

# KNN++: An Enhanced K-Nearest Neighbor Approach for Classifying Data with Heterogeneous Views

Ying Xie

**Abstract** In this paper, we proposed an enhanced KNN approach, which is denoted as KNN++, for classifying complex data with heterogeneous views. Any type of view can be utilized when applying the KNN++ method, as long as a distance function can be defined on that view. The KNN++ includes an integral learning component that learns the weight of each view. Furthermore, the KNN++ method factors in not only the training data, but also the unknown instance itself when assessing the importance of different views in classifying the unknown instance. Experimental results on predicting SPY daily open price demonstrates the effectiveness of this method in classification. The time complexity of the KNN++ method is linear to the size of the training dataset.

## 1 Introduction

Most of the methods of machine learning were developed based on the assumption that each data instance can be represented by a unified view, such as a vector, a sequence, a matrix, or a graph. However, in reality, many complex data may have multiple heterogeneous views, each of which reflects one aspect of the data. For instance, imaging our task is to develop a machine learning algorithm to predict the price of a stock for tomorrow. For this prediction, we may factor in multiple heterogeneous views of historical stock data, including statistics of daily price movement (may be represented as a vector of daily open, daily high, daily low, and daily close), stock movement over the past week or a longer time frame (may be represented as a time series), investors' sentiment (may be represented as a bag of words), and business earning reports (may be represented as a matrix). Another example is the data-driven detection of Alzheimer's disease based on patient data. Different types of patient data are worth examining for this purpose, such as brain images (including structural and

---

Y. Xie (✉)

Department of Computer Science, Kennesaw State University, 1000 Chastain Road,  
Kennesaw, GA 30144, USA  
e-mail: yxie2@kennesaw.edu

functional images), patients genetic risk profiling, trajectories of multiple biomarkers over the time course, and patient symptoms and records [1].

Given complex data with multiple heterogeneous views, it is very challenging, if not impossible, to train a single model over those heterogeneous views of the given data [2]. In order to cover all those important views, a multiple-classifier system [3] can be employed. With a multiple-classifier system, different models are trained by using different machine learning techniques on different views of the data. Then the final prediction of an unknown instance is reduced from those predications generated by different models. The reducing process can be a selection process, such as using cross-validation to pick the best model [4]; or a voting process that aggregates predications generated by different models; or a stacking process that uses another layer of learning to optimally combine the underneath predications [5].

It is observed that, by a multiple-classifier system, the process of selecting or stacking multiple models trained on different views of the training data are completely independent from the unknown instance that the final predication will be made for. In other words, no matter how distant two unknown instances are, the model or underneath view selecting or stacking process are completely the same. However, the author argues that there are situations where unknown instances themselves, for which predications are made for, should be factored in when selecting or stacking underneath views in prediction. In the example of data-driven detection of Alzheimer's disease, functional images may be the most important view in detecting the disease for a patient; whereas trajectories of certain biomarkers may be the most import view in detecting for another patient.

Therefore, the primary goal of this research is to design a new machine learning strategy such that, (1) it can utilize different heterogeneous views of a given data in classification; (2) not only the training data, but also the unknown instance itself are factored in when assessing the importance of different views in classifying the unknown instance. When designing such a strategy, we considered enhancing the K-Nearest Neighbor (KNN) approach [6, 7] towards the above two design goals, because of the following reasons. First, KNN can be applied to any type of data, as long as a distance function is defined between any two given instances. This characteristic allows us to apply KNN separately to different heterogeneous views for a given data. Second, KNN is essentially an instance-based learning strategy, where the class label of an unknown instance is derived from the class labels of its nearest neighbors. We were thinking that this characteristic might be expanded such that the unknown instance is able to influence the assessment of different views of the given data through its nearest neighbors.

Based upon the above thinking, we proposed a enhanced KNN approach that is called KNN++ for classifying data with heterogeneous views. As we will show in the following sections, the KNN++ satisfies the two design goals that we mentioned above and is able to effectively and efficiently predicting stock price movement for the next time period. The rest of the paper will be organized as follows. In Sect. 2, we will describe the proposed KNN++ in details. In Sect. 3, we will model the task of predicting stock price movement for the next time period as a classification problem and show that the unknown instance for prediction should play a role in

assessing the importance of different views of the given data. In Sect. 4, preliminary experimental results on stock price movement prediction will be presented. Finally, Sect. 5 contains some further discussion of this proposed method as well as some possible future improvement on it.

## 2 KNN++: An Enhanced KNN Approach for Classification

Given a set of data instances  $U$  with  $N$  elements  $\{u_1, u_2, \dots, u_N\}$  and a set of class labels  $C$  with  $M$  elements  $\{c_1, c_2, \dots, c_M\}$ ,  $U$  is divided into  $M + 1$  disjoint regions  $\{r_{c_1}, r_{c_2}, \dots, r_{c_M}, r_{c_{M+1}}\}$ , such that if a data instance  $u_i \in r_{c_j}$  (where  $1 \leq j \leq M$ ), then the class label  $c_j$  is assigned to  $x_i$ ; if  $u_i \in r_{c_{M+1}}$ ,  $u_i$  is viewed as an unknown instance. Now, the classification problem that is addressed here is that, for each  $u_i \in r_{c_{M+1}}$ , we need to assign a class label  $c_j \in M$  to it. We further assume that a set of distinct distance functions  $D = \{d_1, d_2, \dots, d_L\}$  can be defined on  $U$ , such that for any  $d_x \in D$  and any  $u_i, u_j$ , and  $u_k \in U$ , we have  $d_x(u_i, u_j) + d_x(u_j, u_k) \geq d_x(u_i, u_k)$ .

The classical KNN approach assumes that  $|D| = 1$ ; in other words, only one distance function is used in the classification process. However, a complex data set may have multiple heterogeneous views. It is often challenging, if not possible, to define one single comprehensive distance function that is able to take into consideration of multiple heterogeneous views. Therefore, the proposed KNN++ method utilizes multiple distance functions, each of which is defined on one heterogeneous view of the data. Let's take as an example the data-driven detection of Alzheimer's disease based on patient data. One distance function on patient cases may be defined on brain images; one distance function may be defined on patients' genetic risk profiles; another distance function may be defined on trajectories of certain biomarker; and so on. In this case, it is obviously difficult to define one single distance function based on all of these heterogeneous views. However, different distance functions may be defined on different views of the given data, such that one distance function represents the view upon which the function is defined. Therefore, in order to take advantages of multiple views, the proposed KNN++ method utilizes multiple distance functions.

We also need to consider that not every view of the data has equal significance towards the classification of a given instance. Therefore, an important component of this proposed KNN++ method is to learn the weight of each distance function that is defined on each view. Furthermore, the weights of distance functions should not remain unchanged for different unknown instances. For instance, given certain patient case, brain image may be more important than others in detecting the disease; while for another case, a biomarker may serve as a better indicator. Hence, the learning process of the proposed KNN++ method is instance based. In other words, different unknown instances may favor different views.

Informally, the KNN++ method can be described in the following way. Given an unknown instance, the method first learns the weight of each distance function that

is defined on each view of the data. The weight of a distance function is determined by the labelled representatives of the unknown instance with respect to this distance function. More specifically, the  $K$  nearest neighbors of the unknown instance, which are found using this distance function, serve as the labelled representatives of the unknown instance corresponding to this distance function. For each of the labelled representatives, the KNN++ method finds the  $K$  nearest neighbors of this labelled representative by using the same distance function; then counts how many instances within the  $K$  nearest neighbors of this labelled representative actually have the same class label as this representative. The weight of this distance function is then determined by summing up all those numbers across all the labelled representatives. After the weights of all those distance functions are calculated, the set of the  $K$  nearest neighbors found by each of the distance functions for the unknown instance is weighted by the weight of that distance function. That means, the class label of each instance in those sets of  $K$  nearest neighbors is weighted by the weight of the set that this instance belongs to. Then, the final class label that is assigned to the unknown instance by this KNN++ method is the one with the highest weighted sum across all the sets of  $K$  nearest neighbors of this unknown instance.

Formally, given an unknown instance  $u \in r_{c_{M+1}}$ , the proposed KNN++ method can be described as follows.

```

for each class label  $c_i \in C$ 
    initialize  $count_{c_i,u}$  to be 0;
for each  $d_l \in D$ 
    set the weight of  $d_l$  as  $w_{dl}$ ; and initialize  $w_{dl}$  to
    be 0;
    find the  $K$  nearest neighbours of  $u$  in the set of
     $(U - r_{c_{M+1}})$  by using  $d_l$  , and store them in a set
     $NN_{dl}(u)$ ;
    for each  $u_i \in NN_{dl}(u)$ 
        find the  $K$  nearest neighbours of  $u$  in the
        set of  $(U - r_{c_{M+1}})$  by using  $d_l$  , and store them
        in a set  $NN_{dl}(u_i)$ ;
        for each  $u_j \in NN_{dl}(u_i)$ 
            if  $u_j$  and  $u_i$  has the same class label
                 $w_{dl}++$  ;
    for each  $u_i \in NN_{dl}(u)$ 
        let  $u_i$ 's class label to be  $c$ ;
         $count_{c,u} += w_{dl}$ ;
initialize  $final\_label$  to be "unknown";
initialize  $final\_label\_count$  to be 0;
for each class label  $c_i \in C$ 
    if ( $count_{c_i,u} > final\_label\_count$ )
         $final\_label\_count = count_{c_i,u}$ ;
         $final\_label = c_i$ ;
output  $final\_label$ ;

```

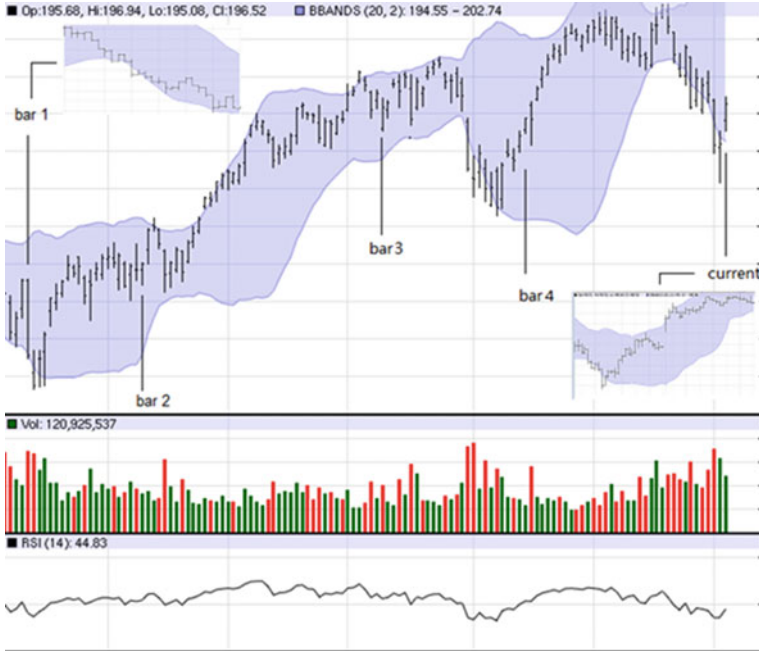
As one can see that, the proposed KNN++ method is able to utilize any view of the data, as long as a proper distance function can be defined on that view. The KNN++ method also automatically learns weights for each view with respect to a given unknown instance through the distance function defined on that view. Furthermore, just as the classic KNN approach, the KNN++ only takes one parameter, which is the K value.

### 3 Categorize Predicting Stock Price Movement as Classifying Data with Heterogeneous Views

As being well known, stock price prediction is a broad yet very challenging research area, given that many different factors may influence stock price movement. A variety of machine learning techniques have been used in predicting stock price movement, such as SVM [8], decision tree [9], and artificial neural network [10]. However, these methods only take features that can be expressed in certain homogeneous format, such as vectors. What if we want to utilize multiple heterogeneous types of information in our predictions? For example, some information is expressed in value vectors, some in time series, some in bags of words, and some in matrix. In this section, we will categorize one particular stock price prediction task as a classification problem on data with heterogeneous information views.

The prediction task that we will categorize can be described as follows. Given the historical price movement of a stock or index, such as SPY, in a particular time frame (which can be weekly, daily, hourly, 5 min, and so on), that task is to predict the price movement for the next time unit (next week, next day, next hour, or next 5 min). If we visualize the price movement at a historical time unit using a bar as shown in Fig. 1, the task is to predict what will be the upcoming bar. To categorize this task as a classification problem, we first annotate each historical bar by using the bar that immediately follows it. For instance, bar1 in Fig. 1 can be annotated as “Open Down”, given that its following bar has a lower open price compared to its own close price; bar2 can be annotated as “Open Up”, given that its following bar has a higher open price compared to its own close price. Therefore, we identify a set of 2 class labels {“Open Up”, “Open Down”}. Now, the task is to assign a class label to the current bar that is the last bar shown in Fig. 1, given that the class label assigned to the current bar is actually the predication of the price movement for the upcoming time unit. Please note that we use the prediction of open price movement for the next time unit as an illustration; nevertheless, without loss of generality, we may also predict the close price (green or red), the highest price (higher or lower), and the lowest price (higher or lower) for the next time unit in the same way.

In order to predict the class label for the current bar, we need to identify features that can be used to describe each bar. This is where complexity comes into the picture, given that each historical bar could be associated with numerous factors



**Fig. 1** A sample chart of stock price movement (chart was copied from [www.barchart.com](http://www.barchart.com))

that may indicate or correlate with the stock price movement at next time unit. Moreover, those factors may be heterogeneous in nature, formats, and scales; and some features are complex data by themselves. For instance, just consider the price chart as shown in Fig. 1 alone, each individual bar (an instance) may be described as a vector of 4 prices (open, close, high, and low); or a more detailed time series of intraday price movement, assuming each bar represents a daily price (intraday details for bar 1 and the current bar are shown in Fig. 1); or a time series for price movement within a bigger time frame that ends with this bar. Therefore, it is very challenging for any classic machine learning algorithm to utilize all these features together. However, by using our proposed KNN++ method, each of those features or a combination of a group of features can serve as a view for a given bar, and all views together deliver comprehensive information for that bar. A view can be certain technique analysis (TA) features as described above, or certain sentimental features, or fundamental analysis (FA) features. As long as a proper distance function can be defined on a view, that view can be unitized by the proposed KNN++ method. Therefore, the KNN++ method provides a framework for factoring in heterogeneous information in supervised learning.

## 4 Experimental Study of KNN++ on Predicting SPY Daily Open Price Movement

In this section, we demonstrate the experimental results of applying KNN++ to stock price prediction. In our experiments, SPY historical daily prices were used as our data set. SPY is the ticker for the SPDR S&P 500 ETF Trust that corresponds to the price and yield performance of the S&P 500 index. The dataset of SPY historical prices was downloaded from <http://finance.yahoo.com>. Part of the records in this data set is shown in Fig. 2. Each record of this data set includes the following price values for a particular day: Open (open price of the day), High (highest price of the day), Low (lowest price of the day), and Close (close price of the day). Our task in this experiment is to predict the open price movement for the next day.

Based on the categorization process described in Sect. 3, we label each day in the data set as either “Open Up”, if the open price of the next day is greater than or equal to this day’s close price; or “Open Down”, if otherwise. The goal is to predict whether it is “Open Up” or “Open Down” for tomorrow.

In real-life prediction, we may factor in a variety of heterogeneous views; however, in this experiment, which serves as a proof of concept, we only generate the following views based on the available information in the data set.

**Daily\_Move:** a vector defined as

$\{ \text{Close-Open}, \text{High-Low}, \text{High-Close} \}$

**Daily\_Move\_Relative\_to\_Yesterday:** a vector defined as

$\{ \text{High-yesterdayHigh}, \text{Low-yesterdayLow}, \\ \text{Open-yesterdayOpen}, \text{Close-yesterdayClose} \}$

**Daily\_Move\_Relative\_to\_21MovingAverage:** a vector defined as

$\{ \text{High-21MovingAverage}, \text{Close-21MovingAverage}, \\ \text{Low-21MovingAverage}, \text{Open-21MovingAverage} \}$

**Relative\_Close\_8:** a time series defined as

$\langle \text{Close\_at\_7\_days\_ago} - \text{Close}, \\ \text{Close\_at\_6\_day\_ago} - \text{Close}, \\ \dots, \\ \text{Clos\_at\_1\_day\_ago} - \text{Close} \rangle$

**Relative\_Close\_21:** a time series defined as

$\langle \text{Close\_at\_20\_days\_ago} - \text{Close}, \\ \text{Close\_at\_19\_day\_ago} - \text{Close}, \\ \dots, \\ \text{Clos\_at\_1\_day\_ago} - \text{Close} \rangle$

Although all these views are derived from daily price information, they are heterogeneous in the sense that some views are vectors, others are time series. It is challenging to define one single distance function across all these views. Therefore,

Date	Open	High	Low	Close	Volume	Adj Close*
Jul 31, 2015	211.42	211.45	210.16	210.45	97,697,400	210.45
Jul 30, 2015	210.16	211.02	209.42	210.82	89,368,700	210.82
Jul 29, 2015	209.48	211.04	209.31	210.77	102,056,400	210.77
Jul 28, 2015	207.79	209.50	206.80	209.31	118,553,000	209.31
Jul 27, 2015	208.00	208.00	206.26	206.74	124,398,800	206.74
Jul 24, 2015	210.30	210.37	207.60	207.94	109,271,100	207.94
Jul 23, 2015	211.53	211.65	209.75	210.14	87,846,600	210.14
Jul 22, 2015	210.93	211.77	210.89	211.29	84,385,700	211.29
Jul 21, 2015	212.43	212.74	211.39	211.76	75,035,700	211.76
Jul 20, 2015	212.75	213.18	212.21	212.62	65,523,100	212.62
Jul 17, 2015	212.29	212.55	211.80	212.47	85,410,300	212.47
Jul 16, 2015	211.87	212.30	211.58	212.27	98,172,600	212.27
Jul 15, 2015	210.73	211.28	210.04	210.63	93,588,200	210.63
Jul 14, 2015	209.72	211.05	209.65	210.72	78,370,300	210.72
Jul 13, 2015	208.99	209.90	208.94	209.76	103,805,000	209.76
Jul 10, 2015	207.29	207.98	204.95	207.48	126,039,800	207.48
Jul 9, 2015	207.04	207.35	204.77	204.80	139,210,600	204.80
Jul 8, 2015	208.02	208.02	204.25	204.53	159,250,700	204.53
Jul 7, 2015	206.96	208.17	204.11	208.01	170,938,200	208.01

Fig. 2 Sample records from the data set downloaded from Yahoo! Finance

we follow the KNN++ method by defining a distance function on each of these views. To test the performance of the KNN++ method on SPY daily open price prediction, we select the subset of data beginning from March 1, 2015 through July 31, 2015 as the test data. You may notice that there is no obvious trending in this selected time frame, as shown in Fig. 3. The reason for selecting this subset as the test data is to avoid the situation where an obvious up-trending (or down-trending) may include majority of the instances in that trend having an “Open Up” (or an “Open Down”) class label.

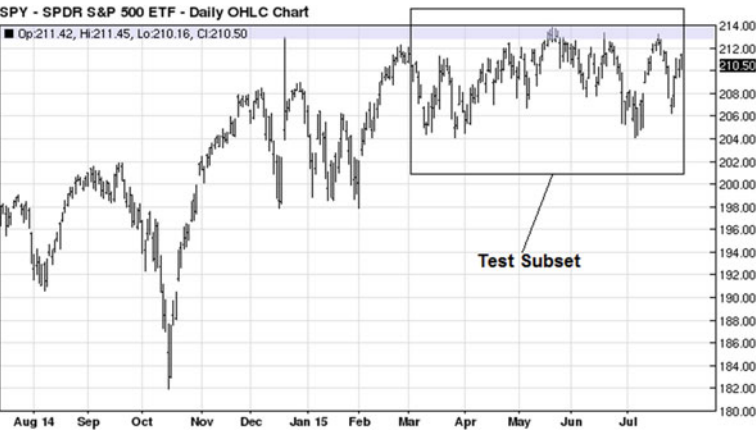


Fig. 3 Illustration of the test subset (chart was copied from [www.barchart.com](http://www.barchart.com))



**Table 1** Results of predicting open price for each trading day from March 2, 2015 through July 31, 2015

Tested month	# Decidable instances	# Correctly predicted instances	Accuracy rate (round to 2 digits) (%)
July, 2015	14	9	64
June, 2015	16	11	69
May, 2015	12	9	75
April, 2015	10	5	50
March, 2015	12	8	67
Total	64	42	66

For each instance in the test subset, we use the dataset ranging from January 3, 2000 through the last instance before the month of the test subset as the training set. For example, if the test instance is March 5, 2015, the training subset goes from January 3, 2000 through February 27, 2015; if the test instance is July 31, 2015, the training subset goes from January 3, 2000 through June 30, 2015. Finally, we use 7NN++ for the open price prediction. In other words, for each test instance, we find 7 nearest neighbors of this instance by using the 7NN++ method. If 5 or more out of the 7 nearest neighbors are consistent in one class label, we assign that class label to the test instance; otherwise, we view the test instance as undecidable based on the given information. The final experimental results are listed in Table 1.

As shown in Table 1, the total accuracy rate for these 5 months is 66 %; 4 out of 5 months have accuracy rate greater than 64 %; the highest monthly accuracy reaches 75 %; and the only month that has accuracy rate lower than 60 % has accuracy rate 50 %. Given that only four daily prices (open, close, high, and low) are used in the experiment, it is reasonable to expect that the performance of the proposed KNN++ would be further improved if more information is available for usage.

## 5 Further Discussion on KNN++

Now, let's examine the time complexity of the KNN++ method. Assume the size of the training dataset is  $N$ , and the data has  $M$  views. For each test instance, we need to find its  $K$  nearest neighbours as its representative on each view. The complexity of this process is  $M * K * N$ ; Then on each view, we need to find the  $K$  nearest neighbours for each of the  $K$  nearest neighbours of the test instance in order to calculate the weight of this view. The complexity of this process is  $M * K^2 * N$ . Since both  $K$  and  $M$  are just small constant values, the process for finding the  $K$  nearest neighbour by the KNN+ method is linear to the size of the training data set. Furthermore, the KNN++ method can be easily parallelized over the  $M$  processes; and the process of searching the  $K$  nearest neighbors from the training data set can be easily implemented with the MapReduce or Spark framework.

If the size of the training set is really large, some other strategies can be applied to filter out those instances that are obviously not close to the test instance. Taking the task of predicting stock open price as an example, we can first filter out those instances with close prices on the opposite side of certain moving average line to the unknown instance for predicting.

## 6 Conclusion

In this paper, we proposed an enhanced KNN approach, which is denoted as KNN++, for classifying complex data with heterogeneous views. Any type of views can be utilized when applying the KNN++ method, as long as a distance function can be defined on that view. In other words, a distance function that is defined on a view serves as the representation of that view for the KNN++ method.

Given an unknown instance, the KNN++ method learns to weight each view by examining its  $K$  nearest neighbors found by the distance function defined on that view. Each instance of the  $K$  nearest neighbors of the unknown instance by that distance function will search its own  $K$  nearest neighbors by using the same distance function, in order to count how many of its  $K$  nearest neighbors actually have the same class label as this instance. The final weight of the distance function is the aggregation of such numbers across all  $K$  nearest neighbors of the unknown instance by that distance function. The final  $K$  nearest neighbors of the unknown instance will be selected from all different  $K$  nearest neighbors of the unknown instance found by different distance functions with factoring in the weights that are learned for those distance functions.

Experimental study show that the proposed KNN++ method can effectively predict up or down movement for SPY daily open price, based on historical SPY daily open, close, high, and low price data. As part of the future work, we will try to further improve the stock prediction performance by incorporating different types of stock information, such as the sentiment information obtained from stock-oriented social networks such as stocktwits.com. We will also apply the KNN++ method to other applications with complex data such as Alzheimer's early detection.

## References

1. Young, A., Oxtoby, N.P., Schott, J.M., Alexander, D.C.: Data-driven models of neurodegenerative disease. *Adv. Clin. Neurosci. Rehabil. (ACNR)* (2014) <http://www.acnr.co.uk/2014/12/data-driven-models-of-neurodegenerative-disease/>
2. Džeroski, S., Panov, P., Ženko, B.: Machine learning, ensemble methods in. In: Meyers, R.A. (ed.) *Computational Complexity Theory, Techniques, and Applications*, pp. 1781–1789. Springer, New York (2012)
3. Woźniaka, M., Grañab, M., Corchadoc, E.: A survey of multiple classifier systems as hybrid systems. *Information Fusion* **16**, 3–17 (2014)

4. Ženko, B.: Is Combining Classifiers Better than Selecting the Best One. *Mach. Learn.* **54**, 255–273 (2004)
5. Wolpert, D.: Stacked generalization. *Neural Networks* **5**(2), 241–259 (1992)
6. Cover, T., Hart, P.: Nearest neighbour pattern classification. *IEEE Trans. Inf. Theor.* **13**(1), 21–27 (1967)
7. Bhatia, N., et al.: Survey of nearest neighbour techniques. *Int. J. Comput. Sci. Inf. Secur.* **8**(2), 302–305 (2010)
8. Kercheval, A.N., Zhang, Y.: Modelling high-frequency limit order book dynamics with support vector machines. *Quant. Financ.* **15**(8), 1315–1329 (2015)
9. Wang, J.L., Chan, S.H.: Stock market trading rule discovery using two-layer bias decision tree. *Expert Syst. Appl.* **30**(4), 605–611 (2006)
10. Kara, Y., Boyacioglu, M.A., Baykay, Ö.K.: Predicting direction of stock price index movement using artificial neural networks and support vector machines: the sample of the Istanbul Stock Exchange. *Expert Syst. Appl.* **38**(5), 5311–5319 (2011)

Hybrid Intelligent Systems

15th International Conference HIS 2015 on Hybrid  
Intelligent Systems, Seoul, South Korea, November  
16-18, 2015

Abraham, A.; Han, S.Y.; Al-Sharhan, S.A.; Liu, H. (Eds.)  
2016, XIII, 316 p. 110 illus., 39 illus. in color., Softcover  
ISBN: 978-3-319-27220-7