Machine and Deep Learning for Stock Price Prediction: Comparison of Classification and Regression Techniques

Henri Woodcock - HenriWoodcock@gmail.com

Contents

---

1 Experiment

This experiment compares different algorithms for predicting a stocks price and its trend, it also compares the use of classification and regression algorithms for predicting the trend while also analysing other problems like the importance of the amount data and the use of auto-encoders.

This section will start with a literature review to discuss developments in financial time series prediction, then discuss the models to be compared and the tools used to create them, followed by an explanation of the conducted experiment and finally a dicussion on the results.

2 Literature Review

The 21st century has stimulated a huge growth in the accessibility of data and the amount of data collected. This has brought around a rise in Machine and Deep Learning; as defined in Section 2.1, Machine Learning algorithms can automatically detect patterns in data and in this era, with vast amounts of data, algorithms that can automatically analyse data are needed.

In finance, investors can now download years of data for hundreds of securities, from stock prices to accounting ratios, leading to the use of Machine and Deep Learning techniques in finance. Stock price prediction was (and still is to an extent) dominated by the use of statistical time series models like ARIMA, both [1],[2] both found that the ARIMA method is still effective, with [1] getting a mean absolute percentage error as low as 21% for the Nifty 50 index. However, [2] showed that ARIMA models can only produce satisfactory results with short-term predictions this is also supported by the results in [3].

The use of Machine and Deep Learning algorithms to predict financial time series is growing. [4] used SVMs to predict trend movements and was able to achieve a hit ratio of 54% on the Korean stock index over a far greater time frame of 581 steps (20% of the data) instead of a standard 10-20 steps used in ARIMA research. [4] also showed the sensitivity of the SVM parameters in this context. This was then extended by [5] where a denoising auto-encoder was used prior to SVM algorithm and managed to achieve a hit ratio of 67.2%. However, this was tested on far fewer time steps: [6] used a hybrid ARIMA and SVM algorithm which achieved better results than ARIMA, but still was only performed on so few time steps that long-term accuracies could not be seen.

A slightly different approach was performed by [7], which inspired by recent development in Convolution Neural Networks (CNNs) (and the AlphaGo project) for image recognition converted financial time series data into 2D images for a CNN to classify the trend of the Taiwan stock exchange. A hit rate of 57.88% on one of their examples was achieved, but concluded that more data could reduce the difficulty of classification. [8] compared the use of a simple RNN to a LSTM model and found that a simple RNN had a trend classification accuracy of 75%, 1% higher than an LSTM. He also found that implementing strategy used by following trend prediction can lead to profits exceeding buy and hold strategy on the S&P500 index with a prediction on 480 steps (20% of the data).

On the regression side, [9] found that RNNs outperform in prediction compared to feed-forward neural networks. However, RNNs have a far greater training time, which could be a factor in model selection. [10] managed to achieve a mean absolute percentage error as low as 0.71% when predicting 60 future values after training on 1000 examples.

One final important paper is [11] which creates a huge deep learning model trained on a large dataset of billions of orders and transactions [11] for 1000 US stocks. This gives remarkable results in which the authors believe they have uncovered evidence universal price formation mechanism[11] by showing their model can accurately predict prices of unseen stocks after being trained on multiple stocks, outperforming models trained on the singular stock. [11] concludes that creating one universal model is less costly than training thousands of stock specific models.

3 Models for Comparison

The experiment will use 3 classification and 3 regression models as well as auto-encoded variants of these models and so there are 12 models in total.

The models are: SVM (and SVR), FFN (feedforward network), RNN (which uses an LSTM layer). All 12 models have been written in Python. To implement the SVM and SVR models, scikit-learn library [12] is used, and Keras [13] is used to implement the FFN, RNN and auto-encoder.

Data Data used is daily closing price of KPN, AKZA, RAND, VPK, HEIA and the KPN trading volume from the 4th Febuary 2008 to 16th January 2019 in the Amsterdam stock exchange, taken from Yahoo Finance [14], as well as technical indicators applied to the KPN price and trading volume which can be found in Appendix A. The use of multiple stock prices to predict one is inspired by [11]. The use of techincal indicators was inspired by [4, 5] who were able to achieve high results with just the use of techincal indicators.

For the classification data set, the output, y_i at each time step was created as follows (S_t = closing price at time t):

$$y_i = \begin{cases} 1, & \text{if } S_{i+1} > S_i \\ 0, & \text{otherwise} \end{cases}$$

where S_t is the closing price at time t. So the classification problem is attempting to predict the trend of the next timestep. For the regression problem, $y_i = S_{i+1}$ and so the regression models are attempting to predict the actual real value of the stock price at the next timestep.

The data is then split into 6 subsets and setup like K-fold validation for time series, with each subset being 467 time steps.

The input data ($\mathbf{X}$) for each fold is standardised, this is so all features have a mean of 0 and variance of 1, this makes training perform better as the algorithms are not skewed by features which are alot larger than others, for example volume traded $\gg$ close price (the data before preprocessing can be seen in Appendix A. This is done by:

$$\tilde{\mathbf{x}} = \frac{\mathbf{x} - \mu_x}{\sigma_x}$$

The testing data input is standardised using the same μ_x and σ_x as the training data so that model knows how to use the data with respect to the training data.

4 Performance Measures

The models will compare the models using 3 different prediction performances. The first measure is the *hit ratio*. This measures how many times the models predict the next trend correctly. For regression models, the output will be converted into a trend by:

$$\text{trend}_i = \begin{cases} 1, & \text{if } \hat{y}_i > S_i \\ 0, & \text{otherwise} \end{cases}$$

i.e. a 1 is the model predicts that the next days closing price is higher than todays closing price. The hit ratio is calculated as follows:

$$\text{hit}_i = \begin{cases} 1, & \text{if trend}_i = \text{actual trend}_i \\ 0, & \text{otherwise} \end{cases}$$

$$\text{hit ratio} = \frac{1}{m}\sum_{i=1}^{m} \text{hit}_i$$

where m is the number of testing samples.

The other two measures are the mean squared error (MSE) and the mean absolute error (MAE):

$$\text{MSE} = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

$$\text{MAE} = \sum_{i=1}^{m} |y_i - \hat{y}_i|$$

both are used as it can allow for comparison with anomalies. The MSE weights anomalies high and so if a model has a high MSE it can be seen it had a few anomaly results in which the distance from the actual value was high. This is important in stock price prediction because if there is a lot of anomalies the model is risky and should not be used as it could risk investments.

5 Results and Discussion

Hit Rates	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Models			Classification		
SVM	0.49464668094218417	0.5224839400428265	0.5053533190578159	0.5139186295503212	0.5074946466809421
FFN	0.47109207708779444	0.47109207708779444	0.4989293361884368	0.5032119914346895	0.5331905781584583
RNN	0.5010706638115632	0.5053533190578159	0.48822269807280516	0.5010706638115632	0.5546038543897216
AE-SVM	0.5267665952890792	0.5032119914346895	0.5331905781584583	0.5353319057815846	0.5117773019271948
AE-FFN	0.5353319057815846	0.5053533190578159	0.5246252676659529	0.5396145610278372	0.5353319057815846
AE-RNN	0.5203426124197003	0.5460385438972163	0.48822269807280516	0.5267665952890792	0.5139186295503212
			Regression		
SVR	0.5257510729613734	0.5493562231759657	0.4871244635193133	0.4978540772532189	0.5515021459227468
FFN	0.5236051502145923	0.5236051502145923	0.51931330472103	0.4957081545064378	0.4871244635193133
RNN	0.5364806866952789	0.5021459227467812	0.4699570815450644	0.49356223175965663	0.5064377682403434
AE-SVR	0.5021459227467812	0.5085836909871244	0.5364806866952789	0.5236051502145923	0.5150214592274678
AE-FFN	0.5515021459227468	0.5515021459227468	0.4957081545064378	0.5128755364806867	0.48068669527896996
AE-RNN	0.5343347639484979	0.51931330472103	0.44849785407725323	0.5321888412017167	0.48497854077253216

Table 1: The hit ratio results from the experiment, (AE stands for Auto-Encoded so AE-FFN is an Auto-Encoded Feed-Forward Network).

MAE	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Models			Classification		
SVM	0.505353	0.477516	0.494647	0.486081	0.492505
FFN	0.499883	0.512223	0.499994	0.496335	0.492974
RNN	0.496748	0.498501	0.503213	0.507818	0.486838
AE-SVM	0.473233	0.496788	0.466881	0.464668	0.488223
AE-MLP	0.495340	0.495697	0.490096	0.490472	0.495237
AE-RNN	0.497863	0.496855	0.500870	0.498710	0.499280
			Regression		
SVR	0.397346	2.380246	0.400028	0.133774	0.132990
FFN	0.583937	1.231039	0.085746	0.056172	0.028865
RNN	0.388919	1.598820	0.534221	0.070322	0.091938
AE-SVR	0.415982	3.041816	1.106633	0.552469	1.192021
AE-MLP	0.325144	2.939524	0.700055	0.462934	0.665415
AE-RNN	0.542022	2.877192	0.799074	0.351736	0.583431

MSE	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Models			Classification		
SVM	0.505353	0.477516	0.494647	0.486081	0.492505
FFN	0.267794	0.335361	0.28168	0.257908	0.273414
RNN	0.251286	0.346753	0.323996	0.301227	0.253361
AE-SVM	0.473233	0.496788	0.466809	0.464668	0.488223
AE-MLP	0.262679	0.263104	0.262437	0.269021	0.279262
AE-RNN	0.250470	0.248034	0.252261	0.249302	0.249948
			Regression		
SVR	0.218233	9.223313	0.243253	0.027218	0.024404
FFN	0.398000	2.670620	0.010610	0.005356	0.001403
RNN	0.171895	4.643264	0.322304	0.008180	0.014587
AE-SVR	0.267384	11.530005	2.463743	0.454033	1.995171
AE-MLP	0.163384	10.861558	0.676369	0.281788	0.523037
AE-RNN	0.479668	10.426144	0.827516	0.166185	0.410738

Table 2: The Mean Absolute Error (MAE) and Mean Squared Error (MSE) results from the experiment.

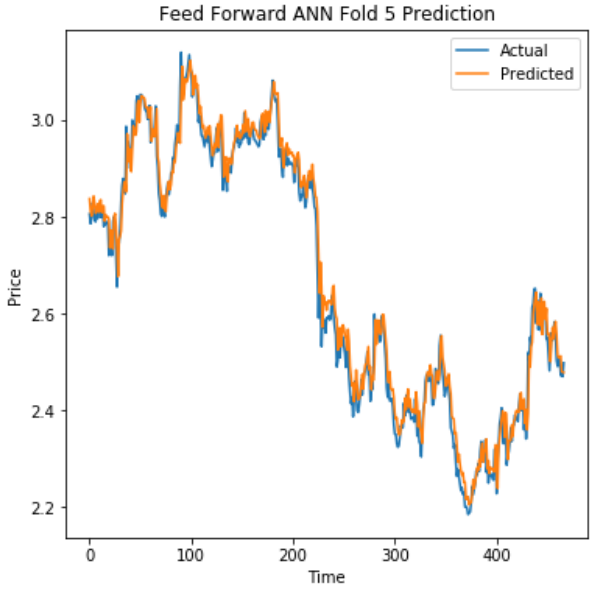
The hit ratio, MAE and MSE results can be seen in Table 1 and Table 2. The highest hit ratio was a classification RNN model at 0.555 in fold5 with a close second of a 0.552% for the Auto-Encoded feedforward network in fold2 and the Support Vector Regression in fold5. The models with the lowest MAE and MSE were all regression models, with the lowest being the feedforward regression network with a MAE of 0.02886502 and an MSE of 0.00140317.

On average, classification models had a higher hit rate, as more training data was used going from an average 0.508 in Fold 1 to 0.526 in Fold 5. This was expected, as can be seen in [15], and relationships begin to be found with increased data. This was mentioned in Section **??**: as data is increased, it becomes easier to generalise, because you learn relationships in more situations. However, in the regression algorithms, the average hit rate actually decreased as the folds increased, suggesting that more data was, in fact, problematic for predicting the trend. However, when looking at the other measurements (mean squared error and mean absolute error), the lowest error was actually in Fold 4. This means that the models were, in reality, incredibly close to predicting the actual price (an MSE of 0.157 and a MAE 0.271) of the stock but not the trend. One surprising result is that in Fold 1 and Fold 2 regression models on average achieved a higher hit rate than the classification models suggesting that if little data obtainable then regression models will perform better. In Fold 1 and 2 regression models achieved an average hit rate of 0.529 and 0.526 respectively compared to the average classifcation hit rates in Fold 1 and 2 which was 0.508 and 0.509 respectively.

The hit rate is a good measurement as it demonstrates how an investor could use these models. If an uptrend is predicted, then the investor will know to buy the stock and can sell it at the next time step for a profit (if the prediction is correct). However, the hit rate makes it difficult to compare models in any other way, as mentioned before. Regression models had a decreasing hit rate over increasing the training size, but the MSE and MAE reached a low in Fold 4, suggesting that more training data was helping prediction. MSE and MAE can be used to assess the models accuracy in predicting its intended output.

It became clear that, as the training data increased, classification models had a decreasing MAE until Fold 4 (similar to regression). That being said, the MSE actually increased with more training data, suggesting that the model was more affected by outliers, which could mean that the models began to overfit with more data. Regression and Classification models achieved similar MAE and MSE on average, yet regression was the most accurate in both of these measurements, suggesting that it is more accurate to predict actual price than price trend.

Another important factor to note is the effect of auto-encoding. To do this, an average is taken for each of the performance measures when auto-encoded and when not auto-encoded for both classification and regression. Doing this showed that, on average, the auto-encoded classification models outperformed the standard models in all three performance measures. Also, it showed that, when the training data increased, the performance increased in both standard and auto-encoded models. However, for regression, the results



fold5.png

Figure 1: The Price Prediction of The Feed-Forward Network vs The Actual Price in Fold 5.

were slightly different: On average, the auto-encoded models outperformed the standard models in the hit ratio in all but Fold 5, suggesting that auto-encoding helps to improve the models accuracy in predicting movements in the price and its ability to predict the trend. On the other hand, the standard models outperformed the auto-encoded models in both MSE and MAE in all five folds. In fact, on average, the standard models were able to achieve a MAE of 0.0867 suggesting that the models were, on average, 0.0867 away from the actual price. Figure 1 of the Feed-Forward Network in Fold 5 shows how close the regression models were to actually predicting the true price.

A reason for the auto-encoder making the performance of regression models worse for the MAE and MSE can be explained by [16]. Regression in time series requires long term dependencies which is not represented through standard auto-encoders. [16] presents a recurrent auto-encoder which can allow for these long-term dependencies, and thus represent relationships over time in the compressed (encoded) data.

Looking at specific models, the best model for the hit ratio was the classification RNN with a hit ratio of 0.555 (3 s.f.) (as mentioned at the beginning), suggesting that, for the standard investor who will simply want to predict an uptrend or downtrend to make profit, a classification algorithm will work best. This also occurred in Fold 5, suggesting that the more data there is, the better. The best MSE and MAE was achieved by the regression feedforward neural network, with 0.0289(3 s.f.) and 0.00140(3 s.f.), respectively. This is incredibly close, and a plot of this can be seen in Figure (insert figure) to show how close this was to predicting the actual price.

6 Conclusion

To conclude, if the user is trying to achieve the highest hit rate to profit from the model, then the best model to use would be a classification RNN. This model performed better as the training data increased so it is assumed that, if using this model, then the more training data there is, the better the outcome will be. However, if little training data is available, then the user may be more inclined to use a regression model: Fold 1, 2 and 3 all had a regression model for the top spot. It should be noted, though, that if the auto-encoded feedforward network or the auto-encoded support vector regression model is used, then the prices predicted should not be taken as an accurate prediction of the actual price. In short, the models are good at predicting the movement (hit ratio) but not the actual price (having a high MSE and MAE).

However, if wanting to predict the actual price, then a classification model cannot be used. This experiment has shown that regression models are able to do this with a high accuracy, and that results improve

with more training data. In this experiment, a regression feedforward network achieved the best results and are far quicker to train than the LSTM due to there being fewer components to learn. However, LSTMs have proved to be more successful in other areas such as in [17] where "state-of-the-art"[17] performance is achieved in acoustic modelling, and so more research needs to be done here to see if their performance can be enhanced by other features or a different architecture to the LSTM used.

Auto-encoding proved to be successful for classification algorithms but not regression algorithms, suggesting more research could be done to find different types of auto-encoders. This could then enhance the performance of regression models in the scenario, like the recurrent auto-encoder mentioned in Section 5.

Finally, this experiment could have been extended to try different stocks, so as to see if these results generalise to many different stocks or whether the results achieved only happen on this specific stock. To see how these results can be useful for investors see Appendix which discusses trading strategies that could be implemented with these results.

References

[1] Uma, D.B., Sundar, D., Alli, P. An Effective Time Series Analysis for Stock Trend Prediction Using ARIMA Model for Nifty Midcap-50. *International Journal of Data Mining & Knowledge Management Process*. 2013, **3**(1), pp. 65-78.

[2] Adebiyi, A., Adewumi, A., Ayo, C. Stock price prediction using the ARIMA model. In: *UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, 26 - 28 March 2014, Cambridge*. Washington: IEEE Computer Society, 2014, pp. 106-112.

[3] Ho, S.L., Xie, M., Goh, T.N. A Comparative Study of Neural Network and Box-Jenkins ARIMA Modeling in Time Series Prediction. *Computers & Industrial Engineering*. 2002, **42**(2), pp. 371 - 375.

[4] Kim, K. J. Financial Time Series Forecasting Using Support Vector Machines. *Neurocomputing*. 2003, **55**(1), pp. 307-319.

[5] Qian, X.Y. and Gao, S. [E-print]. Financial Series Prediction: Comparison Between Precision of Time Series Models and Machine Learning Methods. *arXiv*. arXiv:1706.00948. 2017.

[6] Pai, P.F. and Lin, C.S. A hybrid ARIMA and support vector machines model in stock price forecasting. *Omega*. 2005, **33**(6), pp. 497-505.

[7] Chen, J., Chen, W., Huang, C., Huang, S., Chen, A. Financial Time-Series Data Analysis Using Deep Convolutional Neural Networks. In: *2016 7th International Conference on Cloud Computing and Big Data (CCBD), 16 - 18 November 2016, Macau*. Washington: IEEE Computer Society, 2016, pp. 87-92.

[8] Edet, S. Recurrent Neural Networks in Forecasting S&P 500 Index. *SSRN Electronic Journal*. 2017, Available at SSRN: https://ssrn.com/abstract=3001046 or http://dx.doi.org/10.2139/ssrn.3001046.

[9] Iqbal, Z, Ilyas, R., Shahzad, W., Mahmood, Z., Anjum, J. Efficient Machine Learning Techniques for Stock Market Prediction. *International Journal of Engineering Research and Applications*. 2019, **3**(1), pp. 855-867.

[10] Wanjawa, B. and Muchemi, L. [E-print]. ANN Model to Predict Stock Prices at Stock Exchange Markets. *arXiv*. arXiv:1602.06561. 2014.

[11] Sirignano, J. and Cont, R. Universal Features of Price Formation in Financial Markets: Perspectives From Deep Learning. *SSRN Electronic Journal*. 2018, Available at SSRN: https://ssrn.com/abstract=3141294 or http://dx.doi.org/10.2139/ssrn.3141294.

[12] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011, **12**(1), pp. 2825-2830.

[13] Chollet, F. *Keras* (2.2.0). [Software]. 2018. [Accessed 10th January 2019].

[14] Yahoo! Finance. Koninklijke KPN N.V. (KPN.AS). 30th Jan. *Yahoo! Finance*. 2019. [Online]. [Accessed 5th Feb 2019]. Available from: https://finance.yahoo.com/quote/KPN.AS

[15] Halevy, A., Norvig, P., Pereira, F. The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems*. 2009, **24**(2), pp. 8-12.

[16] Maas, A., Le, Q.V., O'Neil, T.M., Vinyals, O., Nguyen, P., Ng, A.Y. Recurrent Neural Networks for Noise Reduction in Robust ASR. In: *13th Annual Conference of the International Speech Communication Association, 9-13 September 2012, Portland*. Published in INTERSPEECH 2012. Available at: https://www.isca-speech.org/.

[17] Sak, H., Senior, A., Beaufays, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. 2014, **15**(1), pp. 338-342.

A Data

Date	KPN Close	AKZA Close	RAND Close	VPK Close	HEIA Close	Volume	%K 14day	%D 3	Slow %D 3	RSI	ROC	Momentum	OSCP	Disparity 3	Disparity 14	output
04/02/2008	7.49368	57.014999	25.959999	17.059999	37.560001	25915216	90.51742524	83.84348135	64.65266776	50.35984028	100.6514282	0.29708	0.028366613	100.4877992	103.3382978	0
05/02/2008	7.48762	54.360001	24.66	16.465	37.689999	61834147	89.65576078	91.95448061	80.63779771	51.28205128	99.03774551	0.13945	0.0343443	99.78455322	103.2115839	1
06/02/2008	7.59069	54.866199	24.370001	16.594999	36.299999	31237632	100	93.39106201	89.72967466	50.92267991	101.1308573	0.06063	0.036820586	100.8864083	104.601105	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14/01/2019	2.472	79.110001	40.700001	43.34	75.080002	11574857	0	-0.098959	-0.098959	39.13043478	96.94117647	-0.019	-0.013257147	99.21070234	97.89545146	1
15/01/2019	2.473	79.379997	41.130001	43.639999	76.480003	14198900	0.900900901	-0.098959	-0.098959	38.7755102	98.21286736	-0.023	-0.014145996	99.55716586	98.14883062	0
16/01/2019	2.47	79.357498	41.509998	43.66	75.120003	13640391	0	-0.098959	-0.098959	42.3566879	99.51651894	-0.037	-0.017704223	99.93256912	98.16334062	1

Table 3: Classification Data

Date	KPN Close	AKZA Close	RAND Close	VPK Close	HEIA Close	Volume	%K 14-day	%D 3-day	Slow %D 3-day	RSI	ROC	Momentum	OSCP	Disparity 3	Disparity 14	output
04/02/2008	7.49368	57.014999	25.959999	17.059999	37.560001	25915216	90.51742524	83.84348135	64.65266776	50.35984028	100.6514282	0.29708	0.028366613	100.4877992	103.3382978	7.48762
05/02/2008	7.48762	54.360001	24.66	16.465	37.689999	61834147	89.65576078	91.95448061	80.63779771	51.28205128	99.03774551	0.13945	0.0343443	99.78455322	103.2115839	7.59069
06/02/2008	7.59069	54.866199	24.370001	16.594999	36.299999	31237632	100	93.39106201	89.72967466	50.92267991	101.1308573	0.06063	0.036820586	100.8864083	104.601105	7.66344
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14/01/2019	2.472	79.110001	40.700001	43.34	75.080002	11574857	0	-0.098959	-0.098959	39.13043478	96.94117647	-0.019	-0.013257147	99.21070234	97.89545146	2.473
15/01/2019	2.473	79.379997	41.130001	43.639999	76.480003	14198900	0.900900901	-0.098959	-0.098959	38.7755102	98.21286736	-0.023	-0.014145996	99.55716586	98.14883062	2.47
16/01/2019	2.47	79.357498	41.509998	43.66	75.120003	13640391	0	-0.098959	-0.098959	42.3566879	99.51651894	-0.037	-0.017704223	99.93256912	98.16334062	2.498

Table 4: Regression Data

Feature	Description	Formula	Reference
%K 14-day	Stochastic Oscillator. Compares the current closing price relative to its price range over a given time period.	$\frac{S_t - L_{14}}{H_{14} - L_{14}} \times 100$ where L_{14} means lowest price in last 14 days and H_{14} means highest price in last 14 days.	[4]
%D 3-day	Moving average of %K.	$\frac{1}{n}\sum_{i=0}^{2} \%K_{t-i}$	[4]
slow %D 3-day	Moving average of %D.	$\frac{1}{n}\sum_{i=0}^{2} \%D_{t-i}$	[4]
RSI	Relative strength index. It measures the magnitude of recent price changes to evaluate overbought or oversold conditions in the price of a stock.	$100 - \frac{100}{1+(\sum_{i=0}^{13} Up_{t-i}/n)/(\sum_{i=0}^{13} Dw_{t-i}/n)}$ where Up_t means upward-price change and Dw_t means downward price at time t.	[4]
ROC	Price rate of change. It is a ratio of the current price and the price n-days ago.	$\frac{S_t}{S_{t-n}} \times 100$	[4]
Momentum	It measures the difference between todays price and the price n-days ago, used to spot positive or negative trends.	$S_t - S_{t-n}$	[4]
OSCP	It displays the difference between two moving averages of a security.	$\frac{\text{MA}_5 - \text{MA}_{10}}{\text{MA}_{10}}$	[4]
Disparity 3-day	A ratio between the current price and the 3-day MA.	$\frac{S_t}{\text{MA}_3}$	[4]
Disparity 14-day	A ratio between the current price and the 14-day MA.	$\frac{S_t}{\text{MA}_{14}}$	[4]

Table 5: Features and their respective formula.