Predicting the short-term direction of the DAX using the AdaBoost algorithm

Kevin O’Mahony (ID: R00105946)

**Abstract:**Prediction of tomorrow daily close of DAX index relative to todays close using AdaBoost. Comparison of AdaBoost to SVM, Random Forest, and Gradient Boost Trees shows AdaBoost to be the top performer with 80% accuracy for day ahead prediction. Short term forecast of DAX close out to 10 days possible at greater than 70% accuracy level. Models tested on alternative instruments: Precious Metals and Forex, show similar one day ahead prediction results.

1. Introduction

Stock market prediction has long been a focus of research effort. Forecasting future prices is important to many market participants from retail traders, market makers, institutional traders, banks, hedge funds and other investment entities. The global financial system has become ever more interconnected to the degree that local market movements can have immediate impact on markets on the other side of the world as witnessed in the global financial crisis of 2008. Hence it has never been more relevant to have reliable forecasting models. Creating accurate prediction models is difficult due to the complex nature of typical financial time series. They are normally non-stationary and display different seasonal and trend components. Traditional financial time series analysis has typically used combinations of signal decomposition, smoothing, moving average, spectral analysis, auto-regression (ARIMA) and generalized auto-regression conditional heteroskedasticity (GARCH) models with mixed success.

There are generally two approaches taken to financial forecasting: either predict a future actual price or predict a future direction, e.g. closing price today greater than or less than closing price yesterday. These correspond to the two main categories of machine learning: regression, the prediction of a continuous variable versus classification, predicting which class a variable belongs to. Here we will focus on predicting the short-term direction of the closing price of a stock market index relative to today’s closing price. This is in the form of a classification problem i.e. assign a class ‘1’ for today’s close price greater than yesterday’s close price and assign a class ‘0’ for today’s close price less than yesterdays.

Over the last decade machine learning techniques have increasingly been investigated as a means of producing more efficient prediction models. This study will focus on some of the more popular models developed, in particular a form of enhanced decision tree known as AdaBoost [1] and compare its performance to Support Vector Machine (SVM), Random Forest and another boosted algorithm, Gradient Boosting.

AdaBoost (short for adaptive boosting) is from a family of classifiers known as ensemble methods. They work on the principle of increasing classification accuracy by creating multiple classifiers and combining their outputs into a single decision. AdaBoost is a general method for generating a strong classifier out of a set of weak classifiers. It’s first classification attempt runs on an unweighted or equally weighted training data set. The incorrectly classified data is then weighted so that it is more likely to appear as input to subsequent classifiers. As each new weak learner is added it will process increasingly more difficult to classify data. More classifies are generated until a predefined error rate is achieved and the process stops. Classifier’s results are then combined based on an accuracy weighting to produce the final outcome. AdaBoost has been applied to many different classifiers but is most often used with decision trees.

The other models chosen are: SVM, Random Forest and Gradient Boosted Machine. SVM [2] performs classification of non-linear problems by mapping the data to a higher dimension space where the classes become linearly separable. The mapping is performed by transforming the data via a kernel function. We will examine three popular non-linear kernel functions: sigmoid, radial basis and polynomial. Random Forest [3] is an ensemble method using a collection of decision trees. Each tree is built using a bootstrap sample from the data set. While a standard decision tree split’s each node using the best split amongst all the features, a Random Forest will split each node based on the best from a random set of features. Gradient Boosted Machines [4] are another ensemble method where a collection of models are trained in order to minimize the error loss function using gradient descent.

The DAX is used in this study as a representative stock index. It has a large daily range and is more volatile than most other European stock indices. It is an index of the 30 largest German companies trading on the Frankfurt Stock Exchange. It was created in 1988 with a base value of 1000. The DAX components represent roughly 75% of the total market capitalisation of the Frankfurt Exchange.

The rest of this paper is organised as follows. The next section discusses some recent related research. Section 3 describes the data set and Section 4 discusses the experimental setup and results. Finally, Section 5 details the conclusions and possible future work.

2. Related Research

There is a wide range of recent research on the application of ensemble methods to stock market prediction. Ballings et al. [5] benchmarked the performance of Random Forests, AdaBoost and Kernel Factory[[1]](#footnote-1) against single classifier models. They used data on 5767 publicly listed European companies for one year ahead price predictions and found Random Forests to be the top performer with the three ensemble methods ranked in the top four. They recommended that new studies should include ensemble methods in their test set of algorithms. Yutong and Zhao [6] used AdaBoost to classify Shanghai A-share stocks as good or bad based on their historical returns along with other fundamental factors. The basic AdaBoost algorithm selected stocks that generated higher than average returns over the following year. Zhou et al. [7] applied a modified gradient boosted model for short-term prediction of price movement. Their simple trading system utilising the model generated significant net average returns of 0.045% per trade. The ensemble technique has also seen other hybrid approaches. Narayanan [8] combined AdaBoost with Support Vector Machine and also Naïve Bayes for stock price prediction. It was showed that accuracy and error rate were improved over the single classifier performance. Rodriguez [9] looked at predicting the direction of the daily index close for three exchanges: IPC (Mexico), KLSE (Malaysia) and Bovespa (Brazil) from 1990 to 2003. A number of classifiers where used including, Neural Network, Random Forest and Gradient Boosted Machine. The Random Forest model was the only classifier to perform better that random for all three exchanges.

3. Model Detail

***3.1 Data Set***

The data set consists of a combination of daily price and volume data downloaded from GKFX.com [10], some generated time related features and basic technical indicators. The GKFX data is made up the following features:

|  |  |
| --- | --- |
| **Open** | Opening price for the day |
| **High** | Highest price for the day |
| **Low** | Lowest price for the day |
| **Close** | Closing price of the day |
| **Volume** | Total volume traded (via GKFX) |

The following features are added in an attempt to capture some of the time related behaviour of the price data.

|  |  |
| --- | --- |
| **DayOfMonth** | Day of the month (1 – 31) |
| **DayOfWeek** | Day of the week (0= Mon…6=Sun) |
| **WeekOfYear** | Week of the year (1 – 52) |

The technical indicators were added to capture some idea of the direction and trend of the daily price movement.

|  |  |
| --- | --- |
| **dirn** | 1 = close > open, 0 = close < open |
| **ma0\_trend** | 1 = close > EMA[[2]](#footnote-2)(close, 10), 0 = close > EMA(close, 10) |
| **ma1\_trend** | 1 = close > EMA(close, 20), 0 = close > EMA(close, 20) |
| **ma2\_trend** | 1 = close > EMA(close, 50), 0 = close > EMA(close, 50) |
| **ma3\_trend** | 1 = close > EMA(close, 100), 0 = close > EMA(close,100) |
| **close\_at\_high** | 1 = close price equal to highest price of day, otherwise = 0 |
| **close\_at\_low** | 1 = close price equal to lowest price of day, otherwise = 0 |

We want to predict the price trend i.e. is today’s close price greater than or less than yesterday close price. We use the class variable, trend to capture this.

|  |  |
| --- | --- |
| **trend** | 1 = close > yesterdays close, 0 = close < yesterdays close |

In total there are 15 features. The data set covers the period 7/5/2011 to 13/04/2017, which is 2031 trading days.

4. Empirical Evaluation

The Scikit Learn Python library was used to evaluate our models. Scikit Learn provides a wide range of machine learning models for both classification and regression. The Python library TA-Lib [11] was used to create the technical analysis indicators used as input features to the models.

Our task is to predict if the DAX closing price for one to ten days into the future is greater than or less than the close price for today using the data set and models discussed above.

***4.1 Experimental Procedure***

There are orders of magnitude differences in the ranges of our features e.g. dirn = [0,1] versus close = [5000,12,300]. All models and algorithms used are sensitive to varying feature ranges so all input features are first rescaled into the range [0,1].

Looking at boxplots for the all price data reveals no outliers, e.g. below is the plot for the Dax daily close price from 2011 to 2016. We note there are also no missing values in the price data.

Table 1. Boxplot for Daily Close Dax prices (2011 – 2016)

|  |
| --- |
|  |

Cross fold validation is the main method used model parameter tuning and testing. N-fold cross validation will split the data set into N-folds, N-1 folds being used for training and the Nth used for testing. Each fold will be used in turn as the test fold e.g. fold 5 might be used as test, with folds [1-4,6-10] used as training for a 10-fold test. This method can’t be applied to our data set as the price data is time order dependent. We can’t train on future prices and then test on past data. Therefore, we use a walk forward style of train/testing. This involves training for a specified number of days, predicting for the nextday(s) after the last day of training and then moving the training window forward one day and repeating the train/test cycle.

To select the number of days to use for the training window we run the AdaBoost model with default parameters multiple times using training days from the range [5,55]. We used data from 2014 to 2015 for the test and looked at one day ahead prediction of close trend. Figure 2. shows there is a sharp drop-off in performance when using anything more than 5 days training. This makes intuitive sense the best predictor of price is the most recent price behaviour. We select 3 days as the optimum training period.

Model parameter tuning methods provided in Scikit involves use of cross fold validation to validate each combination of parameters tested. As this is not an option for this time series data set a basic trial and error approach was used to examine the parameters of the AdaBoost model.

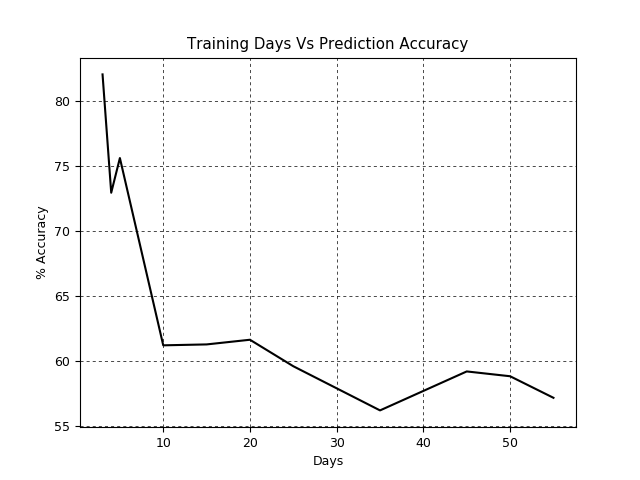
******

Figure 2. Training Days versus one day ahead prediction accuracy for AdaBoost model

The AdaBoost model implemented in SciKit has two parameters of interest:

* n\_estimators: the maximum number decision tree classifiers created before boosting is stopped (default = 50)
* learning\_rate: reduces the contribution from each classifier by this factor when combining predictions (default = 1)

Varying n\_estimators over a range of [25,200] with default learning rate showed minimal changes in performance. A value of 25 gave the best prediction accuracy.

Learning rate likewise showed slight impact on performance when varied from the default value. From this test, we select n\_estimators=25 and learning\_rate=1 for the AdaBoost model.

We looked at feature selection using a Random Forest model’s importance rating based on its out-of-bag test process. We used data from 2010 to 2013. Figure 1. shows the results based on the feature numbering:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 DayOfMonth | 5 high | 9 dirn | 13 ma3\_trend |
| 2 DayOfWeek | 6 low | 10 ma0\_trend | 14 close\_at\_high |
| 3 WeekOfYear | 7 close | 11 ma1\_trend | 15 close\_at\_low |
| 4 open | 8 volume | 12 ma2\_trend |  |

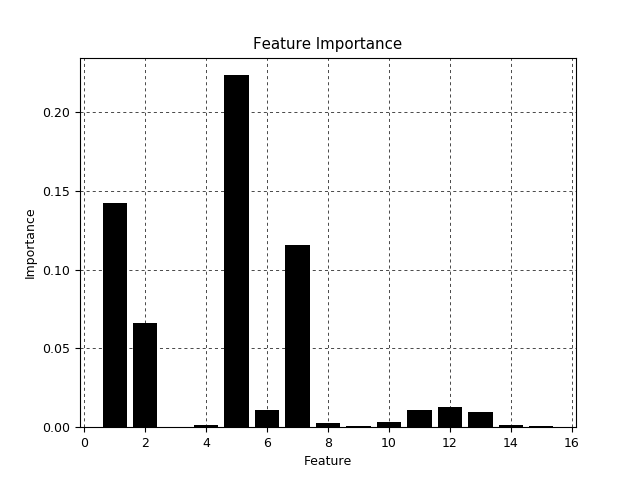


Figure 3. Feature importance reported by RandomForest model.

We see that the most dominate features are DayOfMonth, DayOfWeek, daily high and close. Features contributing very little include: WeekOfYear, dirn, close\_at\_high and close\_at\_low. It would appear that removing any of these features would have practically no effect on the prediction outcome. However, when any combination of these features was removed there was a significant (> 4%) reduction in prediction accuracy. Combined with the small data set size and relatively small feature count it was decided to retain all features.

***4.2 Results***

The results from running a train/test cycle for 1/1/2016 through to 31/12/2016 on the DAX using a training period of 3 days produced the average forecasts as shown in tabular and graphical format below.

Table 1. Average short term forecasts for DAX index for 2016

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Days**  **Ahead**  **Forecast** | **Ada**  **Boost** | **Support Vector Machine** | **Random**  **Forest** | **Gradient**  **Boost** |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10** | 80.29  84.56  82.96  79.10  86.57  83.70  81.34  79.85  75.76  73.28 | 70.07  66.18  71.11  70.15  69.40  67.41  66.42  70.90  67.42  63.63 | 70.80  71.32  68.15  66.42  68.66  68.89  71.64  67.16  66.67  62.59 | 77.37  68.38  67.41  67.16  69.40  68.89  75.37  78/36  73.48  57.25 |

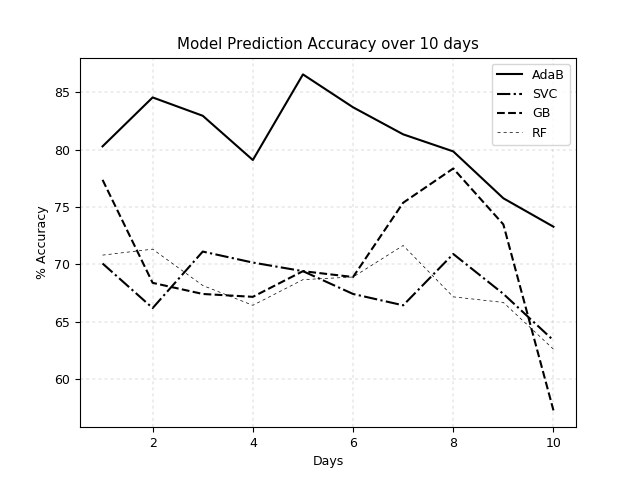


Figure 3. Average short term forecasts for DAX index for 2016

Two additional markets were tested using the same default model parameters and training period as before to see if the approach could be generalized. Two foreign exchange instruments, the British Pound versus the Japanese Yen and the US Dollar versus the Japanese Yen plus two precious metals: spot Gold and Silver prices were used. The one day ahead prediction accuracies achieved were comparable to the results from the DAX.

Table 2. Average one day ahead forecasts accuracy for additional markets for 2011 - 2016

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Instrument** | **Ada**  **Boost** | **Support Vector Machine** | **Random**  **Forest** | **Gradient**  **Boost** |
| **GBPUSD**  **USDJPY**  **GOLD**  **SILVER** | 76.53  76.44  80.91  77.82 | 65.60  68.06  70.19  67.92 | 72.92  74.46  75.32  72.97 | 78.15  78.03  78.32  83.88 |
| **Average** | 77.92 | 65.94 | 73.92 | 79.59 |

***4.3 Interpretation***

The AdaBoost model is the clear winner from the 2016 prediction results shown in Figure 3. There is very little to separate the remaining three models results. The Gradient Boosting just about performs better than the other two when we consider all of the forecast horizons. All models show one or more peaks in accuracy after the first day e.g. between day 7 and 9 all models bar AdaBoost show a new peak.

The AdaBoost model performs significantly better than the rest when we look at the 5 and 10 day ahead forecast prediction accuracies. The accuracy drops below 80% for the first time at day 8.

Running the models on different markets produced comparable results. When we average the one day ahead prediction accuracies on the four new markets shown in Table 2. we see that the AdaBoost model comes in second behind Gradient Boosting. Both of these models showed similarly strong one day ahead figures for the DAX test. However, the fact that the overall results are similar provides confidence that the approach can be generalised to other markets.

**5. Conclusions and Future Work**

Tradable predictions for the direction of the DAX closing price from one to ten days into the future have been produced using the AdaBoost model. Prediction accuracies of 80% were achieved up to 7 days out from the end of the training period. Test results for 2016 show that AdaBoost outperformed SVC, RandomForest and Gradient Boosting. Similar results were obtained for one day ahead predictions for GBPUSD, USDJPY, Gold and Silver price data.

Future work on this topic could include:

* Looking at predicting closing price direction for a lower timeframe e.g. four or one hour intraday closing price.
* Investigating viability of using regression models to predict future daily closing price.
* Investigate using Extreme Gradient Boosting – a new evolution of gradient boosted decision trees designed for speed and performance. It could be applied to either the classification or regression of closing price.

References

[1] Y. Freund and R. Schapire, “A desicion-theoretic generalization of on-line learning and an application to boosting,” *Comput. Learn. theory*, vol. 55, pp. 119–139, 1995.

[2] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A Training Algorithm for Optimal Margin Classiiers,” *Proc. fifth Annu. Work. Comput. Learn. theory*, pp. 144–152, 1992.

[3] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[4] J. H. Friedman, “Greedy function machine: A gr,” *Statistics (Ber).*, vol. 29, no. 5, pp. 1189–1232, 2011.

[5] M. Ballings, D. Van Den Poel, N. Hespeels, and R. Gryp, “Evaluating multiple classifiers for stock price direction prediction,” *Expert Syst. Appl.*, vol. 42, no. 20, pp. 7046–7056, 2015.

[6] S. Yutong and H. Zhao, “Stock Selection Model Based on Advanced AdaBoost Algorithm,” no. Icmic, pp. 1–7, 2015.

[7] N. Zhou, W. Cheng, Y. Qin, and Z. Yin, “Evolution of high-frequency systematic trading: a performance-driven gradient boosting model,” *Quant. Financ.*, vol. 15, no. 8, SI, pp. 1387–1403, 2015.

[8] B. Narayanan, “Prediction of Stock Market using Ensemble Model,” vol. 128, no. 1, pp. 18–21, 2015.

[9] P. Rodriguez and A. Rodriguez, “Predicting stock market indices movements,” *Comput. Financ. its …*, 2004.

[10] “Platforms|GKFX.” [Online]. Available: https://www.gkfx.com/Platforms. [Accessed: 02-Apr-2017].

[11] J. et al. Benediktsson, “TA-Lib.” [Online]. Available: https://mrjbq7.github.io/ta-lib/index.html. [Accessed: 02-Apr-2017].

1. Kernel Factory is an ensemble method where each base classifier of a random forest is fit on the kernel matrix of a subset of the training data. It is not used in this study. [↑](#footnote-ref-1)
2. EMA is exponential moving average. EMA(close,10) indicates a 10 period exponential moving average of the data item, close. [↑](#footnote-ref-2)