Financial Series Prediction: Comparison Between Precision of Time Series Models and Machine Learning Methods

Xin-Yao Qian, Shan Gao

June 6, 2017

Abstract

Precise financial series predicting has long been a difficult problem because of unstableness and many noises within the series. Although Traditional time series models like ARIMA and GARCH have been researched and proved to be effective in predicting, their performances are still far from satisfying. Machine Learning, as an emerging research field in recent years, has brought about many incredible improvements in tasks such as regressing and classifying, and it's also promising to exploit the methodology in financial time series predicting. In this paper, the predicting precision of financial time series between traditional time series models and mainstream machine learning models including some state-of-the-art ones of deep learning are compared through experiment using real stock index data from history. The result shows that machine learning as a modern method far surpasses traditional models in precision.

Keywords: Financial series predicting, Machine Learning, ARIMA, Deep Learning

1 Introduction

Financial time series prediction is an interesting topic that has been researched broadly. Prediction results can effectively assist investors in deciding investment strategies, which can contribute to higher profit earning.

Among traditional predicting method, there are several classical models which have proved to be useful. For example, *Box-Jenkins ARIMA* makes use of information underlying in lag terms of variable itself and errors in the past; *GARCH* can capture variability of variance effectively to help judge volatility of stock return. Also, there are many other derivative models like *nonlinear GARCH(NGARCH)*, *integrated GARCH(IGARCH)*, *exponential GARCH(EGARCH)*, which can performs well in situations with different settings.

With the fast development of computing and storing ability of computers, *machine learning (ML)* has been an emerging study field belonging to Computer Science, and has brought many great enhancements in tasks concerning predicting within all kinds of fields. ML mainly takes use of artificial algorithms(e.g. artificial neural network) to learn "pattern" underlying given data, and derive a model with ability to predict on new data with the same structure. ML shows outstanding performance in predicting using algorithms most of which have been proved effective rigidly in mathematics. So, it's also quite exciting and promising if ML can be exploited in financial time series predicting.

Also, *deep learning(DL)* has been a increasingly popular subfield of ML in recent years because of its surprisingly outstanding performance on precision in many fields such as computer vision, automatic speech recognition, natural language processing and so on. It mainly consists of various derivatives of *artificial neural network*, which includes models like Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Deep Belief Network (DBF) and so on. Therefore, we will also introduce some state-of-the-art DL models in later comparison in predicting precision.

Later sections of this paper mainly consists of four parts:

(i) Review of previous research work on financial time series predicting

(ii) Brief principle introduction of traditional time series models and machine learning models including deep learning models, all of which are to be compared in this paper.

(iii) Experiment of predicting on real financial data using different models and comparison between performance of different models.

(iv) Conclusion and advice for further research.

2 Literature Review

Financial time series forecasting has long been a topic of interest. Traditional mainstream methods to deal with the problem mainly consist of fundamental analysis and technical analysis, while there has been more and more trials with introducing new edge-cutting methods, i.e. machine learning into prediction in recent years.

2.1 Fundamental Analysis

In general, fundamental analysis tries to analyze some macro features a company or business is showing. It based its principles that the market value of a stock tends to move towards its "real value" or "intrinsic value" [6]. For investors who are in favor of fundamental analysis, he/she would collect information that can help assess the prospect of goal company or business, and if he came to the conclusion that the current value of certain company is underestimated, then he would expect the stock value of the company to return to its higher "true value" later. Therefore, he could buy stocks now and got premium in the future. Or if he concluded that the company is overestimated, then he could short sell the stocks. There are some widely used indicators that can help assess a company's true value, such as *The Price-to-Book Ratio*, *Price-to-Earnings Ratio* and so on, which are calculated using public information of the company.

However, fundamental analysis is limited in that it's useful for trend predicting, i.e. it's better to exploit it in assisting to design long-period trading strategy instead of short-period one.

2.2 Technical Analysis

Technical Analysis is an analysis methodology for forecasting the direction of prices through the study of past market data, primarily price and volume [8]. Generally, investors using this methodology would formulate their trading strategy based on some technical indicators calculated by price, volume and time. Various indicators are results of different formula, and investors establishes their own beliefs of the patterns underlying the relationships between the indicators and stock price's movement direction.

2.3 Time Series & Machine Learning Models

Kim made a incipient contribution to the introduction of machine learning models into financial predicting [7], and specifically, he uses Support Vector Machine (SVM) to predict the movement direction of stock price with precision measured by hit ratio (57.831%), and it outperforms Back-Propagation Network (BP Network) (54.7332%) and case-based (51.9793%) reasoning. S.L. Ho compares ARIMA with two kinds of BP Network and Recurrent Neural Network (RNN), and it turns out that both ARIMA and RNN perform well on short-period prediction than BP Network [12], while Iqbal concludes that artificial neural network (ANN) is still the best tool to predict financial time series after that he compares the performance of ARIMA with various ANN models [14]. Hossain compares GARCH(1,1) with SVM and Neural Network on the precision of volatility prediction on six stock markets indices, and it turns out that GARCH(1,1) and SVM are better on the whole than BP Network [3].

2.4 Deep Learning Models

Troiano shows that both Autoencoders (AE) and Restricted Boltzmann Machines (RBM) are effective in feature reduction for later financial series' trend prediction by SVM, while it turns out that AE can be trained faster and is more accurate than RBM [13]. Heaton illustates that AE and Long Short Term Memory Models (LSTMs) can be put into use for many Financial problems such as Factor Models, Default Detection and Event Study [5].

3 Methodology

3.1 Model Introduction

3.1.1 ARIMA(Autoregressive Integrated Moving Average)

ARIMA is a generalization model of ARMA which is a combination of AR process and MA process, and ARIMA is capable of dealing with time series data with feature of non-stationary because of its "integrate" step, which is denoted by "I" in "ARIMA". If a time series is stationary, then we can draw ACF and PACF figure of the series data to help decide the order of ARIMA model. ARIMA's common form can be written as below:

$$y_t = \Sigma_{i=1}^{p} \alpha_i y_{t-i} + \Sigma_{i=1}^{q} \beta_i u_{t-i} + u_t.$$

y_t stands for our goal variable to forecast, and we would fit the above model to the time series data to obtain fitted model, using which we can forecast value of y_{t+1}, y_{t+2} and so on. p and q here denote the orders of AR process and MA process respectively.

3.1.2 LR(Logistic Regression)

The principle of LR is simple though, it's a classification method quite popular in many realistic applications owing to its good performance. The basic model can be written as:

$$y = \frac{1}{1 + e^{-\beta^T Z}}$$

where:

$$\beta = (\beta_0, \beta_1, \beta_2, \beta_3 ..., \beta_n)^T$$

denotes parameter vector, and

$$Z = (x_0, x_1, x_2, x_3 ..., x_n)^T$$

denotes the feature vector. The function $y = \frac{1}{1+e^{-x}}$ in fact represents a distribution function called *logistic distribution*, whose figure is depicted in Figure 1.

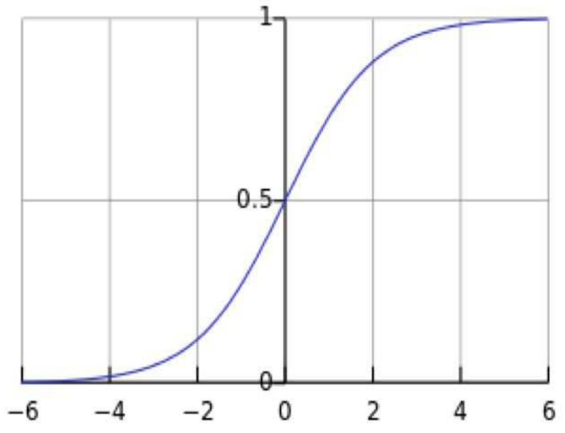

Figure 1: Logistic Curve

LR obeys such classification rule that:

$$classification = \begin{cases} 1 & if\ y > 0.5 \\ -1 & else \end{cases}$$

We fit the logistic function using our training data, which consists of quantities of feature vectors Z_i to obtain fitted parameters β and then use the fitted model to do predicting job. The fitting process normally uses *Maximum Likelihood Estimation.*

3.1.3 SVM(Support Vector Machine)

Vapnik invented SVM and improve the model by introducing kernel method and soft margin later [4]. For sample points which belong to only two classes denoted by feature vectors in R^2 space, SVM tries to find a best straight line that can rightly classify most of data points. As is shown in Figure 2, all L_1,L_2 and L_3 can divide two classes of data points correctly, but L_2 tend to be the "best" as it helps to achieve maximum distances from both nearest data points to L_3, the classification line w.r.t. two classes, which would equip the line with strong ability of generalization, i.e. strong power to classify other data points not given in training set correctly. In general, SVM can be extended to high-dimensional space(R^n), within which data points also become high-dimensional feature vectors, while SVM would try to find a *hyperplane* to do the similar job illustrated above.

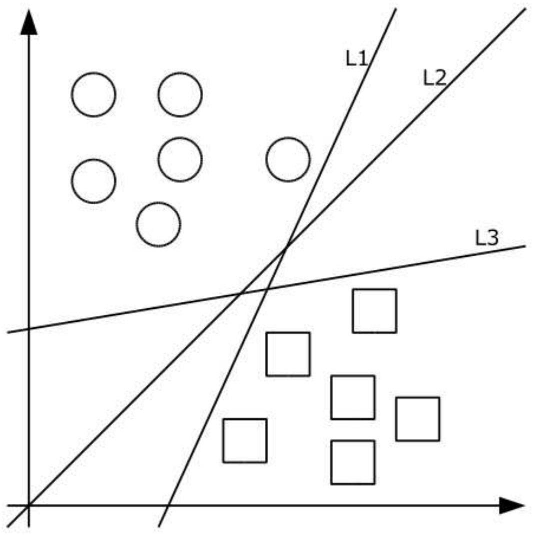

Figure 2: SVM's division of data points

Kernel method used in SVM allows it to deal with classification job concerning non-linear relation between class and feature vector of sample, which makes SVM a powerful tool in non-linear relation discovering. Besides, different from many traditional ML models who try to minimize *empirical risk*, i.e. the training error on training set, SVM tried to minimize *structural risk*, a combination of both Empirical Risk and *complexity* of model, which effectively avoids the problem of overfitting, which means that fitted model can perform well on training set but bad on test set.

3.1.4 MLP(Multilayer Perceptron)

MLP is a kind of *artificial neural network*, a series of model partly inspired by working principles of human brain. It fits a non-linear function(mapping) from input variable vector to output value vector. It's essentially a directed graph consists of layers of *neurons* linked with each other. Every link between neurons has a weight. As is shown in Figure 3, every circle stands for a neuron, while a line of neurons forms a *layer*. Layers in MLP can be mainly classified into input layer, hidden layer and output layer according to their different role. Input layer accepts input variable vector, and output layer outputs the result of processing input vector. As for hidden layer(s) between input layer and output layer, they accept vector from output by input layer or last hidden layer, and output vector to next hidden layer output layer, the process of which can be seen as finding correct mapping function. Each layer is fully connected to next layer, while within each layer there is no any connection. Except for neurons in input layer, every neuron in other layers has a non-linear *activation function*, which normally takes *sigmoid form*:

$$y(v_i) = tanh(v_i)\, or\, y(v_i) = (1 + e^{-v_i})^{-1},$$

where v_i denotes linear weighted inputs for every neuron.

MLP is especially outstanding in finding non-linear relationship between input and output, which makes it a quite useful tool in many jobs like classification and regression.

3.1.5 DAE(Denoising Autoencoders)

DAE is a variant of AE(Autoencoder), a deep learning model. AE *encodes* input variables to a high-level representation and then tries to *decode* high-level representation to input variables, i.e. its original form. The high-level representation is believed to be a good simplification for inputs including too many redundant

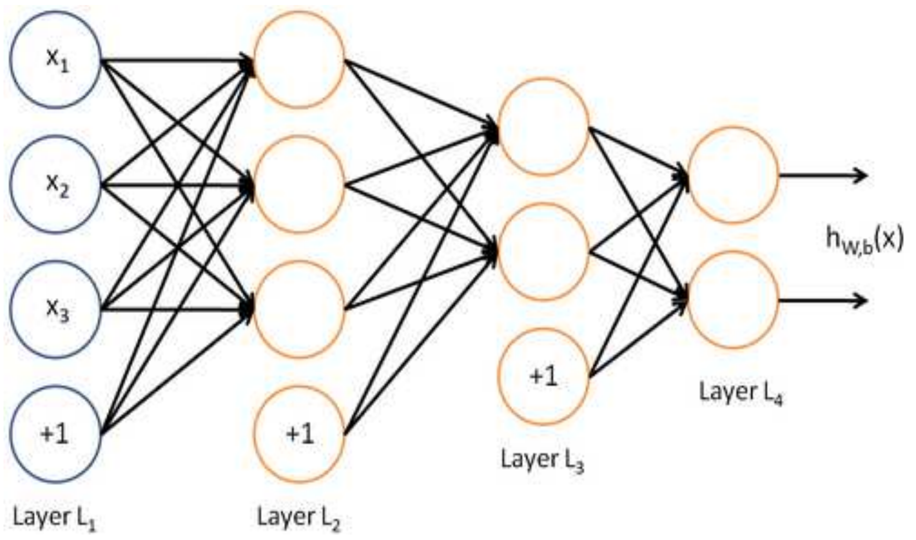

Figure 3: Example framework of multilayer perceptron

information, and is actually proved to be of better performance in replacing original input vector in later jobs [9]. As is shown in Figure 4, there can be many layers of hidden units, which means we can use AE to obtain feature representation that is abstract enough and vital enough to clear noises and redundant information contained in inputs. DAE adds *denoising* process to make AE more robust by randomly corrupting input vectors during training process, so that model is encouraged to capture the statistical dependencies between the inputs.

3.2 Models for Comparison

Inspired by Kim's classical work [7], here we also set our goal on assessment of precision of different models to predict the close price movement direction of next trading day.

Our models include ARIMA, LR(Logistic Regression), MLP(Multilayer Perceptron), SVM, DAE-SVM. While ARIMA only needs to extract information from price time series, machine learning models and deep learning models need to take in certain features into model training, which can include features aside from price time series. Therefore, we decide to take technical indicators used in [7] as input features, which are listed in appendix A.

4 Experiment

4.1 Experiment Setting

We wrote our program in Python and ran it on PyCharm. ARIMA model is realized in Python's open-source tool: statsmodels [11]. All machine learning models are realized in sklearn [10]. DAE's source code comes from Deep Learning [2].

Here, we use tools included in sklearn to tune parameters of machine learning models to obtain models with greatest power of generalization.

4.2 Research Data

We choose S&P 500, Dow 30 and Nasdaq indices as our predicting data, which are representative international stock price indices and have been taken as experiment object in many similar research. We collect indices data from Yahoo.Finance [1], whose time period ranges from 2012.1.2 to 2016.12.26 (Figure 5). Our data includes open price, close price, high price, low price and trading volume for the day. Taking short-time significant volatility into consideration, our unit time period is one week. We directly use close price data in ARIMA models, while performing normalization and technical extracting on all price data until they can be put into ML models' training process. We split training data into two parts randomly. One part (70%) is for

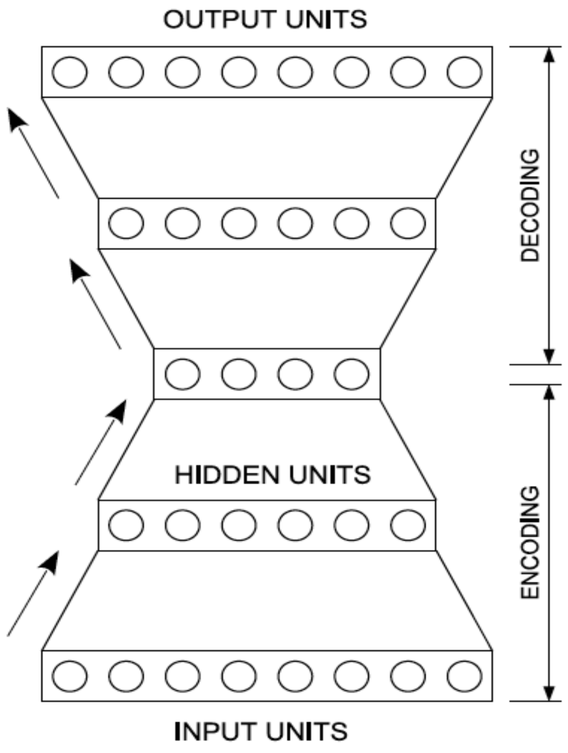

Figure 4: Simple Framework of Autoencoder

training the model, while the other part (30%) is for testing the precision of predicting, which is a common practice to divide data and test the performance of ML model.

4.3 Performance Measurement

Here, we use *hit ratio* to measure the precision of prediction, whose formulation is showed below:

$$hit\,ratio = \frac{\Sigma_{i=1}^{N} prediction_i}{N},$$

and

$$prediction_i = \begin{cases} 1 & prediction_i * real_i > 0 \\ 0 & else \end{cases}$$

where $predcition_i$ denotes the prediction of i th sample's price, and $real_i$ denotes the real value of that. When $prediction_i * real_i > 0$, it means prediction value and real value have same sign, which is regarded as a "hit".

4.4 Results

The hit ratio of different models are showed in Table 1 and Figure 6 below:

ARIMA	LR	MLP	SVM	DAE-SVM
0.593	0.623	0.566	0.642	0.691

Table 1: Best hit ratio of of different models

Hit ratio of all models lies in range of o.5 to 0.7,w hile DAE-SVM and SVM performs better than all other models on different indices, and DAE-SVM also surpasses single SVM completely, with highest hit ratio of 0.691 on Nasdaq prediction.

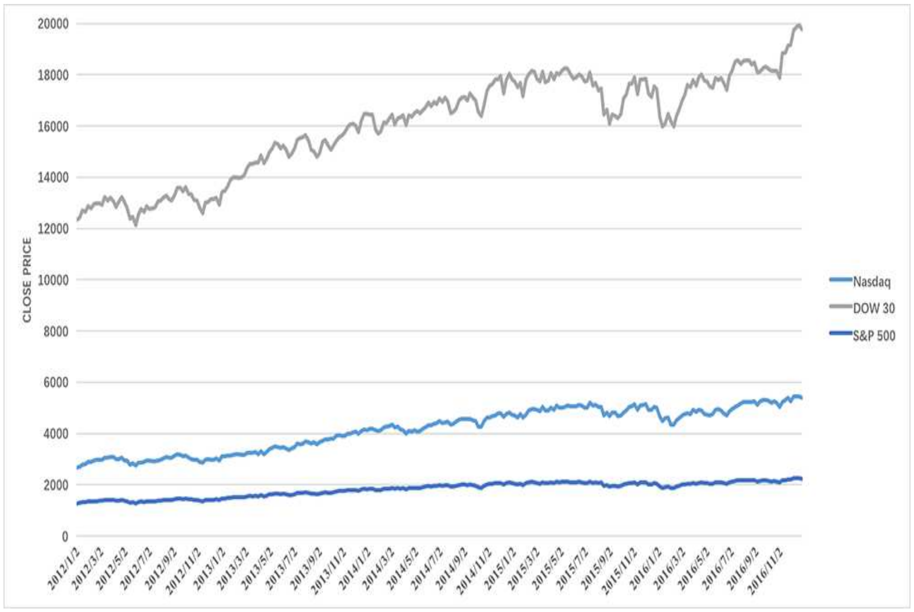

Figure 5: Weekly Close Price Series of Three Indices

5 Conclusion

Our experiment results confirm the results obtained by Kim [7] again that SVM is outstanding in price movement prediction, and also prove that DAE, as one of emerging deep learning model and both unsupervised 'deep' feature extractor, is outstanding in assisting with supervised learning model to enhance its performance. Both ARIMA and LR model performs better than MLP on the whole though, they all fall behind both SVM and DAE-SVM much.

There are some variants of SVM proved to be quite effective than normal SVM, which may be tested in later research. Besides, MLP here performs fairly bad, which may be attributed to the complicated tricks during process of parameter tuning, and it may achieve great improvement if dealt with cleverly. Also, as is mentioned above, there are many other deep learning models such as LSTM that can be tried on financial series' prediction and is promising to get excellent performance.

References

[1] Yahoofinance. https://finance.yahoo.com/.

[2] Deep Learning 0.1. http://www.deeplearning.net/tutorial/da.html.

[3] M. Nasser M. Mufakhkharul Islam Altaf Hossain, Faisal Zaman. Comparison of garch, neural network and support vector machine in financial time comparison of garch, neural network and support vector machine in financial time series prediction. *Pattern Recognition and Machine Intelligence*, 2009.

[4] Corinna Cortes and Vladimir Vapnik. *Support-Vector Networks*. Kluwer Academic Publishers, 1995.

[5] J. B Heaton, N. G Polson, and J. H Witte. Deep learning in finance. 2016.

[6] Dr. A. K. Mittra J. G. Agrawal, Dr. V. S. Chourasia. State-of-the-art in stock prediction techniques. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 2, 2013.

[7] Kyoung jae Kim. Financial time series forecasting using support financial time series forecasting using support vector machines. *Neuro Computing*, 2003.

[8] Julie R. Kirkpatrick, Charles D.; Dahlquist. *Technical Analysis: The Complete Resource for Financial Market Technicians*. Financial Times Press, 2006.

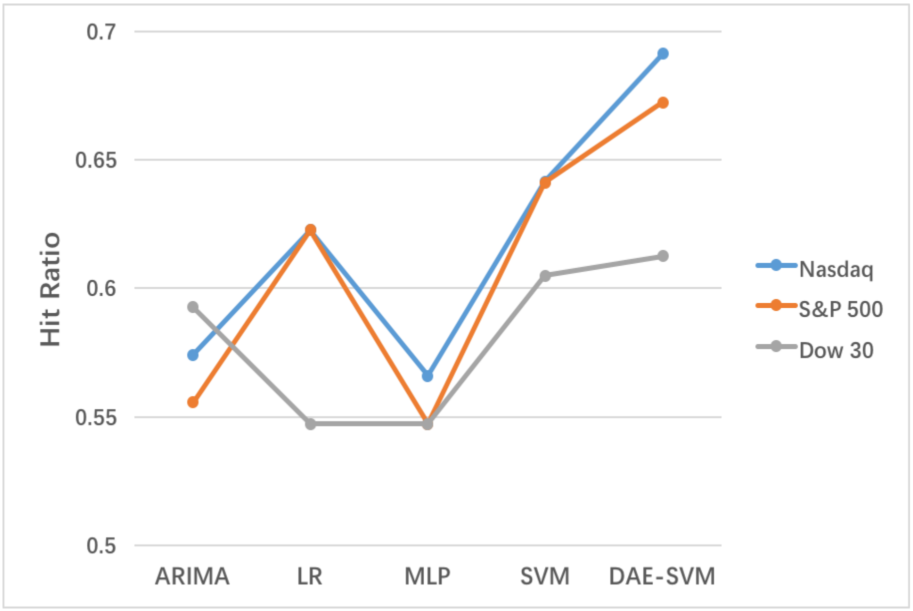

Figure 6: Hit Ratio of Different Models

[9] Bernhard Schölkopf, John Platt, and Thomas Hofmann. Efficient learning of sparse representations with an energy-based model. In *Advances in Neural Information Processing Systems*, pages 1137 – 1144, 2006.

[10] scikit learn. http://scikit-learn.org/stable/.

[11] Skipper Seabold and Josef Perktold. Statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

[12] T.N. Goh S.L. Ho, M.Xie. A comparative study pf neural network and box-jenkins arima modeling in time series prediction. *Computers and Industrial Engineering*, 2002.

[13] Luigi Troiano, Elena Mejuto, and Pravesh Kriplani. On feature reduction using deep learning for trend prediction in finance. *Papers*, 2017.

[14] W. Shahzad Z. Mahmood Zahid Iqbal, R. Ilyas and J. Anjum. Efficient machine learning techniques for stock market prediction. *International Journal of Engineering Research and Applications*, 2013.

A Technical Indicators for Training

Feature name	Description	Formula
%K	Stochastic %K. It compares where a securitys price closed relative to its price range over a given time period	$\frac{C_t - LL_{t-n}}{HH_{t-n} - LLt - n \times 100}$ where LLt and HHt mean lowest low and highest high in the last t days, respectively.
%D	Stochastic %D. Moving average of %K.	$\frac{\Sigma_{i=0}^{n-1} \%K_{t-i}}{n}$
Slow %D	Stochastic slow %D. Moving average of %D.	$\frac{\Sigma_{i=0}^{n-1} \%D_{t-i}}{n}$
Momentum	It measures the amount that a securitys price has changed over a given time span.	$C_t - C_{t-4}$

ROC	Price rate-of-change. It displays the difference between the current price and the price n days ago.	$\frac{C_t}{C_{t-n}} \times 100$
Williams %R	Larry Williams %R. It is a momentum indicator that measures overbought/oversold levels.	$\frac{H_n - C_{t-1}}{H_n - L_n} \times 100$
A/D Oscillator	Accumulation/distribution oscillator. It is a momentum indicator that associates changes in price.	$\frac{H_t - C_{t-1}}{H_t - L_t}$
Disparity5	5-day disparity. It means the distance of current price and the moving average of 5 days.	$\frac{C_t}{MA_5} \times 100$
Disparity10	10-day disparity.	$\frac{C_t}{MA_{10}} \times 100$
OSCP	Price oscillator. It displays the diEerence between two moving averages of a securitys price.	$\frac{MA_5 - MA_{10}}{MA_5}$
CCI	Commodity channel index. It measures the variation of a securitys price from its statistical mean.	$\frac{M_t - SM_t}{0.015D_t}$ where $M_t = (H_t + L_t + C_t)/3, SM_t = \frac{\Sigma_{i=1}^{n} M_{t-i+1}}{n}$, and $D_t = \frac{\Sigma_{i=1}^{n} \lvert M_{t-1+1} - SM_t \rvert}{n}$.
RSI	Relative strength index. It is a price following an oscillator that ranges from 0 to 100.	$100 - \frac{100}{1 + (\Sigma_{i=0}^{n-1} Up_{t-i}/n)/(\Sigma_{i=0}^{n-1} Dw_{t-i}/n)}$ where U_{p_t} means upward-price-change and D_{w_t} means downward-price-change at time t.