactions on Medical Imaging, 18*(11), 1049–1075.
Lin, X., Yang, Z., & Song, Y. (2009). Short-term stock price prediction based on echo state networks. *Expert Systems with Applications, 36*(3), 7313–7317.
Nassirtoussi, A. K., Aghabozorgi, S., Wah, T. Y., & Ngo, D. C. L. (2014). Text mining for market prediction: A systematic review. *Expert Systems with Applications, 41*(16), 7653–7670.
Ni, L.-P., Ni, Z.-W., & Gao, Y.-Z. (2011). Stock trend prediction based on fractal feature selection and support vector machine. *Expert Systems with Applications, 38*(5), 5569–5576.
Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications, 42*(1), 259–268.
Ruiz, E. J., Hristidis, V., Castillo, C., Gionis, A., & Jaimes, A. (2012). Correlating financial time series with micro-blogging activity. In *Proceedings of the fifth ACM international conference on web search and data mining* (pp. 513–522). ACM.
Sakurai, Y., Matsubara, Y., & Faloutsos, C. (2015). Mining and forecasting of big time-series data. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data* (pp. 919–922). ACM.
Schöneburg, E. (1990). Stock price prediction using neural networks: A project report. *Neurocomputing, 2*(1), 17–27.
Schumaker, R. P., & Chen, H. (2009). Textual analysis of stock market prediction using breaking financial news: The AZFin text system. *ACM Transactions on Information Systems (TOIS), 27*(2), 12.
Shynkevich, Y., McGinnity, T., Coleman, S., Belatreche, A., & Li, Y. (2017). Forecasting price movements using technical indicators: Investigating the impact of varying input window length. *Neurocomputing*.
Si, J., Mukherjee, A., Liu, B., Li, Q., Li, H., & Deng, X. (2013). Exploiting topic based twitter sentiment for stock prediction. *ACL, 2*, 24–29.
Timmermann, A., & Granger, C. W. (2004). Efficient market hypothesis and forecasting. *International Journal of Forecasting, 20*(1), 15–27.
Tsai, C., & Wang, S. (2009). Stock price forecasting by hybrid machine learning techniques. In *Proceedings of the international multiconference of engineers and computer scientists* (pp. 755–760).
Tsai, C.-F., & Hsiao, Y.-C. (2010). Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches. *Decision Support Systems, 50*(1), 258–269.
Tsai, C.-F., Lin, Y.-C., Yen, D. C., & Chen, Y.-M. (2011). Predicting stock returns by classifier ensembles. *Applied Soft Computing, 11*(2), 2452–2459.
Tsibouris, G., & Zeidenberg, M. (1995). Testing the efficient markets hypothesis with gradient descent algorithms. In *Neural networks in the capital markets: 8* (pp. 127–136). Wiley: Chichester.
Wang, J.-L., & Chan, S.-H. (2007). Stock market trading rule discovery using pattern recognition and technical analysis. *Expert Systems with Applications, 33*(2), 304–315.
Wu, X., Zhu, X., Wu, G.-Q., & Ding, W. (2014). Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering, 26*(1), 97–107.
Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting, 14*(1), 35–62.