Machine Learning for Technical Stock Analysis

F R E D R I K C E D E R V A L L

KTH Computer Science
and Communication

Master of Science Thesis
Stockholm, Sweden 2012

Machine Learning for Technical Stock Analysis

F R E D R I K C E D E R V A L L

DD221X, Master's Thesis in Computer Science (30 ECTS credits)
Degree Progr. in Computer Science and Engineering 300 credits
Master Programme in Computer Science 120 credits
Royal Institute of Technology year 2012
Supervisor at CSC was Stefan Carlsson
Examiner was Stefan Carlsson

TRITA-CSC-E 2012:088
ISRN-KTH/CSC/E--12/088--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication

KTH CSC
SE-100 44 Stockholm, Sweden

URL: www.kth.se/csc

Abstract

Technical analysis has been applied to the stock market for over a century. In recent times, the technical analysis has become faster and more powerful because of computer-aided calculations and visualizations. The challenge was to take this *financial engineering* one step further and make it intelligent and automating stock price forecasting.

In this master's thesis, machine learning is applied to generate forecasts with a time-horizon of approximately 4.5 days, for stocks listed at Nasdaq OMXS30. A wkNN and a modified HMM were implemented. The latter failed to generate any usable forecasts, but the wkNN generated a hit rate better than chance (with high confidence). The forecasts from the wkNN were aggregated together with news-based forecasts, but the performance did not improve.

Keywords: technical analysis, stock forecast, machine learning, wkNN, HMM

Referat

Maskininlärning för teknisk aktieanalys

Teknisk analys har applicerats på aktiemarknaden i över ett århundrade. På senare tid har den tekniska analysen blivit allt snabbare och kraftfullare på grund av datorstödda beräkningar och visualiseringar. Utmaningen var att ta denna *financial engineering* ett steg vidare och göra den intelligent och automatisera prognostisering av aktiepris.

I detta examensarbete appliceras maskininlärning för att generera prognoser med en tidshorisont på ungefär 4.5 dagar, för aktier listade på Nasdaq OMXS30. En wkNN och en modifierad HMM implementerades. Den senare misslyckades att generera några användbara prognoser, men wkNN:en genererade en hit rate bättre än slumpen (med hög konfidens). Prognoserna från wkNN:en aggregerades med nyhetsbaserade prognoser, men prestandan förbättrades inte.

Nyckelord: teknisk analys, aktieprognos, maskininlärning, wkNN, HMM

Preface

This report is a part of a degree project within computer science and autonomous systems, performed at the School of Computer Science and Communication at the Royal Institute of Technology (KTH), in collaboration with Nordnet.

Many thanks to my supervisor Stefan Carlsson (KTH) who helped me through the academic guidelines, keeping me focused on the defined problem, working within the scope of the master's thesis.

A great tribute to Tommi Lahdenperä, Klas Ljungkvist, and Petter Alvsten at Nordnet, who believed in the idea and showed great enthusiasm, creativity and support throughout the process. Unforeseen issues were remedied, which was the key to enable completion of the project.

Fredrik

Contents

Chapter 1

Introduction

Stock traders try to come up with the magic recipe to beat the market. Many concepts, divided into technical analysis[1] and fundamental analysis[2], are well known, but the tweaks making them successful remain unknown, shrouded in mystery. The transition from manual number crunching goes beyond automated, computational ditto. Intelligent systems might be able to perform the actual analysis as well. The power of an autonomous system with aggregated analysis sources, e.g., stock price analysis *and* news analysis, is yet to be explored academically. Such system could perform rapid scanning of thousands of stocks, instantly making forecasts upon arrival of news articles, picking out interesting investments. Or, it could perform more profound analysis on selected stocks, finding subtle patterns where single-sourced, manual analysis models would perform weakly. The result would be significantly better investments.

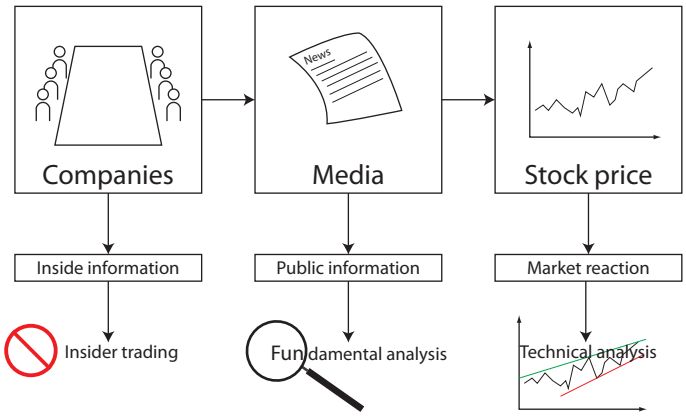


Figure 1.1: Stock price influence: actions, information flow, and reactions.

[1]Technical analysis (TA) is a discipline within analysis of securities, forecasting stock price movements based on historical time-series of stock price and traded volume.

[2]Fundamental analysis (FA) means evaluating securities and forecasting their value based on the overall economy, industry condition, financial condition and management of companies.

1.1 Problem Description

It is said that it is impossible to beat the stock market, due to the claim that it is efficient; all known information is directly incorporated in the price (Fama, 1970). This hypothesis has been challenged by the findings of systematic abnormal returns (Cedervall and Elgh, 2011) and intuitively so; otherwise there would be no raison d'être for equity funds besides index funds. Stock markets are not only based on fundamental financial facts, forecasted discounted cash flows, and rational decisions thereof; they are biased by emotions (Ackert *et al.*, 2003).

Deterministic machines repeat patterns. Humans repeat behaviors. This is the inductive fundament that technical analysis is based on: if A preceded B several times before, B is likely to happen now when A has occurred. A is an indicator that B will happen. The challenge is to find and match the patterns A and B, and the relationship between them. (Rockefeller, 2011)

In classical technical analysis, visual, conspicuous patterns in stock price and traded volume constitute the framework. The appertain theories are derived from the over hundred years old *Dow Theory*, introduced by the *Wall Street Journal* founder and *Dow Jones & Company* co-founder Charles H. Dow (McDonald, 2007). Except effects occurring *due* to the fact that the visual patterns appear—self-fulfilling prophecies, boosted by technical analysis advocates acting on those signals—would there not be effects indicated by subtle, concealed patterns?

The challenge is to extract these subtle, combinatorial patterns from real-time stock data and produce a continuous stock price forecast; automatically. On top of that, by aggregating several analysis sources, representing more of the influencing factors at the same time, a more precise forecast could be drawn. Using techniques from pattern recognition and machine learning could be an appropriate way to achieve this.

1.2 Delimitations

The system must...

- ...be based on stocks listed at Nasdaq OMX Nordic Stockholm[3]
- ...use the nExt API[4] for retrieval of stock data
- ...be written in JAVA
- ...not include any third party software delimiting commercial use
- ...aggregate at least one other analysis module, preferably news analysis
- ...output JSON formatted data through a socket

[3]Nasdaq OMX Nordic Stockholm was previously called Stockholmsbörsen
[4]Nordnet External API

1.3 Goal and Purpose

The goal is to develop an autonomous system for automatic real-time stock price forecasting, which empirically can be proven accurate. Being *accurate* is however not univocal: it could be either discrete or continuous. In the discrete case, only forecasting *up* or *down*, generating a hit rate[5] accuracy beating chance. Successful results require that this accuracy also have high statistical significance ($\alpha = 0.01$). The continuous case is harder. Accuracy could mean getting a small enough expected error and variance. However, if the error would yield better returns than forecasted (e.g. if the stock goes up 2 % when forecasted 1 %), the error impact is positive. That is, having high correlation between the forecast and the outcome, even though the amplitudes might be far off. In the end it comes down to one fundamental thing: is it possible to make money on this? To be able to evaluate that, a trading strategy must be applied, which is outside the scope of this report.

By aggregating two autonomous analysis sources, the goal is to achieve better predictions than using the analyses separately. Successful aggregated results are reached if the accuracy is increased.

In addition to this, the goal is to determine if one machine learning and pattern recognition paradigm is better than the other, in performing the technical analysis of stock time-series data and generating a forecast.

The purpose of developing a system reaching these goals is to enable automatic stock price forecasting with better performance for private investors, professional traders and autonomous stock trading robots. It could facilitate innovative products and services, taking stock trading to another level for the mass-market.

1.4 Definitions

Instrument General term for a financial security, which could be a stock, fixed-income security, option or any other type of derivative.

OHLC The abbreviation stands for *Open, High, Low, Close*, referring to the pricing of an instrument during a given timespan.

Stock tick A stock tick is a quantized timespan (e.g. 1 minute) including OHLC, turnover and volume of the specific stock during that timespan. When presented in a graph, a stock tick is normally called a *bar* or a *candlestick* (depending on the graph type).

[5]Hit rate is the ratio of correct decisions (i.e. trades with positive returns) compared to available decisions (i.e. all trades made)

1.5 Collaboration

The research is performed in close collaboration with Johannes Elgh, writing his master's thesis *Machine Learning News Analysis for Stock Trading*. The non-module-specific sections—regarding the system as a whole, and the results thereof—will be shared between the two reports, and collaboratively written. Cross-referencing between the two reports will also occur.

1.6 Methodology

With the basis in a positivistic perspective, the research is performed based on quantitative data. The ambition is to find empirical evidence for abnormal returns, using analyses facilitated by methods from the discipline of autonomous systems.

1.7 Disposition

For those who are not familiar with the financial theory constituting the foundation of technical analysis (of securities), a brief introduction to the subject initiates the *Theory* part (Page 7). Succeeding the financial theory, a high-level toolbox of machine learning and pattern recognition techniques is presented for those not familiar with the area. At this point, the reader should have an overall understanding. The theory part is completed with previous research within the financial area with applied machine learning and pattern recognition (*Automatic Stock Analysis and Trading*, Page 11).

The second part of the report, *Method and Implementation* (Page 17), starts off by describing the angle of contact of the problem; based on the knowledge from the theory part, how is the problem going to be solved? To get an understanding of the concept of the system, a system overview is presented (Chapter 6, Page 19). This will give a general understanding of the system as a whole, its external integration points and its intermodular relations. Further, the implementation and training of the system and its modules is described.

The finishing part of the report consists of the experiments performed and the results thereof together with discussions around them.

Part I

Theory

Chapter 2

Financial Theory

The premise that this research relies on is that technical analysis works; at least to some extent. For that to be true, stock markets must be driven by more than fundamental facts, which is intuitively true. That would imply that the market is *not* efficient, at least not in its strong form.

The hypothesis that market efficiency reigns in the stock markets, introduced by Fama (1970), has been challenged by (among others) Cedervall and Elgh (2011) and Enke and Thawornwong (2005). Even the father of the *Efficient Market Hypothesis*, Eugene Fama, concludes, and provides evidence for, that stock markets are predictable by means of publicly available data, most importantly time-series data and macroeconomic variables.

Many studies assume a linear relationship between the historical time-series data and the outcome, but there is no evidence supporting that the relationships would actually be strictly linear. Instead, several studies establish the existence of non-linear patterns and relationships. (Enke and Thawornwong, 2005)

Studies support that it is possible, to some extent, to predict future index or stock returns with results that are better than random (Lo and MacKinlay, 1988). Technical analysis has been proven to help making such stock price predictions (Deng *et al.*, 2011) and applying technical analysis in trading strategies have generated profits (at least) until the early 1990's (Park and Irwin, 2007).

As always within economics, proved discoveries are not absolute facts; no two stock markets are equal, and they change over time. What has been shown to apply to a given market at a given time, might very well not apply to another market or another time. This was shown by Tian *et al.* (2002), providing support for technical analysis in the Chinese stock market but finding no support in the U.S. equity market. Since different stocks behave differently and different trends in the market affect our behavior differently, methods proven successful during certain circumstances might be obsolete during others (Rockefeller, 2011). And not to forget: the timespan! Since psychology is an important factor, it is reasonable that it has a bigger effect in the short-term, while fundamental, financial facts take over when the timespan increases.

The idea behind technical analysis is that price discounts everything and that history repeats itself (Kotick, 2003). Any situation, state of information and psychology is consolidated into the price. This is done over and over again, creating patterns, repeated under similar circumstances. Classical technical analysis rely on recurring visual patterns (e.g. *double bottoms*, *head-and-shoulders* and *flags*), indicators (e.g. *moving average* and *relative strength index*), trend line theory and support and resistance levels (Li and Tsang, 1999; Prasad, 2005).

Trend analysis is of great importance since the outcome of a formation depends on the trend it occurs in (Kotick, 2003). Also, the power by which the price curve exits a formation or breaks through a support or resistance depends on the momentum in the stock. Increases in volume can be seen as momentum indicators, quality marks, giving strength and significance to the current price change. Hence, trading volume is seen as one of the most important indicators in technical analysis (Raei *et al.*, 2011). However, volume by itself is uninteresting and does not add any value until it is combined with correlated variables (such as price) (Blume *et al.*, 1994). The consensus included in higher volume indicates that the price change is driven by information; driven by fundamental values (Abbondante, 2010).

Even the most exhaustive technical analysis does not result in much of an actual forecast. A crude price estimation and a vague timeframe together with an indicator of when to take position is normally the case. Investors are more interested in retrieving simple trading decisions (buy/hold/sell) than getting stock price predictions with varying deviation (Chang *et al.*, 2009).

"I have seen the future and it is very much like the present, only longer."

– Kehlog Albran

Chapter 3

Machine Learning

Within the discipline of autonomous systems, several computer science techniques are gathered. The system needs to be able to draw conclusions and classify observations. Observations are input from arbitrary sensors or other data sources, or aggregations thereof. To get the raw observations to make any sense and becoming manageable, characteristics of what to be evaluated need to be isolated. This is called feature extraction.

A group or sequence of features defines an entity. An entity can either be labeled or not. Collections of labeled entities can be used for classification, while unlabeled entities can be used to find relationships without the relationship being pre-defined (by belonging to a group defined by the label). Machine learning is the framework to enable a computer to find these relationships in observations; depicting the reality in a way that conclusions can be drawn for observations in the future.

Different applications put different requirements on the machine learning framework to be used. Speed and space are such. But maybe more importantly, to be able to represent the depicted environment in a way that the best performance is reached in the conclusions. Here follow rudimentary descriptions about how a small selection of different machine learning models work.

3.1 Artificial Neural Network (ANN)

Artificial Neural Networks (ANN) try to resemble the properties of the networks of biological neurons and connecting synapses constituting our brains. The input nodes (dendrites) take quantified input values. Each dendrite has a weight, defining the importance of that input signal in the decision mixture. Between the dendrites and output nodes (axons), one or more hidden layers of nodes (neurons) are connected with edges (synapses). Each neuron has a threshold function and each synapse has a weight. All together, a complex combinatorial rule set is constituted, enabling non-linear statistical modeling.

3.2 *k*-Nearest Neighbors (*k*NN)

One intuitive machine learning method is the k-Nearest Neighbors (kNN). The idea is that an n-dimensional feature vector and a label of its class represent each entity. When classifying a previously unseen entity, its prototype feature vector is placed in the feature space, and the k nearest (e.g. Euclidean distance) vectors in the trained feature space are picked out. The most common class in that set, determines the class of the tested prototype.

Since system training is performed just by adding feature vectors to a collection, training is really fast. The side-effect is that testing—classification of unseen entities—is slow. This is due to the fact that distance calculations have to be performed for every feature vector in the trained feature space, to find ones closest to the prototype feature vector.

3.2.1 Weighted *k*-Nearest Neighbors (w*k*NN)

Some of the k nearest vectors in the feature space might be very far from the prototype to classify, thus irrelevant, and given a disproportionate impact on the result. This is mostly an issue if the k value is set relatively high, in relation to the general sparsity of the feature space. One example is when the feature space contains a regularly distributed, overrepresented class that suppresses small, high-density clusters of other classes.

In regard of this, Kozak *et al.* (2005) propose a distance-weighted variant: weighted k-Nearest Neighbors (wkNN). However, Liu and Chawla (2011) claim that wkNNs, based on either multiplicative inverse or additive inverse distance weights, do not solve the issue regarding class imbalance in a kNN feature space, due to the fact that the dominating class might overlap regions.

3.3 Hidden Markov Model (HMM)

A Hidden Markov Model (HMM) is an encapsulated Markov chain where the state of the model is not observable. Thus, the model, or rather the state sequence in the Markov process, is said to be hidden. Instead, each state has a stochastic output value. Different output distributions give indications of the likelihood of having a specific underlying state generating an observable output.

The system being modeled must be assumed to be a Markov process; fulfilling the Markov property, being memoryless. A memoryless stochastic process implies that the conditional probability of a future state depends only on the present state, ignoring the preceding state sequence.

The properties of a HMM makes it suited for modeling time-series data containing noise, variations and uncertainty, such as voice recognition, song identification, handwriting and bioinformatics.

Chapter 4

Automatic Stock Analysis and Trading

A lot of research has been done in forecasting stock price and price movement (Chang *et al.*, 2009), in this common area of research called *financial engineering* (Chai and Taib, 2010). However, most research within automatic stock analysis (Section 4.1) focus on predicting the direction of a succeeding price change, instead of forecasting an actual price. Such predictions are easy to interpret as buy or sell signals, easily adaptable for automatic trading (Section 4.2).

4.1 Stock Analysis

4.1.1 Neural Networks

ANN is by far the most applied machine learning technology within automatic stock analysis (Enke and Thawornwong, 2005; Chai and Taib, 2010; Raei *et al.*, 2011). Raei *et al.* (2011) state "neural networks has been one of the most important techniques for forecasting stock prices". ANN has successfully been used to effectively find the non-linear relationships existing in the stock market. However, those studies do not present any benchmarking or attempts using alternative techniques (Enke and Thawornwong, 2005).

Also using ANN, da Silva Soares *et al.* (2007) generate continuous forecasts for stocks at the São Paulo stock exchange (BOVESPA). With sixty delay dendrites (input nodes), sixty neurons in the hidden layer and forty or hundred-fifty axons (output nodes), forecasts for forty or hundred-fifty days were made, respectively. When applying wavelet transform and adding fundamental analysis variables, the performance increased. Example forecasts that were presented implied good results, but were not given any statistical significance. Starting out as a trading system, Tan *et al.* (2008) developed an ANN-based system that gained predictive capabilities, benchmarked to have up to five days forecasting capability, with high significance.

A widely used ANN implementation for automatic stock analysis and prediction is the multi-layer feed-forward neural networks with back-propagation (Enke and Thawornwong, 2005). One advantage is that an ANN does not require the non-linear model to be specified before estimation. According to Enke and Thawornwong

(2005), no studies had practically applied generalized regression neural networks (GRNN), which are based on non-linear regression theory. Raei *et al.* (2011) state that already in the early nineties, GRNN were shown to provide better non-linear estimations than other neural network models.

4.1.2 Training Data and Parameter Selection

Raei *et al.* (2011) evaluate three different GRNN models, trained on different parameters, where a one-year forecast is the outcome of a one-year data sequence. In excess of the closing price of each timespan, the selection of aggregated technical analysis parameters (moving average, MFI, OBV, RSI, price differences, etc.) was different. Their study showed that with 95 % confidence, there was no performance difference between the models. At the same time, their implementation of the special form of neural network was shown not to have any positive impact on the return on investment.

Kannan *et al.* (2010) attempt to predict whether the closing price will be higher or lower tomorrow than today. This was done by combining several common technical analysis methods: Typical Price (TP), Chaikin Money Flow indicator (CMI), Stochastic Momentum Index (SMI), Relative Strength Index (RSI), Bollinger Bands (BB), Moving Average (MA) and Bollinger Signal.

One could argue that combining the well-known technical analysis parameters is redundant when applying machine learning. Since they are all based on the same time-series of stock price and traded volume, the pattern recognition and machine learning model should be able to find the multidimensional, non-linear patterns from that raw data. It is another thing when exercising technical analysis manually, where these aggregated values fulfill their purpose as visual and mathematical key indicators.

The study performed by Enke and Thawornwong (2005) aim for identifying which parameters (mainly macroeconomic ones) to use for forecasting. This was executed by creating several ANNs and introducing an information gain technique to evaluate the predictive relationships of machine learning for data mining. The combination of input variables for a model was shown to affect the performance significantly. Unfortunately, the actual results, the fifteen out of thirty-one variables identified to have the most significant impact on the prediction, were not presented.

Studies in the forecasting area often use long-term data for training. For instance, the study performed by Enke and Thawornwong (2005) was based on monthly data over almost 24 years (286 periods). During such long timespan, several global factors, trends and behaviors change, making macroeconomic factors more prominent. To get a perspective of the behavior change, five years ago we did not have high-frequency trading (HFT) robots acting on the stock market. Twenty years ago you were not able to execute your own trading online, but had to call your stockbroker. The speed has changed. The psychology has as well. The information balance is different, now when anybody can get the latest news online, in real-time, for free.

Trends and cyclical variations should be eliminated from the training data; otherwise the model will learn those patterns (making it obsolete in other trends etc.) (Enke and Thawornwong, 2005; da Silva Soares *et al.*, 2007).

4.1.3 Evaluation

Studies show that stock trading based on forecasting with a small error performs worse than trading based on accurate predictions of just the sign (up or down) of a stock (Enke and Thawornwong, 2005). This raises the question regarding *how* to evaluate a stock price forecast: having the lowest deviation, the highest correlation or just enabling abnormal returns when used for trading?

Evaluating the accuracy of a forecast by its deviation from the actual outcome may not be adequate from a practical trading perspective. If the outcome outperforms the forecast, the return would just be better. If the outcome underperforms, even just by a little, it could mean that you actually lose a lot of money. Altogether, reasonable forecasts, determined by deviation, might not be a good predictor of the direction of change in the pricing of an instrument. (Enke and Thawornwong, 2005)

The aggregated algorithm described by Kannan *et al.* (2010), was said to perform "better than chance" with high significance level. However, they defined *chance* to 50/50. Since the stock market have had a positive return in the long term, the definition of *chance* should not be 50/50 between increase and decrease, but instead having the discrete intra-day distribution of the index as the null hypothesis.

In the case of evaluating hit rate, it is important to state how to treat the case of unchanged price since it infers five ambiguous cases ([+,0], [-,0], [0,0], [0,-], [0,+]). When talking about trading, a zero trade is a losing trade, but when evaluating a forecast that would be unfair. Hellström (1998) simply ignores all cases where the forecast or the outcome is zero. This makes benchmarking easier.

4.2 Trading

In contrast to the long-term scope of the stock analysis studies, the trading studies and implementations have a super short perspective. Also, the trading is executed with little or none actual forecasting, but instead implements simple pre-defined static strategies (Feng *et al.*, 2004) or dynamic strategies driven by buy or sell signals derived from technical analysis (Li and Tsang, 1999).

When executing automatic trading, the essence is to formulate trading decisions: buy cheap, sell expensive and to do that often. The short-term includes volatility of high frequency. If it is possible to find local minima and maxima, it is also possible to outperform the "buy and hold" strategy (Tan *et al.*, 2008). Chang *et al.* (2009) use neural networks to find these turning signals (local peak or valley) to maximize the price difference in short-term trading.

One way to automatically find trading strategies, is to apply genetic algorithms (GA) to breed algorithms in several generations, selecting the most successful individuals for further breeding (Raei *et al.*, 2011).

"Prediction is very difficult, especially if it's about the future."

– Nils Bohr

Part II

Method and Implementation

Chapter 5

Method

The practical part of the degree project consisted of two major challenges; building the prototype system and evaluating it. A complete prototype system with live real-time analyses and an offline evaluation mode was built.

The system contains two different analysis modules of which one is the price pattern analysis module. In the price pattern analysis module, two different machine learning paradigms were implemented: a wkNN and a HMM. These were evaluated relative to each other and the best one was used in the final system to create aggregated forecasts.

To be able to test the system efficiently (and effectively), an evaluation routine was built. The routine was able to perform cross-validation to ensure statistical relevance. This was achieved by implementing a k-fold routine. The time dependent results from the system evaluation runs were finally statistically evaluated.

"Prophesy is a good line of business, but it is full of risks."

– Mark Twain

Chapter 6

The Vanir System

The prototype system, named Vanir[1], aggregates stock forecasts from attached modules, trying to achieve better performance than the individual forecasts separately.

Aggregated stock forecasts are driven by the publication of news articles. Vanir subscribes to several news sources through Nordnet FeedAPI (Section 7.1). When a news article is published, Vanir receives a subscription notification. The article is directly downloaded in its whole via the Nordnet REST API (Section 7.1), and the aggregator is invoked to make a forecast. In turn, the aggregator calls the attached modules to retrieve the two forecasts from the analysis modules. The aggregated forecast is broadcasted to all clients connected to VanirAPI.

As can be seen in the component diagram (Figure 6.1), the Vanir system has four major components: the two analysis modules, VanirAPI and the nExt API integration. VanirAPI handles the internal delegation and aggregation. The nExt API is the Nordnet External API, supplying stock data and news. A more detailed description of the nExt API can be found in Section 7.1.

In Chapter 9, the price pattern analysis module is described and in Chapter 10 the aggregator is described. Note that the implementation of the news analysis module is not discussed in this report.

[1]Vanir is a group of gods in Norse mythology, masters of sorcery and magic, especially known for their talent to predict the future.

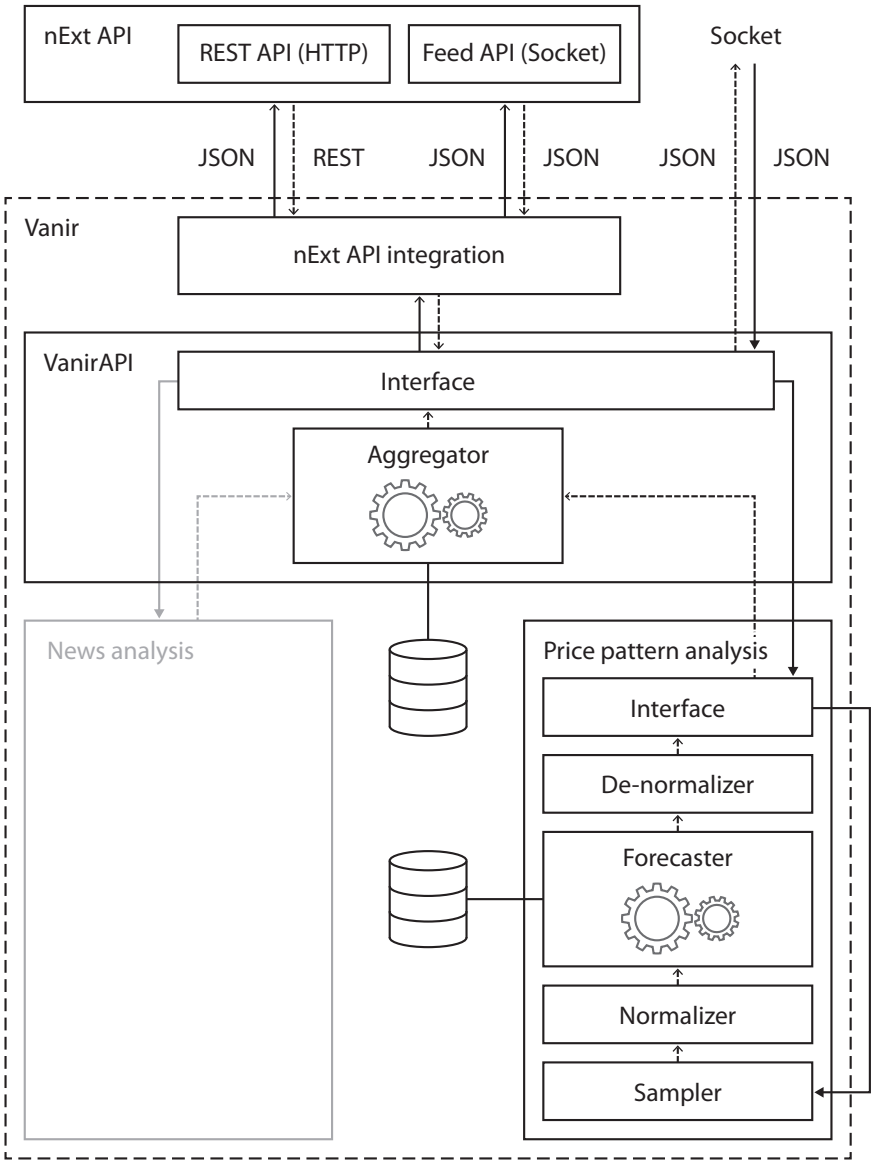


Figure 6.1: Concept diagram of the Vanir system.

Chapter 7

Data Acquisition

To train and evaluate the system, historical stock prices are needed. The concerned stock lists are the ones provided by Nasdaq OMX Nordic Stockholm. To allow a short perspective with fine-grained patterns and trends, high-resolution data is required: minute-long stock ticks containing OHLC, turnover and volume.

High-dimensional machine learning models generally require an extensive amount of training data. To be able to perform cross-validation, even more data is needed.

Stock price data was to be supplied by Nordnet, both historical data and real-time data. Historical data is needed for training, while real-time data can be used for unbiased real-time evaluation and create practical use of the system. However, it turned out that Nordnet only store low-resolution historical data (daily OHLC prices) instead of the high-resolution data that was needed for this project.

Turning directly to the source, Nasdaq OMX offers data through Nasdaq OMX Global Data Products. As a university department, an academic license was available. Unfortunately, the Swedish law regarding freedom of information legislation prohibits fulfilling the agreement. Thus, signing the agreement was declined.

Every good plan includes a backup plan. Long before the setbacks described above, a data feed recording feature was built into the Vanir system, enabling generation of the high-resolution stock ticks needed. The system subscribed to real-time stock trades through Nordnet FeedAPI (see Section 7.1), which were aggregated into stock ticks that were serialized and stored permanently. Acquiring stock data in this way significantly limits the timespan available for training and evaluation, which could affect the results and the evaluation of them.

After about one month of recording, an acquaintance was able to help getting hold of the needed data. A chart data dump with minute-resolution for the stocks included in the OMXS30 index for whole of 2011 was retrieved. To get historical news articles was no problem. Nordnet save these and were able to hand out a dump from their database. News articles published during 2011 concerning the companies in the OMXS30 index from the two sources Nyhetsbyrån Direkt and SIX News were retrieved.

7.1 Nordnet External API (nExt API)

The Nordnet External API (nExt API) was used for acquisition of real-time stock data and news articles. Two interfaces are included in the nExt API: the request-based (poll data) REST API and the subscription-based (push data) Feed API.

On the Vanir side, a nExt API integration layer was implemented, translating nExt API messages into application adapted objects. Also, concurrency handling of requests and subscriptions within the Vanir system was included in the integration layer, which allows for multiple internal subscriptions to the same feed.

"The best qualification of a prophet is to have a good memory."

– Marquis of Halifax

Chapter 8

Sampling

The time t_0 from which a forecast is desired is continuous and the stock ticks are discrete. The temporally closest stock tick T_{t_0} is selected to represent t_0. T_{t_0} is not guaranteed to be a stock tick *including* t_0, since such stock tick might not exist. The stock exchange might not be open at the specific time, or no trade was performed in the given instrument during that time frame.

Stock ticks are seen as coherent time frames of equal size (minute length), even though two successive stock ticks might be from different days or even weeks. An impressionistic description is the last stock tick of the Friday closing call and the first stock tick of Monday open call. With T_{t_0} as pivot, T_{t_0-1} is the preceding stock tick, even though it might not be temporally coherent.

Intuitively, coherent with the financial theory, short-term patterns in the historical pricing are more likely to have a short-term influence in the future pricing. The nature of short-term patterns implies the need for higher resolution to detect and distinguish the patterns. This, in turn, implies the need for higher sampling frequency. The opposite applies for long-term patterns; the influence is long-term and a lower sampling frequency is required. To give the different time horizons the same weight in the end, and at the same time reduce dimensionality, non-equal-distance sampling for T_{t_i} is performed. Every tenth sample, the sampling offset was defined to grow exponentially.

$$t_i = t_{i\mp 1} \pm \lceil 3^{\lfloor \frac{|i|-1}{10} \rfloor} \rceil \tag{8.1}$$

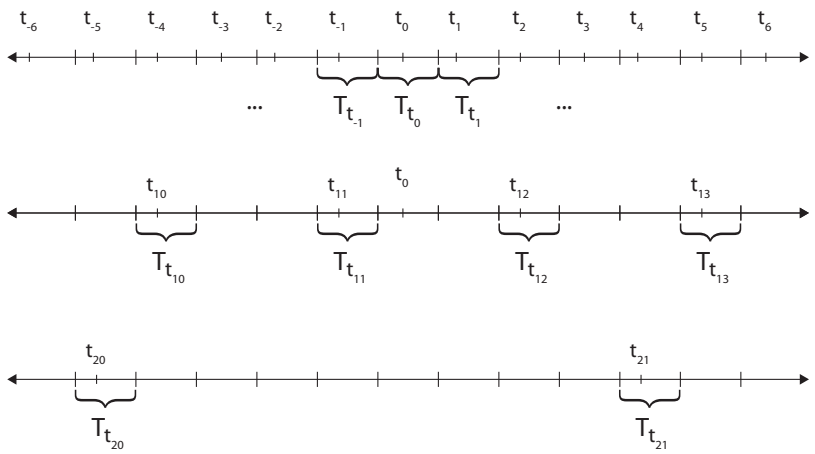


Figure 8.1: Sample distance on a timeline.

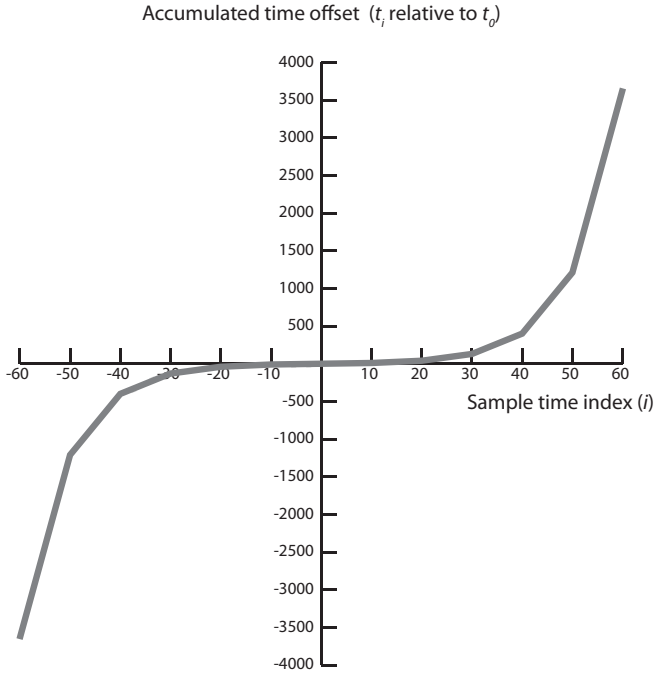


Figure 8.2: Accumulated sample distance.

Chapter 9

Price Pattern Analysis Module

The idea behind the price pattern analysis module is to find the relation between historical and future pricing of an instrument on the stock market.

System training

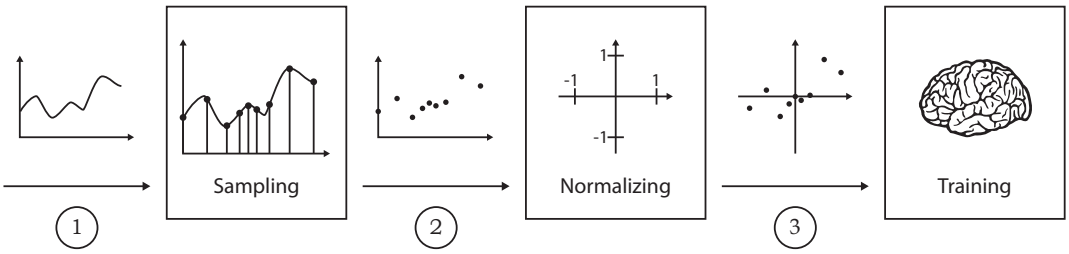


Figure 9.1: Training pre-processing flow.

(1) Retrieve stock data for every instrument

(2) Non-equal-distance sample of the stock data (higher resolution around t_0) from the historical sequence $\{t_{-n}...t_0\}$ and the outcome sequence $\{t_0...t_m\}$

(3) Normalized sample data (with t_0 in the origin)

The machine learning model, the core of the analysis module, is initially trained with a large set of stock price sequences. When sufficient training has been performed, the system runs in operational mode with real-time data.

Both the HMM and the wkNN implementation undergoes automatic supervised training. Together with incremental learning, the system fulfills the demands of being autonomous. The sampling is the only feature extraction; each sampled sequence constitutes a feature vector.

Forecasting

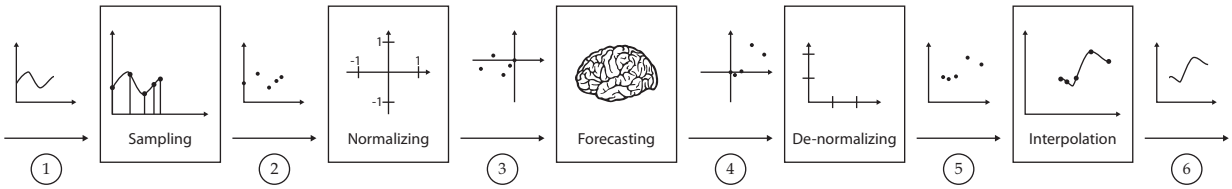


Figure 9.2: Forecasting pre- and post-processing flow.

1. Retrieve stock data for the requested instrument
2. Non-equal-distance sample of the stock data
3. Normalized sample data
4. Forecast (either using the wkNN or the HMM model)
5. De-normalized forecast (absolute dates and prices instead of relative)
6. Continuous forecast, based on interpolation of the forecast points

9.1 wkNN implementation

The use of wkNN is unconventional within this area, since it is normally used for classification. Also, the model never learns the abstract projection between input and output. Instead, being non-parametric, it keeps the full dataset of observations in memory, finding instances that are similar to a prototype instance upon request. Replacing the discrete class labels with continuous vectors from sampled observations, it is adapted to become autonomous in this application.

9.1.1 Training

To represent a stock price time-series in the wkNN model, its sampled sequence is normalized and divided at t_0 into two parts; *history* and *outcome*. The historical sub-sequence becomes the n-dimensional feature vector, which is placed in the feature space simply by adding it to the collection of existing feature vectors. Instead of binding each feature vector to a discrete label, it is bound to its output sub-sequence. Hence, each sampled sequence becomes an instance $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ in the wkNN model: a vector pair made up of the feature vector $\mathbf{x}$ and the sampled outcome $\mathbf{y}$.

$$\mathbf{z} = (\mathbf{x}, \mathbf{y}) = ([T'_{t_{-n}}, ..., T'_{t_{-2}}, T'_{t_{-1}}], [T'_{t_1}, T'_{t_2}, ..., T'_{t_m}]) \tag{9.1}$$

The instances can be interpreted to mean that n sampled points in history determine, or at least tell something about, m sampled points in the future. Or rather, m points in the future is the outcome of n points in history.

The observant reader might notice that T'_{t_0} is not included. Since the sampled sequence is normalized based on the value of T_{t_0}, it would be the same for all instances, thus not contributing to differentiate them.

The trained wkNN feature space instance $\mathbf{Z}$ consists of a collection of N entities.

$$\begin{aligned} \mathbf{Z} &= \{\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_{N-1}\} \\ &= \{(\mathbf{x}_0, \mathbf{y}_0), (\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_{N-1}, \mathbf{y}_{N-1})\} \end{aligned} \tag{9.2}$$

9.1.2 Forecasting

When making a forecast, only the historical sub-sequence has yet been observed. It is used as a prototype feature vector, denoted $\mathbf{x}_p$. Based on the Euclidean distance in n-space (Equation 9.3), the kNN function (Equation 9.4) finds the k feature vectors closest to $\mathbf{x}_p$.

$$d(\mathbf{x}_a, \mathbf{x}_b) = \sqrt{\sum_{i=0}^{n-1} (\mathbf{x}_a^i - \mathbf{x}_b^i)^2} \quad \text{where} \quad \mathbf{x}^i = T'_{t_{-(n-i)}} \tag{9.3}$$

$$k\text{NN}(\mathbf{x}_p) = \arg\min_i^k \{d(\mathbf{x}_p, \mathbf{x}_i)\} \tag{9.4}$$

The outcome of the wkNN function (Equation 9.5) is the forecast, which simply is the normalized weighted sum, based on additive inverse, of the outcomes (labels) of the closest instances.

$$\mathrm{w}k\mathrm{NN}(\mathbf{x}_p) = \frac{1}{w_{\mathrm{total}}(\mathbf{x}_p)} \cdot \sum_{i \in k\mathrm{NN}(\mathbf{x}_p)} w(\mathbf{x}_p, \mathbf{x}_i) \cdot \mathbf{y}_i \tag{9.5}$$

where

$$w(\mathbf{x}_p, \mathbf{x}_i) = d_{\mathrm{total}}(\mathbf{x}_p) - d(\mathbf{x}_p, \mathbf{x}_i)$$

$$w_{\mathrm{total}}(\mathbf{x}_p) = \sum_{i \in k\mathrm{NN}(\mathbf{x}_p)} w(\mathbf{x}_p, \mathbf{x}_i) = d_{\mathrm{total}}(\mathbf{x}_p) \cdot (k-1)$$

where

$$d_{\mathrm{total}}(\mathbf{x}_p) = \sum_{i \in k\mathrm{NN}(\mathbf{x}_p)} d(\mathbf{x}_p, \mathbf{x}_i)$$

9.2 HMM implementation

The core of the HMM implementation is an ergodic Markov chain with uniform smoothing applied. Initially, the transition matrix is randomized. No transition probability becomes zero; a minimum probability threshold is set to a thousandth of the uniform probability (before normalization). This constitutes the smoothing needed to allow never before seen state transitions to be plausible.

The inner states represent the paths of how each stock price in a sequence is reached. For example, the 1 in the sequences $[0, 1, 2]$ and $[2, 1, 0]$ should be derived from different inner states both representing 1, but with different trajectories going through that emission value. The amount of unique paths going through a specific emission value is the *path multiplier* of that value. Since not all unique paths can be covered (exponential growth of combinations), the path multiplier is rather the desired amount of generalized paths to be represented for a specific value.

Due to the discrete nature of a state model, the values must be quantized. A value span (e.g. -10 % to +10 %) and resolution must be defined. Emissions closer to the ends of this span are less likely to happen, hence less likely to occur in several paths, implying the need for a lower path multiplier. Emissions outside this span (which should be rare) are covered by the variance in the emission distribution of the states at the ends of the span. The path multiplier is set to have a Gaussian distribution with zero mean, since the outcome is assumed to have this distribution. The emission distributions are Gaussian with mean values reflecting the quantized values, repeated according to the path multiplier.

The initial probability vector reflects the proportional probability among the states to give zero output. The reason for this is that the sequences are normalized based on the first value (T_{t_n}), making the first value zero for all sequences.

9.2.1 Path multiplier

Let us assume emission values 0, 1 and 2. A zig-zag series like $[0, 1, 2, 1, 0, 1, 2, 1, ...]$ will produce twice as many ones than zeros or deuces. A zero will follow half of those ones, while the other half will be followed by a deuce. A one always follows a zero or deuce. The path multipliers $[1, 2, 1]$ for emission values $[0, 1, 2]$ respectively will enable capturing the different paths reaching the seen emission values.

$$q = \begin{pmatrix} 0.90 \\ 0.03 \\ 0.07 \\ 0.00 \end{pmatrix} \quad A = \begin{pmatrix} 0.2 & 0.4 & 0.3 & 0.1 \\ 0.2 & 0.3 & 0.2 & 0.3 \\ 0.3 & 0.2 & 0.3 & 0.2 \\ 0.1 & 0.4 & 0.2 & 0.3 \end{pmatrix} \quad B = \begin{pmatrix} N(\mu = 0, \sigma = 0.3) \\ N(\mu = 1, \sigma = 0.4) \\ N(\mu = 1, \sigma = 0.5) \\ N(\mu = 2, \sigma = 0.2) \end{pmatrix} \tag{9.6}$$

In a more realistic case the sequence contains noise and not being strictly zig-zag, resulting in the HMM $\lambda\{q, A, B\}$ according to Equation 9.6. Starting in state 0, which is most likely to output 0, you are a bit more likely to go to state 1 (outputting ~ 1) than state 2 (also outputting ~ 1). If you go to state 1, you are more likely

to continue in the same trajectory, continuing to state 3 (outputting ~ 2) than bouncing back to state 0. If you are in state 3, you are more likely to go to state 1, making it more likely to continue downwards to state 0.

In practice, with much higher multiplier, the different states giving the same output work as hubs for generalization of the combinatorial patters in the sequence resulting in its emission value.

9.2.2 Training

The HMM training is almost conventional: $n + m + 1$ long sampled sequences like $[T'_{t_{-n}}, ..., T'_{t_{-1}}, T'_{t_0}, T'_{t_1}, ..., T'_{t_m}]$ are fed to the Baum-Welch algorithm. The only difference is that the output vector is kept intact according to the defined value span and corresponding path multipliers.

9.2.3 Forecasting

Using the Forward-Backward algorithm, the state probabilities at t_0 are calculated.

$$P(S_{t_0} = i|\theta, \lambda) \tag{9.7}$$

Meaning, the conditional probability that you are in state i in model λ at time t_0 given the observed sequence of normalized stock tick samples $\theta = [T'_{t_{-n}}, ..., T'_{t_0}]$. This is the base from which the forecasting is performed.

1. In descending order, choose the states in t_0 with the highest conditional probabilities until the total probability is greater than a threshold (e.g. 90 %).
2. For each of these chosen states in t_0, choose the most probable succeeding states in t_1, until its total transition probability is greater than the threshold.
3. For the chosen states in t_1, calculate the expected value of the Gaussian mixture model (Equation 9.8) made up of their emission distributions, having their state probabilities as weights. This is the forecast value for t_1.

$$\mathrm{E}_{t_1} = \mathrm{E}\Big[\sum_j B_j \cdot P(S_{t_1} = j, S_{t_0} = i|\theta, \lambda)\Big] \tag{9.8}$$

4. For subsequent time steps t_x, repeat the procedure from step 2: Collect the most probable succeeding states. For these states, calculate the expected value of the mixture model (Equation 9.9). This is the forecast value for t_{x+1}.

$$\mathrm{E}_{t_{x+1}} = \mathrm{E}\Big[\sum_y B_y \cdot P(S_{t_{x+1}} = y, ..., S_{t_2} = k, S_{t_1} = j, S_{t_0} = i|\theta, \lambda)\Big] \tag{9.9}$$

5. The forecast:

$$[\mathrm{E}_{t_1}, \mathrm{E}_{t_2}, ..., \mathrm{E}_{t_{m-1}}, \mathrm{E}_{t_m}] \tag{9.10}$$

Chapter 10

VanirAPI

In the Vanir system, the VanirAPI component has the lead role, holding it all together. It makes sure the right external data feeds are subscribed to. It aggregates the forecasts from the analysis modules. It establishes an external JSON over socket API for subscribing to the forecasts.

VanirAPI is built to be utilized by other programs or tunneled through other APIs. The communication is subscription-based: forecasts are generated and pushed to the subscribers directly when a news article arrives. With these premises, a socket interface is the natural choice, with its canonical way of integration. Having the forecasts marked up with JSON, makes it lightweight, straight-forward and standardized; ready for any type of client-side application.

10.1 Aggregator

The aggregator subcomponent (see the overview of the system in Figure 6.1) is responsible for aggregation of the two underlying forecast modules. When a news article is published, the aggregator is invoked to make a forecast for the concerned company. In turn, the aggregator calls the attached modules, making them generate their individual forecasts and return them to the aggregator. However, the two forecasts are different from each other. Both generate m long vectors, representing forecasts over m time steps, but the values have different form. The news analysis module[1] outputs a vector consisting of discrete classes, representing the projected impact of the news article. The forecast from the price pattern analysis module consists of continuous values representing actual price projections.

For each forecasted point $F_{i,t}$ (the value at time t in forecast F_i) the error $e_{i,t}$ is calculated. Each error is paired with its forecasted class (from the news analysis module), resulting in the tuple $(c, e_{i,t})$. Grouping by c and t, the average error is calculated, representing the expected error $\mathrm{E}\left[e|c,t\right]$.

The news analysis module outputs classes in groups of ten (every ten consecutive labels in a forecast are the same). Such group of ten consecutive sample steps is

[1] Described in the master's thesis *Machine learning news analysis for stock trading* (Elgh, 2012)

called a time window. The expected error is generalized to represent the error during a time window w instead of at a single time t. This is denoted as $\mathrm{E}\left[e|c,w\right]$.

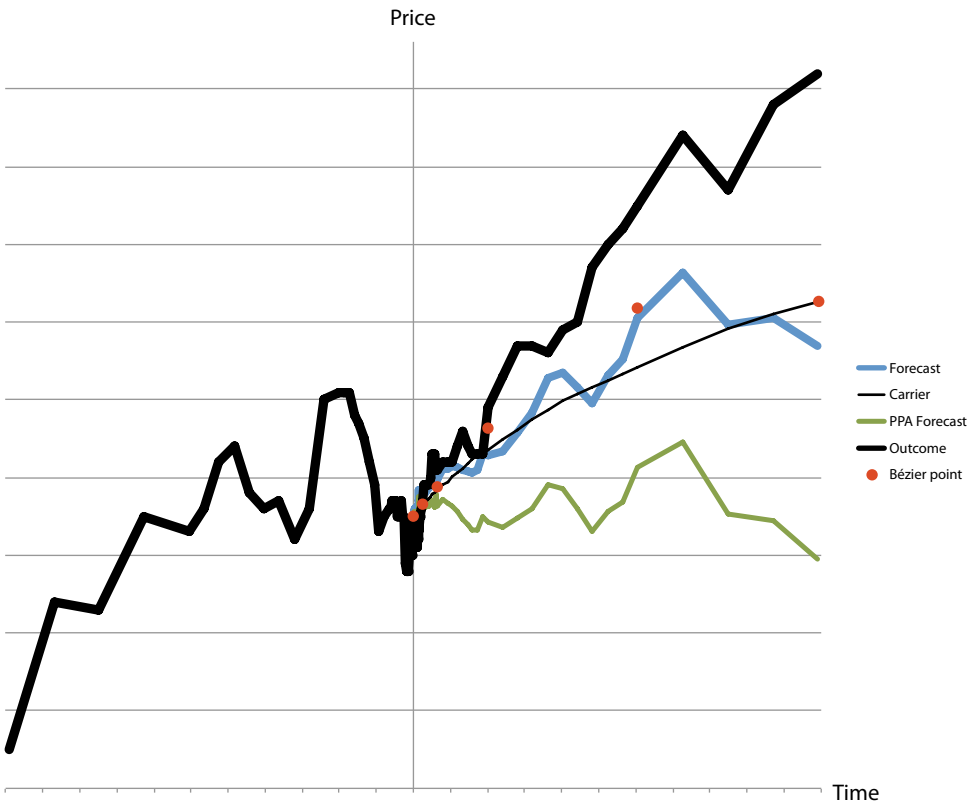


Figure 10.1: Example forecast.

When making an aggregated forecast, the sequence of classes determine the sequence of expected errors that should be corrected. For the last sample point in every time window, $\mathrm{E}\left[e|c,w\right]$ is used for error correction. The acquired sequence of expected error corrections constitute anchor points in a Bézier curve (orange dots labeled *Bézier point* in Figure 10.1). By calculating the Bézier curve, a carrier is retrieved (thin black line labeled *Carrier* in Figure 10.1). By adding the forecast (green line labeled *PPA Forecast* in Figure 10.1) with the carrier, the aggregated forecast is compiled (blue line labeled *Forecast* in Figure 10.1). Like in the example forecast (that actually is a real forecast from the system), the aggregated forecast hopefully resembles the actual outcome better than the original forecast.

10.2 System Training

Since the aggregator is depending on the attached modules of the system, it has to be trained separately. Moreover, the modules have to be trained before the

aggregator can be trained. Also, the aggregator cannot be trained using the same data as the modules.

The system allows for configuring the ratio between the module training and the aggregator training. The timespan corresponding to the training data is divided into parts fulfilling this ratio. The modules are trained using the first part of the data. Thereafter, the aggregator is trained on the second part of the data by requesting forecasts from the modules and comparing them to the actual outcome.

Chapter 11

Evaluator

To evaluate the Vanir system, k-fold cross validation was implemented in the evaluation routine. The full timespan with available time-series data is partitioned into timespans for training and testing. With a t to s training to testing ratio ($\frac{t}{s}$ times as large timespan for training as for testing), the timespan is divided into $t + k \cdot s$ partitions (t and s are integers). The t first partitions are used for training, and the succeeding s partitions are used for testing. For every fold, the whole sequence is offset by s partitions, so that the timespans for testing do not overlap. As can be seen in the 5-fold example in Figure 11.1, the training to testing ratio is 4 to 1. This means that the total time period has to be partitioned into $t + k \cdot s = 4 + 5 \cdot 1 = 9$ partitions.

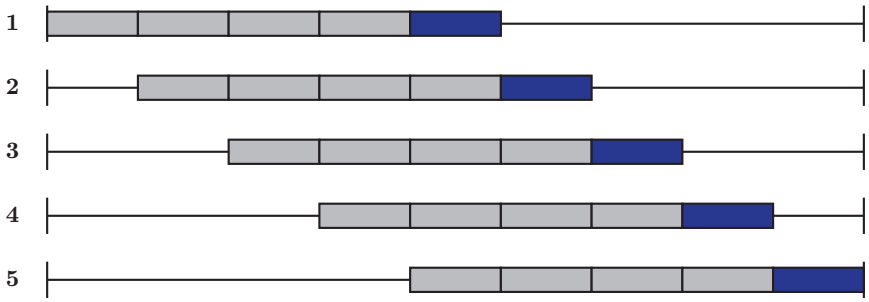


Figure 11.1: Illustration of 5-fold with 4:1 training to testing ratio. The light grey area represents the training and the dark blue area the testing partition.

During the evaluation, statistics are gathered for the two modules and the aggregator separately. To enable analysis of the discrete case, time dependent confusion matrices are assembled. Furthermore, for the continuous time-series data, expected error margin, mean values and deviations are calculated.

The gathered data can then be used for calculating key values, such as accuracy and precision, which can be evaluated over time and compared between the modules and the aggregator.

Part III

Experiments and Results

Chapter 12

Data Selection for Evaluation

For the evaluation, one-minute resolution stock price data for the whole year of 2011 for all the stocks included in the OMXS30 index were used.

Similarly, news articles published during 2011, concerning the companies included in the index, from the two sources Nyhetsbyrån Direkt and SIX News were used. Before being used, the news articles were filtered. Articles with more than two tagged companies in the meta data were removed as they became too vague. Also, all articles published off-market hours were removed since the impact cannot be isolated and bound to the actual time of publication. Suppose for instance that the greatest impact of a news article on the stock price is within 20 minutes after publication. If an article was published at 07.00 in the morning, the impact would not be seen until 09.00 (when the market opens)—if at all—and it may not appear in the same manner as if it were to be published when the market was open.

In Figure 12.1 the distribution of news articles per 15 minutes can be seen. What can be seen in the figure is that a large portion of the published news articles during a day is before the market opens. Hence, they are removed.

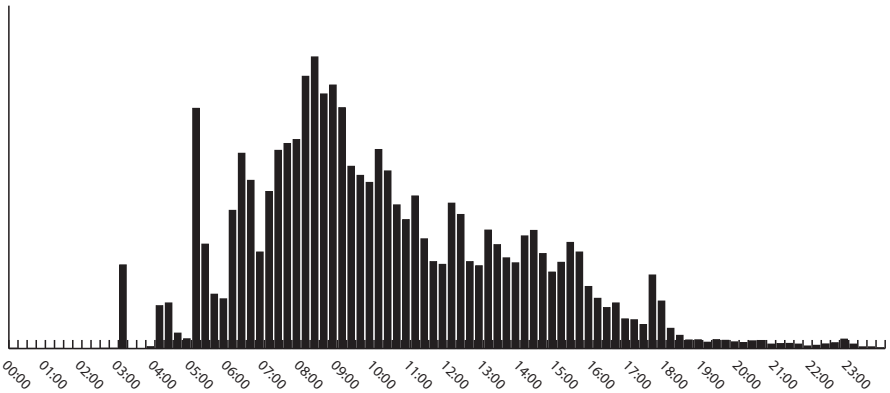


Figure 12.1: News article distribution per 15 minutes. A column represents the number of news articles that were published during that period. The actual height of the columns is not of importance since the purpose is to illustrate the distribution.

Chapter 13

Results

13.1 Price Pattern Analysis Module

13.1.1 *wk*NN

In the discrete case, every sample point is classified as UP, DOWN or ZERO by discretizing the real value. Before discretizing, the normalized value is rounded to four decimals. When translated into real prices, that roughly translates into the tick sizes on the Nasdaq OMX Stockholm stock exchange.

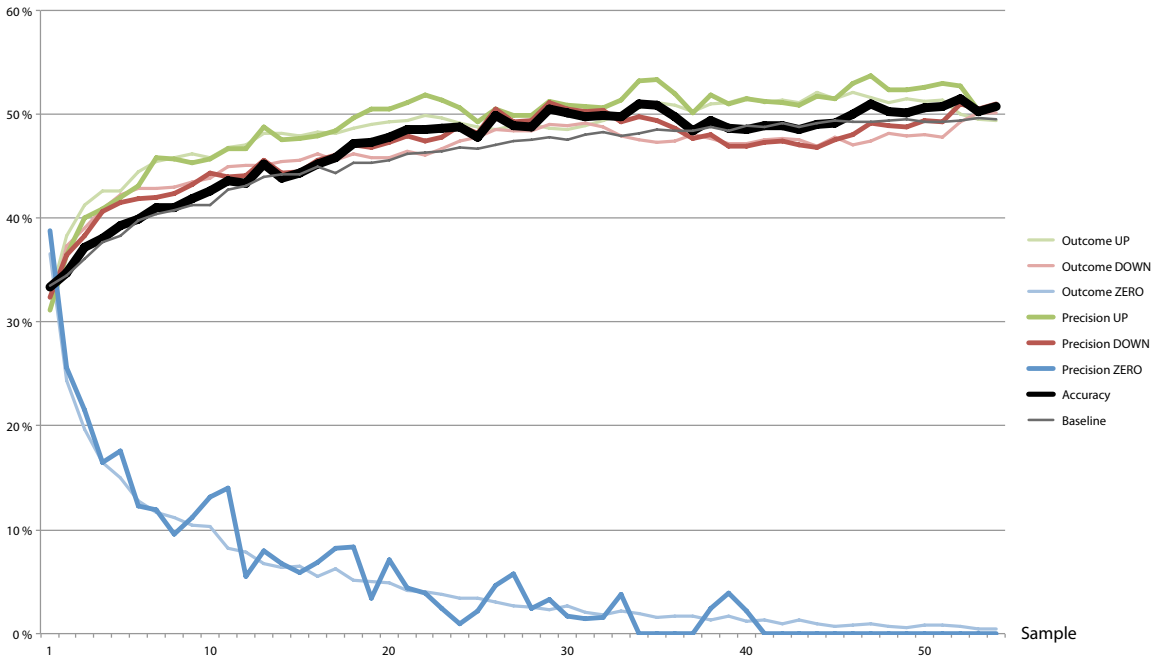


Figure 13.1: Forecast performance over time, with sample scale. k=20

		Outcome			
		UP	DOWN	ZERO	Σ
Forecast	UP	58 633	54 379	6 363	119 375
	DOWN	64 612	62 882	6 440	133 934
	ZERO	3 895	3 783	1 345	9 023
	Σ	127 140	121 044	14 148	262 332

Table 13.1: Confusion matrix. k=20

		Outcome			
		UP	DOWN	ZERO	Σ
Forecast	UP	22.35 %	20.73 %	2.43 %	45.51 %
	DOWN	24.63 %	23.97 %	2.45 %	51.06 %
	ZERO	1.48 %	1.44 %	0.51 %	3.44 %
	Σ	48.47 %	46.14 %	5.39 %	100.00 %

Table 13.2: Confusion matrix in relative numbers. k=20

		Performance	**Relative performance**
Accuracy		46.83 %	103.91 %
Precision	UP	49.12 %	101.34 %
	DOWN	46.95 %	101.75 %
	ZERO	14.91 %	276.39 %

Table 13.3: Performance. k=20

The relative performance is the performance in relation to the expected value based on chance, i.e., the outcome distribution. For example, if you have a 0.50/0.50 outcome distribution and get 0.50/0.50 precision distribution, the relative performance is 100 %, respectively. If every instance would be correctly classified, the relative performance would instead be 200 %.

The 0.4847/0.4614/0.0539 outcome distribution leads to an expected value for accuracy of 45.07 %[1], based on chance. This is the denominator used for the relative performance value of the accuracy. The time dependent value of this denominator is the baseline used in Figure 13.1.

In every evaluation there is always a probability that the results may occur just by chance, even if the performance values are more or less significantly different from the output distribution. The chance is inversely proportional to performance and number of instances evaluated: the bigger difference and the more evaluations performed, the less likely it is to have happened by chance. The confidence value (used in Table 13.4 and 13.5) indicates the probability that the performance value *did not* occur by chance.

[1]Each value is weighted by its own probability, yielding $0.4847^2 + 0.4614^2 + 0.0539^2 = 0.4507$

$$\mathrm{P}(\text{hits} = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

$$\mathrm{P}(\text{hits} \leq k) = \sum_{i=0}^{k} \mathrm{P}(\text{hits} = i) = \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i} = \mathit{binom\_cdf}(k, n, p) \quad (13.1)$$

$$\mathrm{P}(\text{hits} > k) = 1 - \mathrm{P}(\text{hits} \leq k) = 1 - \mathit{binom\_cdf}(k, n, p)$$

The probability that one gets more than k hits by chance is given by Equation 13.1. The binomial cumulative distribution function (CDF), called *binom_cdf* in the equation, returns the probability of getting *at most* k correctly classified instances out of n trials with the probability p of success in each trial.

		Performance	**Confidence**
Accuracy		50.52 %	99.99 %
Precision	UP	51.88 %	99.99 %
	DOWN	49.32 %	99.99 %

Table 13.4: Performance of non-zero forecasts. k=20

When ZERO is ignored for both forecast and outcome (due to the ambiguity it would imply by keeping them, discussed by Hellström (1998), cited in Section 4.1.3), the output distribution becomes $p = 0.5123$ for UP and $p = 0.4877$ for DOWN. Even though the precision for UP and DOWN, and the total accuracy, is just barely over their expected values respectively (Table 13.4), the confidence is high ($\alpha = 0.0001$). This is due to the extensive amount of evaluations performed (239 236 non-zero forecast points).

The hit rate is highly volatile when plotted with the samples on the x-axis (Figure 13.2). Note that this representation is non-linear in time. The actual time intervals between the samples are not equal, according to the sample distance (described in Chapter 8).

Between sample 35 and 50 (∼4.5–20 market hours ahead, i.e., mostly the next day and the day after that), the output distribution (and precision) is significantly different between UP and DOWN. At the same time the average performance is close to random, slightly predominated towards underperforming. This is also true for all of k=5, k=10, k=20 and k=30 (see Appendix A).

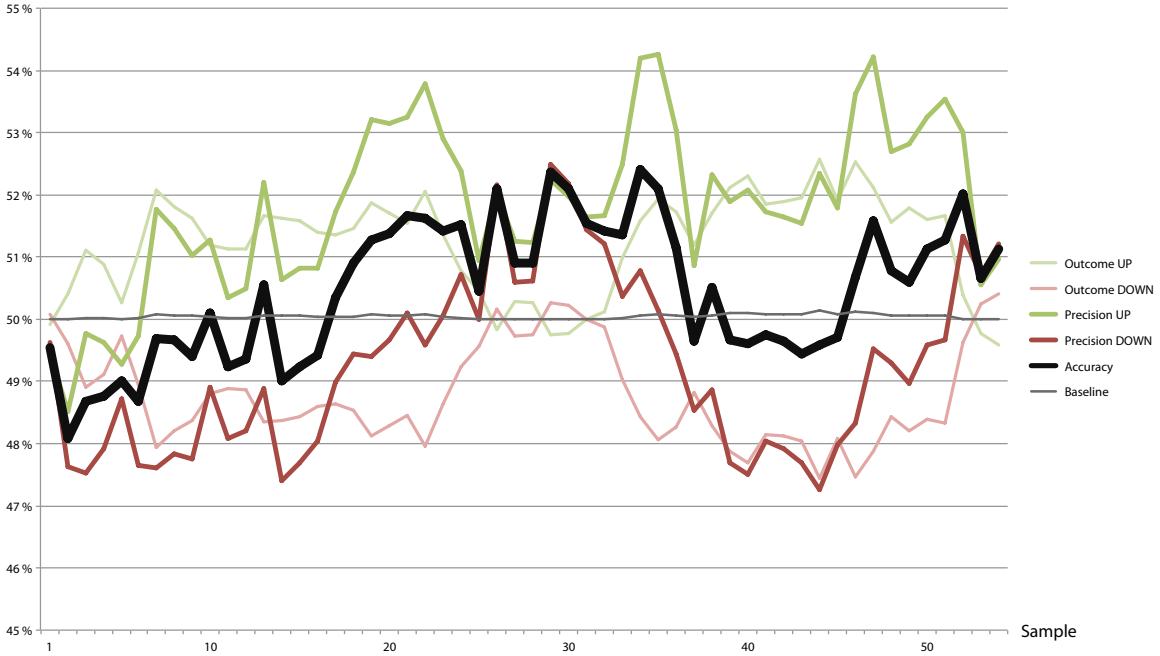


Figure 13.2: Hit rate over time, with sample scale. k=20

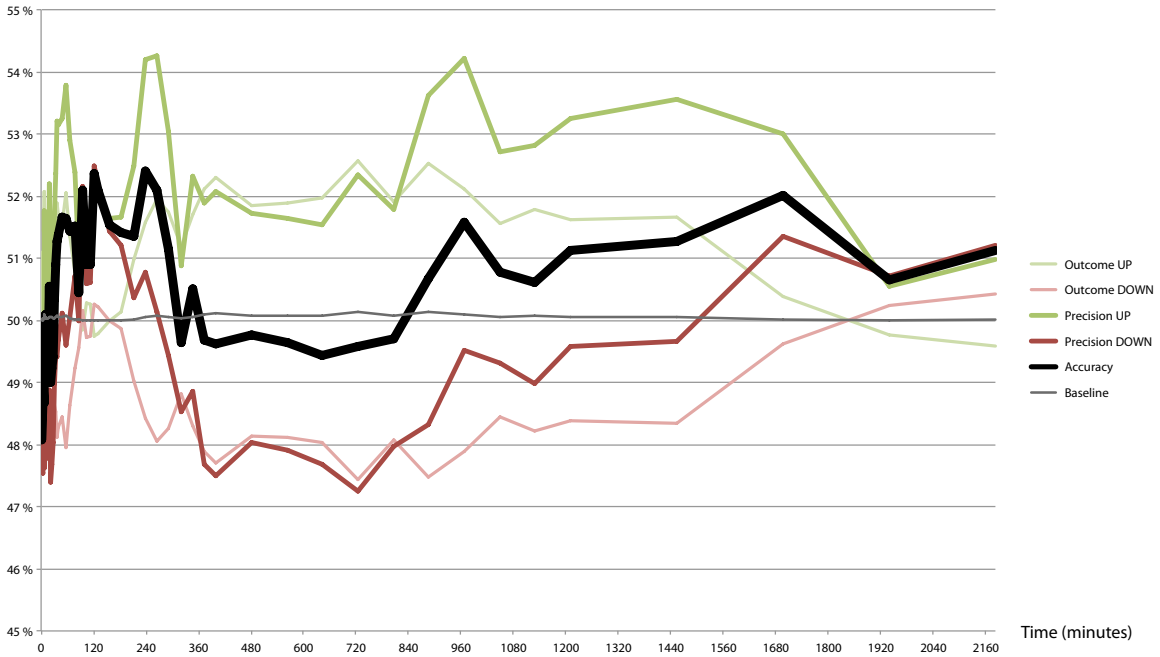


Figure 13.3: Hit rate over time, with time scale in minutes. k=20

The choice of scale on the x-axis reveals different parts of the performance. By using the sample scale (Figure 13.2) the visual resolution is kept higher in the short perspective. When plotting the performance in relation to actual time (Figure 13.3) the long-term performance gets more visually apparent.

		Performance	Confidence
Accuracy		51.09 %	99.99 %
Precision	UP	52.80 %	99.99 %
	DOWN	49.89 %	99.98 %

Table 13.5: Performance from sample 46 and forward (∼14–36 market hours ahead)

From approximately 840 minutes (14 market hours ahead) and forward (∼36 market hours ahead, effectively ∼4.5 days), the forecast is strictly over-performing. At the same time, the actual outcome (the baseline) converges to a 0.50/0.50 distribution.

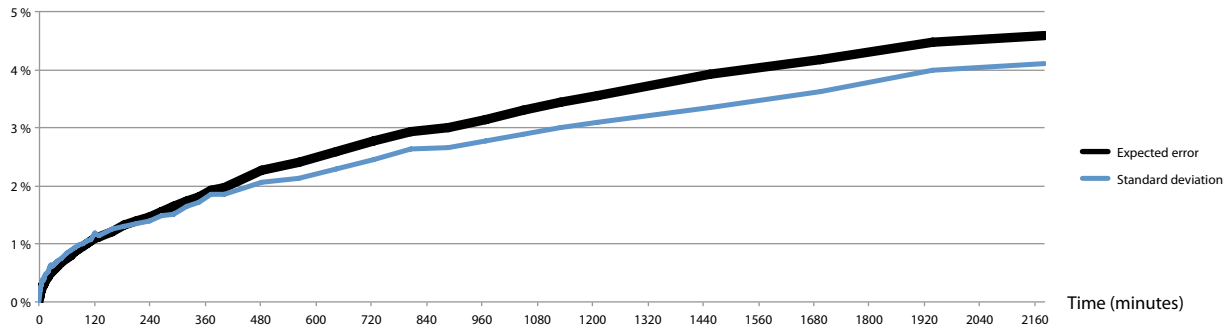


Figure 13.4: Continuous forecast error, with time scale in minutes.

The continuous case is more straight-forward: simply calculate the expected error and variance between each forecast and its actual outcome. The expected error after approximately 36 market hours (the last sample, effectively ∼4.5 days ahead) is 4.6 percentage points with a standard deviation of 4.1 percentage points. The expected error and its deviation over time can be seen Figure 13.4.

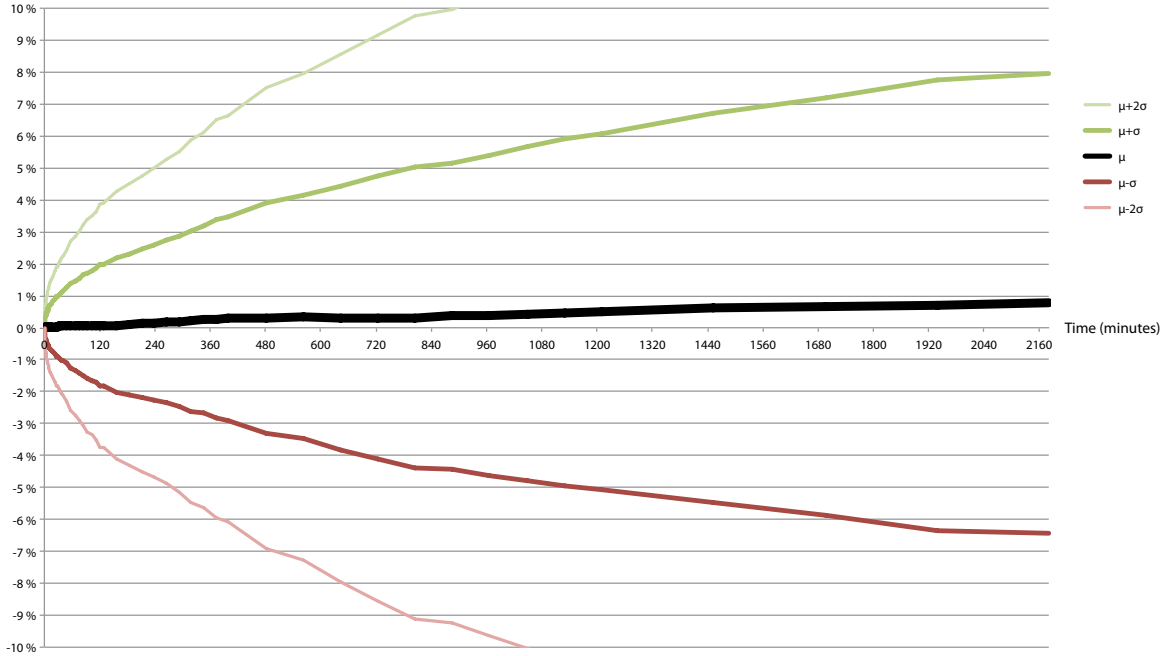


Figure 13.5: Error impact (assuming Gaussian distribution).

Not being correct is not necessarily as bad as it seems: if a forecast saying "up 1 %" will make one take a long position and the actual outcome is 2 % up, the forecast is off by one percentage point but one will make an even better trade. The same thing applies for negative forecasts, when taking a short position. In both cases the *impact* is positive: one would make more money than expected. If the outcome has lower amplitude or wrong direction, the impact would be negative (i.e. forecasting 2 % up and the outcome is just 1 % up, or forecasting up when the outcome is down).

The error impact $I_{i,t}$ when forecasting F_i and having the outcome O_i is given by Equation 13.2 (for any given time t). As can be seen in Figure 13.5, the impact would yield positive returns due to underestimations of amplitude in the forecasts.

$$I_{i,t} = sign(O_{i,t} \cdot F_{i,t}) \cdot (|O_{i,t}| - |F_{i,t}|) \tag{13.2}$$

The forecast distribution describes how the forecasts are biased due to the trend in the training data. Figure 13.6 visualizes a slight bias towards negative forecasts, ending up in an expected value of -0.61 %. This bias is obviously the same for each of k=5, k=10, k=20 and k=30 (see Appendix A), since they all represent the same timespan of training, with the same trend.

During testing, when the forecasts show a negative bias, the expected value for the outcome is slightly up (∼0.20 %) as can be seen in Figure 13.7. Hence, the expected error impact at the last sample is more than 0.8 percentage points due to the different trends in the training and testing data (confirmed by Figure 13.5).

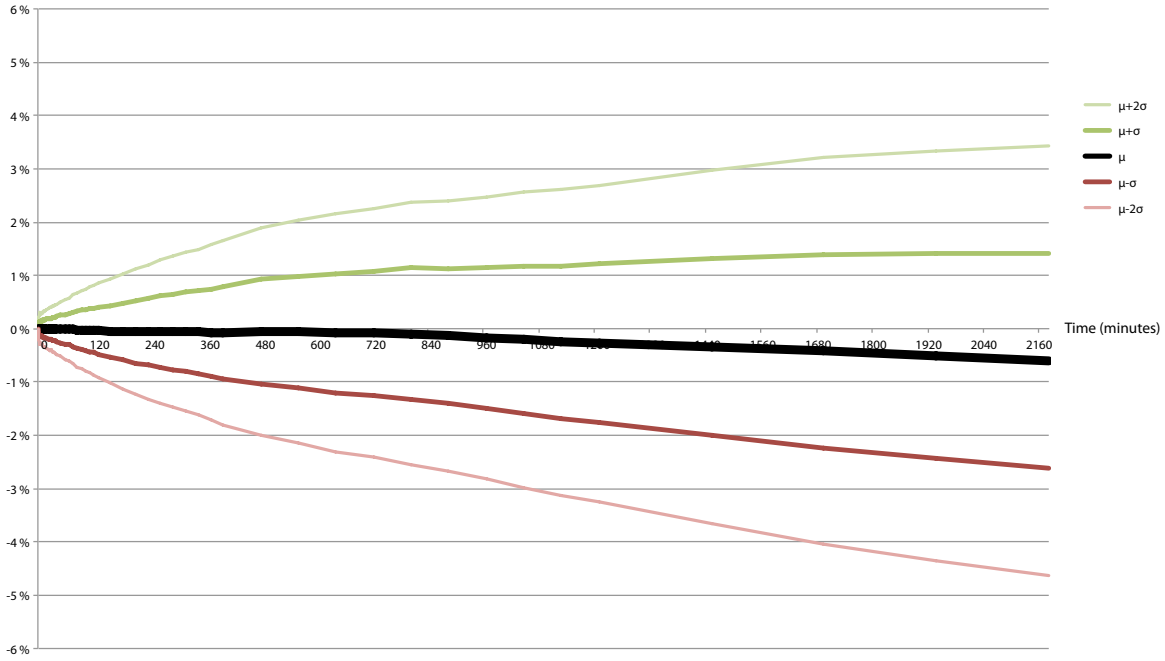


Figure 13.6: Forecast distribution over time (assuming Gaussian distribution), with time scale in minutes. k=20

The lower variance of the forecasts, caused by the fact that every forecast is a generalization of several historical outcomes, makes it impossible to forecast the highly volatile outcomes. The higher k-value used in the wkNN model, the lower variance in the forecast (see Appendix A).

The last evaluation is whether the forecast correlates with the outcome. If high correlation could be shown, the error and hit rate would not matter; it would just demand application of a different trading strategy. The Pearson's correlation varies a lot: some forecasts have over 90 % correlation, but several also show negative correlation. Altogether, the average Pearson's correlation is just 1.5 %, making it impracticable.

Visual inspections show that forecasts with low correlation values sometimes are unfairly judged as bad. Intricate patterns are often depicted correctly but in an incorrect trend or with a time offset, resulting in low or negative correlation.

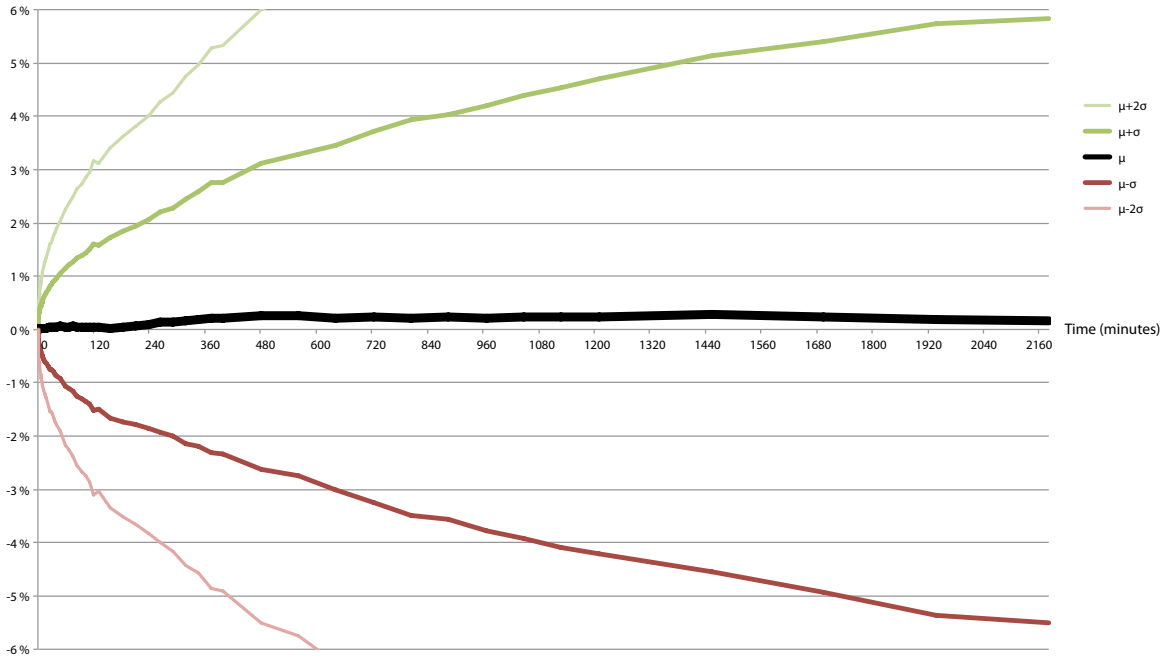
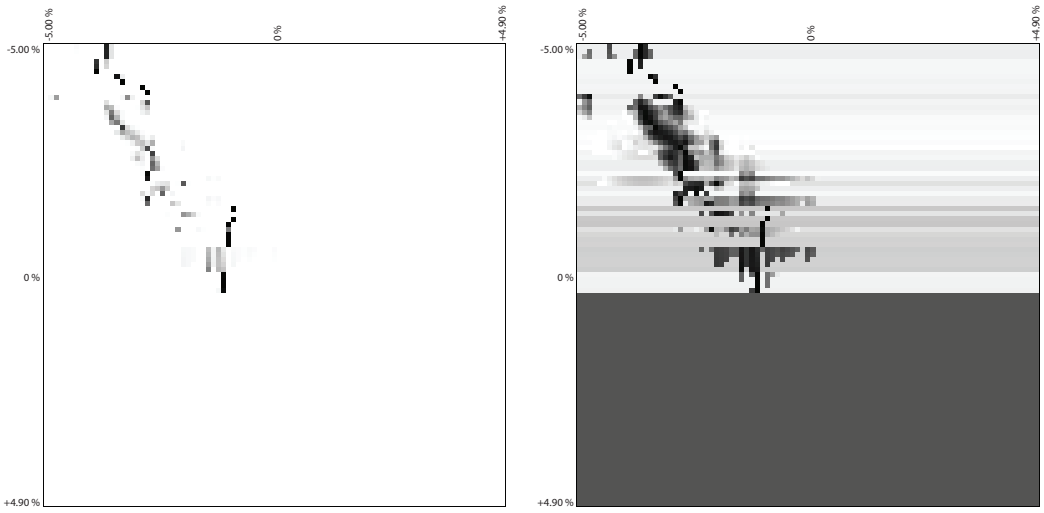


Figure 13.7: Outcome distribution over time (assuming Gaussian distribution), with time scale in minutes.

13.1.2 HMM

A reasonable resolution—when quantizing the sequences to work in the discrete environment of a Markov model—is 0.1 percentage points. For the time horizon of the forecasts, a span between -5 % and +5 % should be sufficient. That results in 101 discrete output values. To calculate the distribution for the path multiplier, $N(\mu = 0, \sigma = 3)$ would make the span [-5, 5] cover 90 % of the outcome values. To get a path multiplier of about 40 for the 0 % output, around 1 000 states would be required in the HMM. Tries were made, but they were too slow[2]. Even when decreasing the amount of states to 100, every sequence took 1.8 seconds (making one day take more than 20 minutes for all instruments). It is important to remember that the HMM has the time complexity of $\mathcal{O}(N^2T)$, making this 100 times faster than the original attempt.



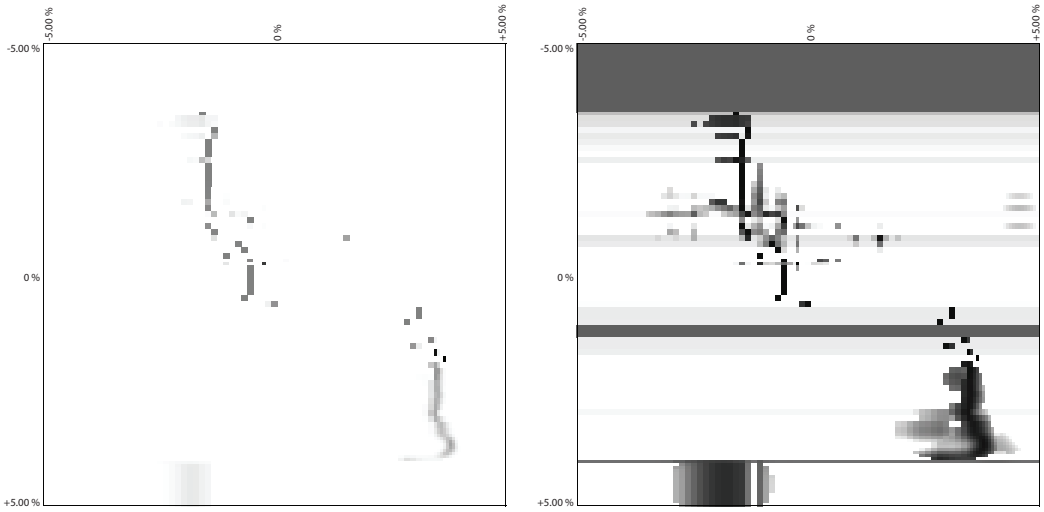
(a) Linear scale

(b) $\log_{10}$ scale

Figure 13.8: Transition matrix for a trained HMM (with 91 states) as a heat map. White represents 0 and black represents 1. December 1st to December 20th 2011.

With 100 states the path multiplier becomes 1, and the original Markov property reigns: the historical path is disregarded when looking ahead. This disconcerts the whole idea about customizing the HMM for forecasting. All forecasts are the same: during the first few sample steps, the forecast represents a time-compressed version of the overall trend, where it gets stuck in its final value.

[2]The evaluations were run on a MacBook Pro 8.2 with a 2.2 GHz Intel Core i7 (quad core) processor and 8 GB 1333 MHz DDR3 RAM, running Mac OS X Snow Leopard (10.6.8).



(a) Linear scale (b) $\log_{10}$ scale

Figure 13.9: Transition matrix for a trained HMM (with 154 states) as a heat map. White represents 0 and black represents 1. December 10^{th} to December 23^{rd} 2011.

When doubling the number of desired states, training becomes four times slower; each sequence taking about 8 seconds. The centermost 70 % of the emission values get 2 as path multiplier, which is of course far to low (compare to the zig-zag pattern in Section 9.2.1). Still, making a 1-fold evaluation over December 10^{th} to December 31^{st} 2011 takes one half hour. The forecasting abilities are unchanged.

No further evaluation was performed since no concrete potential could be shown and it was too slow to be evaluated properly either way.

13.2 Vanir

13.2.1 Aggregator

Table 13.6 and Figure 13.10 show the expected errors, $\mathrm{E}\left[e|c,w\right]$, which were used to correct the forecasts from the price pattern analysis module (see Section 10.1).

The table entries equal to zero were not calculated. This is because there were no occurrences of the class c (from the news analysis) during the time window w during training of the aggregator. The further away from t_0, the fewer classes occur.

	w_1	w_2	w_3	w_4	w_5	w_6
UP_GOOD	-0.0002	0	-0.0002	0.0013	0.0060	0.0116
UP_NEUTRAL	0	0	0	0	0	0
UP_BAD	0.0005	0.0006	0.0019	0.0021	0.0082	0
ZERO_GOOD	-0.0002	-0.0041	0.0019	-0.0036	0	0
ZERO_NEUTRAL	-0.0001	0	0	0	0	0
ZERO_BAD	0	-0.0015	0.0003	0	0	0
DOWN_GOOD	-0.0002	0.0003	0.0005	0.0127	0.0032	0
DOWN_NEUTRAL	0	0	0	0	0	0
DOWN_BAD	0.0003	0.0006	0.0016	0.0016	0.0027	0.0041

Table 13.6: The expected error, $\mathrm{E}\left[e|c,w\right]$.

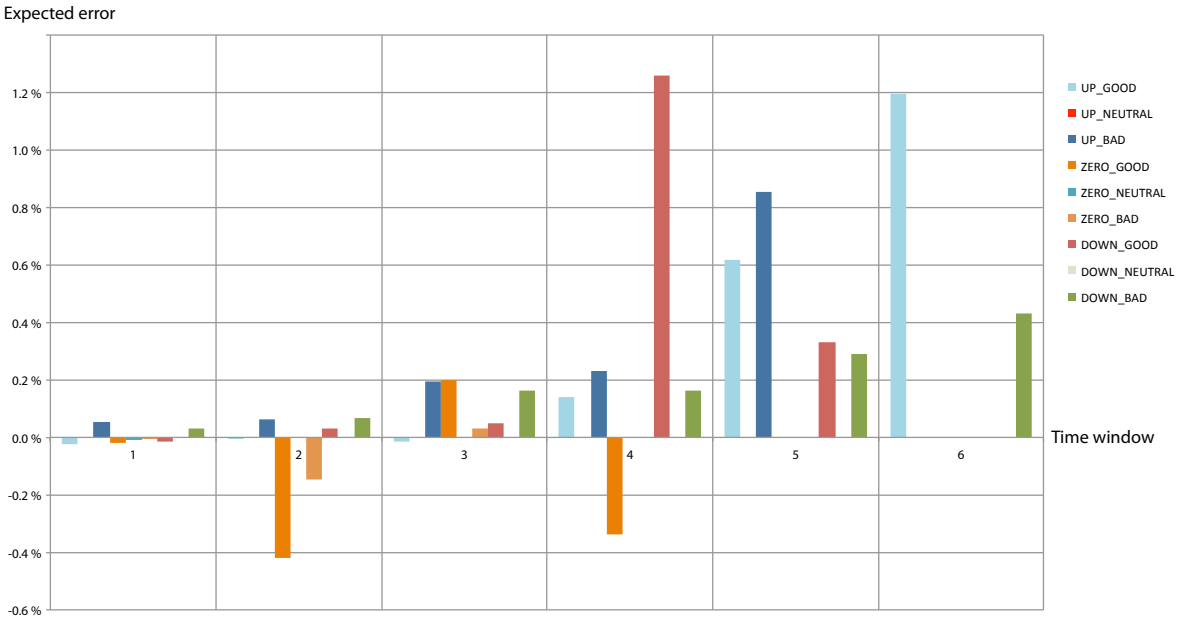


Figure 13.10: The aggregated expected error per class, per time window.

		Outcome			
		UP	DOWN	ZERO	Σ
Forecast	UP	63 165	60 540	6 696	130 401
	DOWN	60 158	56 868	6 103	123 129
	ZERO	3 817	3 636	1 349	8 802
	Σ	127 140	121 044	14 148	262 332

Table 13.7: Confusion matrix.

		Outcome			
		UP	DOWN	ZERO	Σ
Forecast	UP	+4 532	+6 161	+333	+11 026
	DOWN	-4 454	-6 014	-337	-10 805
	ZERO	-78	-147	+4	-221

Table 13.8: Net change compared to the wkNN (k=20) confusion matrix.

		Outcome			
		UP	DOWN	ZERO	Σ
Forecast	UP	24.08 %	23.08 %	2.55 %	49.71 %
	DOWN	22.93 %	21.68 %	2.33 %	46.94 %
	ZERO	1.45 %	1.39 %	0.51 %	3.35 %
	Σ	48.46 %	46.15 %	5.39 %	100.00 %

Table 13.9: Confusion matrix in relative numbers.

		Outcome			
		UP	DOWN	ZERO	Σ
Forecast	UP	+1.73 %	+2.35 %	+0.13 %	+4.20 %
	DOWN	-1.70 %	-2.29 %	-0.13 %	-4.12 %
	ZERO	-0.03 %	-0.06 %	0.00 %	-0.08 %

Table 13.10: Net change in percentage points compared to the wkNN (k=20) confusion matrix in relative numbers.

		Outcome			
		UP	DOWN	ZERO	Σ
Forecast	UP	+7.73 %	+11.33 %	+5.23 %	+9.24 %
	DOWN	-6.89 %	-9.56 %	-5.23 %	-8.07 %
	ZERO	-2.00 %	-3.89 %	+0.30 %	-2.45 %

Table 13.11: Relative net change compared to the wkNN (k=20) confusion matrix.

Table 13.10 shows the net difference between the confusion matrices of relative numbers (Table 13.9 minus Table 13.2) while Table 13.11 shows the relative change of the confusion matrices of absolute numbers (Table 13.7 element-wise divided by Table 13.1, minus one).

As can be seen in these tables, the aggregator infers a bias towards forecasting UP. 9.56 % (net) of the correct DOWN forecasts were shifted into incorrectly forecasting UP. 6.89 % of the incorrect DOWN forecasts were correctly shifted to UP. The net result from this readjustment is negative.

		Performance	**Significance**
Accuracy		49.86 %	8.80 %
Precision	UP	51.06 %	12.04 %
	DOWN	48.59 %	11.27 %

Table 13.12: Performance of non-zero forecasts.

As a result of the negative effect revealed by the change in the confusion matrix, the final performance values (Table 13.12) are bad: the performance is totally random. As can be seen in Figure 13.11 and 13.12, the negative effect applies to the whole time spectra. The good parts have been heavily suppressed and the bad parts have become worse.

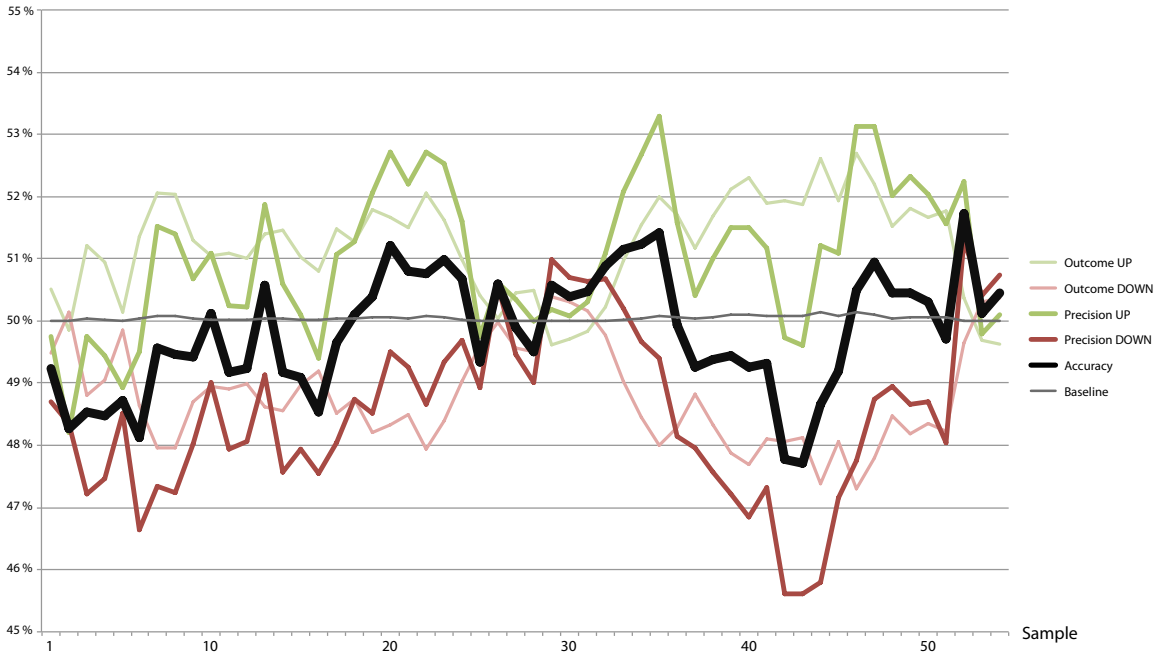


Figure 13.11: Hit rate over time, with sample step scale.

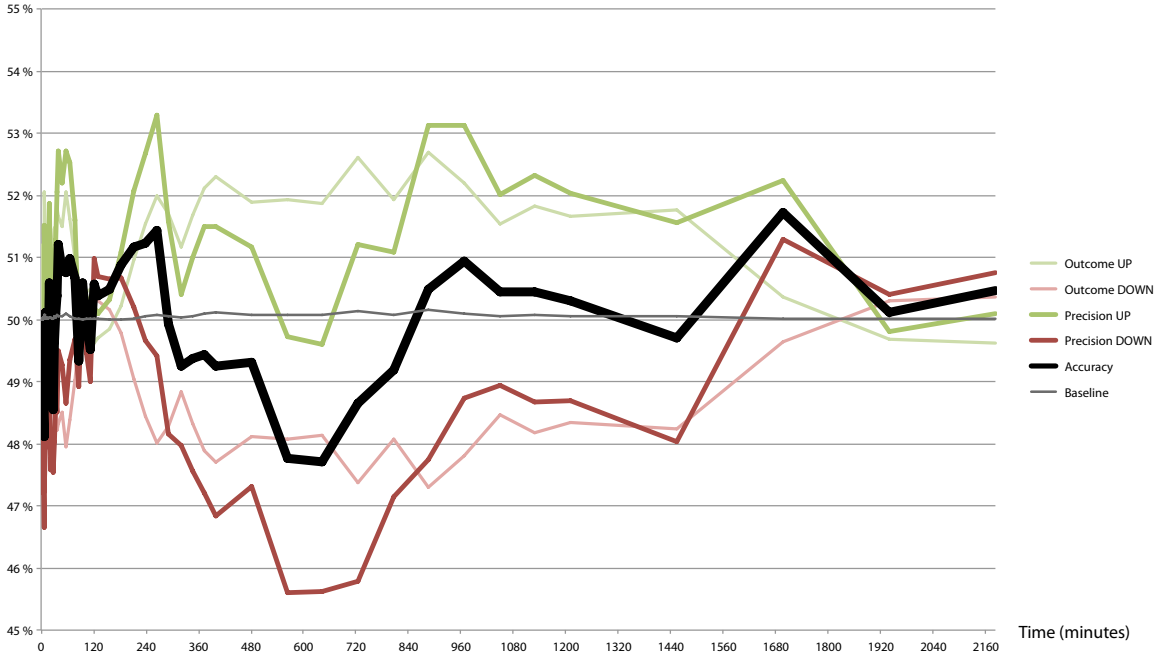


Figure 13.12: Hit rate over time, with time scale in minutes.

As can be seen in Figure 13.13, the forecast distribution has a mean value closer to zero, compared to the original forecast from the price pattern analysis module. This shows that the aggregation adjusts (slightly) for the negative bias that exists in the original forecast (Figure 13.6). This is reflected in the error impact (Figure 13.14), which shows a more neutral result. This could either mean that the aggregated forecast actually makes more correct forecasts (instead of underestimates), and/or make even worse forecasts in those cases where the impact was already negative.

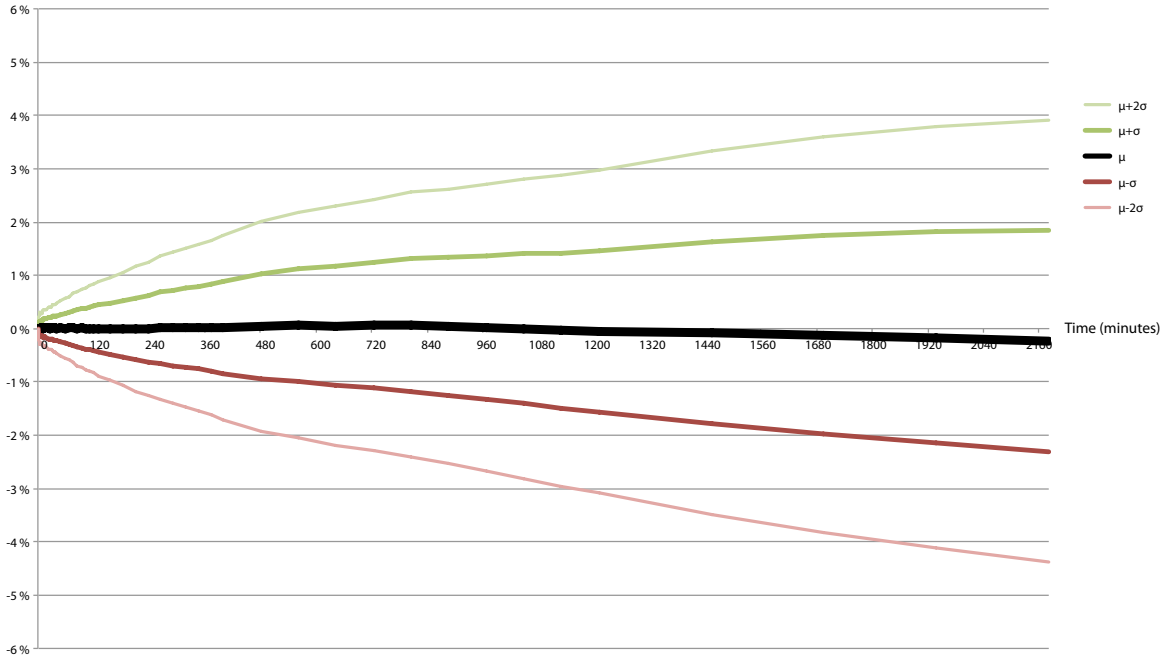


Figure 13.13: Forecast distribution over time (assuming Gaussian distribution), with time scale in minutes.

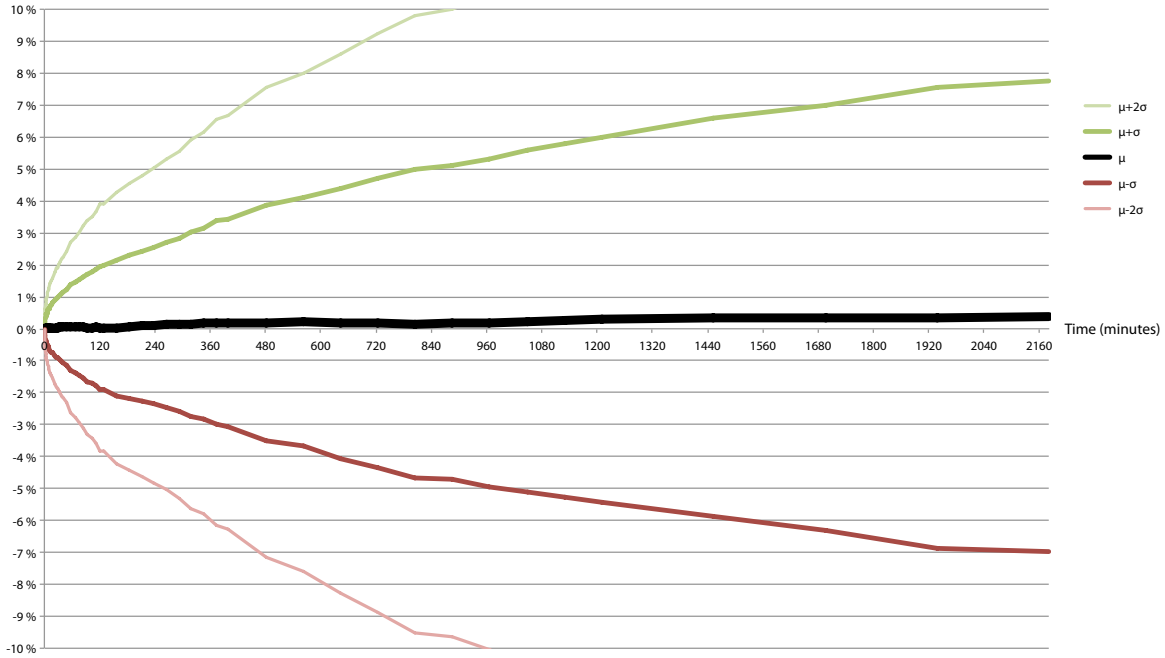


Figure 13.14: Error impact (assuming Gaussian distribution).

Chapter 14

Discussion

14.1 Price Pattern Analysis Module

The first goal was met; the discrete accuracy, the hit rate, was proven better than chance with high confidence. Without applying a trading strategy, which is outside the scope of this thesis, the results did not confirm or discard whether it could be profitable in practice.

The wkNN forecaster performs weakly in the short term: it underperforms significantly during the first half hour after the forecast. This can be seen in Figure 14.1. Also, a systematic performance drop is seen between six and fourteen hours after the forecast was made. This corresponds roughly to the day after the forecast. These underperforming timespans constitute ∼45 % of the sample points.

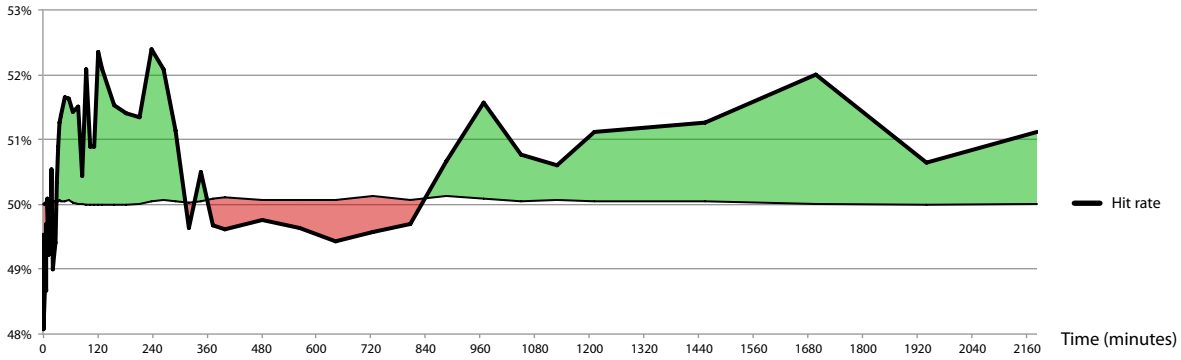


Figure 14.1: Hit rate over time. Red area means underperforming and green area means outperforming. The green area constitutes 90 % of the colored area. 75 % of the time, the forecast outperforms (i.e. has green area above the baseline).

In the short term, the price variations are small. This causes them to be suppressed in the model; more distant sample points, where the variations are larger, might outvote the short-term pattern matching. A prototype with a perfect match for the immediate half-hour, might not be even close to that instance in n-space, due to large variations in more distant sample points.

It is harder to explain why it underperforms the day after the forecast. It could be that the market overreacts to published news during the same day, to find its way back to a more rational value the day after. Since the price pattern analysis is not trained driven by news, this could be a reason for failing to include the readjustment in the patterns. A more abstract explanation could be that one to two days in the future is too far for the short-term patterns to tell anything about, yet to close for the long-term patterns to do the same.

In traditional technical analysis of securities, traded volume is fundamental (discussed in Chapter 2). This was rejected in the compromise to keep it simple and reduce dimensionality to make it lighter and faster. It is reasonable to believe that the model could perform better if volume was included, due to the findings in previous research. However, higher dimensionality would require more training data to reach a stable level. The training data was delimited to one year due to the time it took to run a 5-fold cross-validation. If the dimensionality would be doubled *and* a larger timespan would be needed to include sufficient training data, evaluation would be exhaustingly time consuming.

Increasing the k-value showed the expected behavior that the forecasts eventually get averaged out. Together with the way the neighbors are weighted, stability is reached at a certain k where the marginal distance is so far that the weight converges to zero. No evaluation of the wkNN feature space was made. This could have given answers whether the weighting function was relevant or not.

The HMM implementation was unfortunately a practical failure. Even from the beginning it was a wildcard since the original model is designed to ignore the elapsed path to a specific state. However, the basic idea of path multipliers worked well in theory with a minimal amount of emission values, thus few potential trajectories to be captured. No practical potential could be shown.

14.2 Vanir

14.2.1 Aggregator

Since the mean value of the forecast distribution for the aggregated forecast is higher than the original forecast from the price pattern analysis module, the forecasts are generally adjusted upwards. This is consistent with that the mean value of the outcome is higher than the original forecast as well. Intuitively, the results should be better since the adjustment is generally in the right direction. However, the (continuous) error is only marginally improved while the hit rate is significantly impaired (which can be seen in Figure 14.2, in comparison to Figure 14.1).

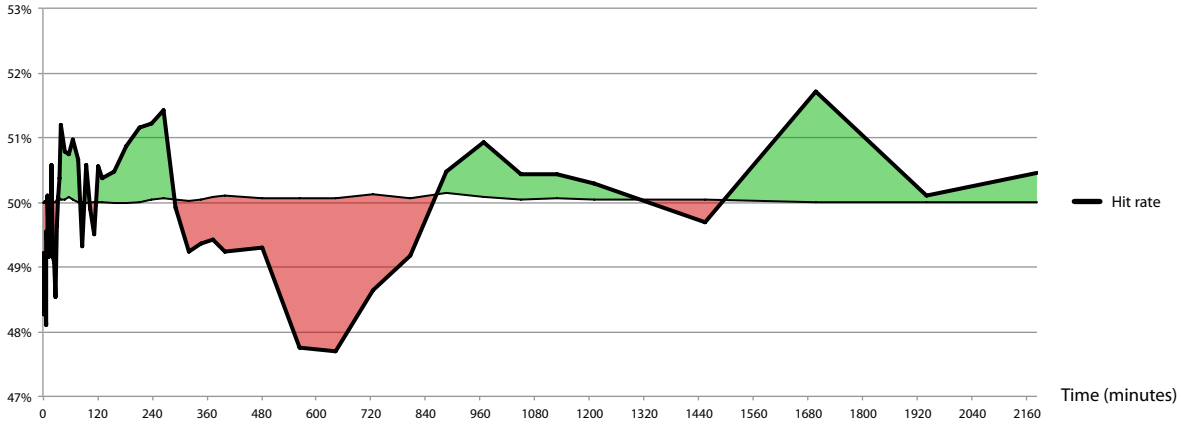


Figure 14.2: Hit rate over time. Red area means underperforming and green area means outperforming. The green area constitutes 51 % of the colored area. 60 % of the time, the forecast outperforms (i.e. has green area above it).

The results show that the forecasts in many cases were incorrectly adjusted upwards and becoming positive, when they originally were correctly negative. This indicates a heavy bias upwards, independent of the news article, which can be seen in Table 13.6 (or Figure 13.10 for better visualization). Almost all expected errors are positive (if not zero).

There is also a large number of expected errors that are equal to zero (seen in Table 13.6). This is because they were never computed since their corresponding class never occurred during the aggregator training phase. This is a direct consequence of the uneven class distribution, discussed by Elgh (2012).

14.2.2 Evaluation

There is a significant issue about how the hit rate is evaluated. The basis for the evaluation is the amount of discrete samples, instead of the continuous time. During the first six hours of a forecast 70 % of the sample points occur, which only represents 16 % of the timespan.

The non-equal distance sampling was intuitively designed to give high resolution in the short term and to weight the proximity higher compared to more distant patterns. These sample points were used for evaluation as well. Because of that, the overall (time independent) performance was heavily biased by the short-term results. To evaluate the forecasts more fairly without this bias, the continuous (interpolated) forecast should be sampled with equal distance and *then* compared to the outcome. Then the sample points would represent the time extension equally weighted. In Figure 14.1—which is just a subset and re-coloring of Figure 13.3—it is revealed that the hit rate outperforms 75 % of the time (but only 52 % of the samples). And when considering the hit rates as weights during the elapsed time, the outperform to underperform ratio is 9 to 1.

Correlation, either Pearson or Spearman, is commonly used when evaluating how the prices of different instruments or goods behave in relation to each other. This is useful when compiling a portfolio with a hedge to get a desired risk and expected yield. Using the same method to evaluate forecasts was not as successful: offset in time could result in being classified as uncorrelated or inversely correlated. The same applies to similar patterns but in the wrong major trend.

Two examples can of this can be seen in the Atlas Copco example in Figure 14.5. Forecast 1 produces a forecast with a 2.5 % plateau that should last for a day. The outcome is almost identical, but offset with a day. The correlation during these two days is -16 %. If making the scope tighter to only fit the two plateaus, the correlation is -71 %. For the whole forecast, which fails miserably after two and a half days, the correlation is -32 %. Until it fails (even by visual inspection), the correlation is 27 % (from t_0). How could that be, when four fifths of the timespan has negative correlation? Once again, the sample distances imply a bias that values the short perspective higher due to higher density of sample points: the first fifth of the timespan constitutes over 60 % of the sample points. Subjectively, the depiction of the plateau is far more impressive from a forecasting perspective than the apparent correlation of sub-percent price changes in the beginning. Actually, forecast 2 is also good by visual inspection, but one day early and linearly compressed amplitude: a plateau lasting one day, a valley lasting one day and then a longer plateau.

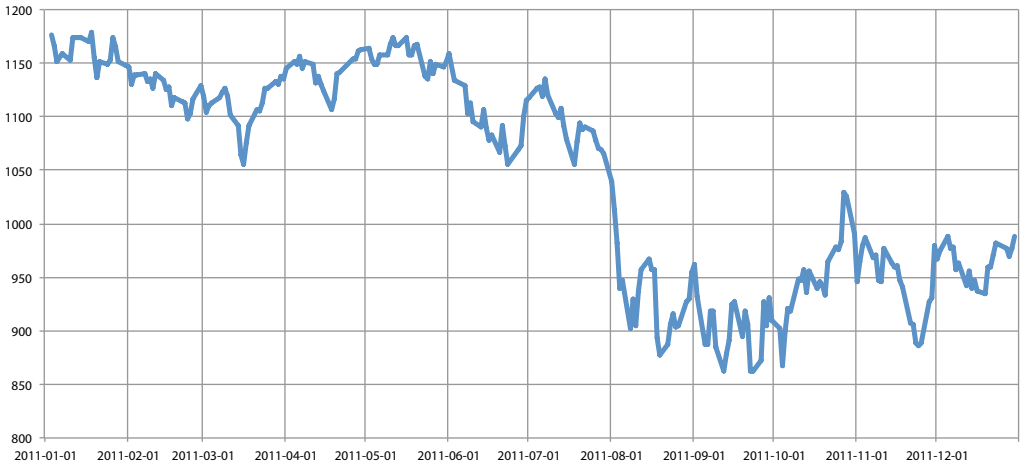


Figure 14.3: OMXS30 2011.

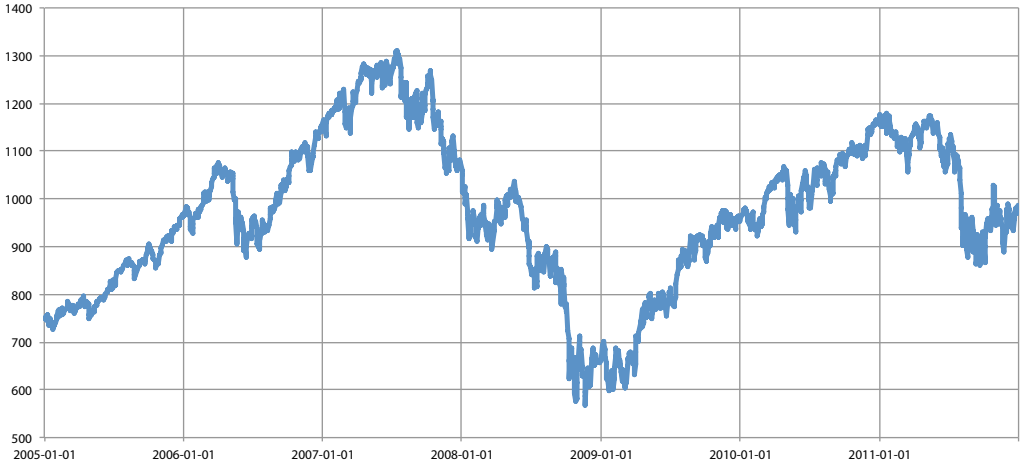


Figure 14.4: OMXS30 2005 to 2011 (inclusive).

As can be seen in Figure 14.3, 2011 was a special year—like every other year. Until August, the volatility was low and the price had declined about 10 % over the year. In just a few days, the price dropped 15 %, and the rest of the year was highly volatile. This causes a problem for the 5-fold evaluation, since the training is mostly performed on the first half of the year and the evaluation is performed during the second half. Hence, the forecasts, based on low volatility training data, will systematically have too low amplitudes.

A weakness in the evaluation of the Vanir system is illustrated in Figure 14.5. As can be seen, the forecasts overlap. If an article a_1 is published at time t_1, and then another article a_2 gets published at time $t_1 + \delta t$, the information from a_1 might become irrelevant and obsolete. A trader would reconsider his or her position in the instrument after seeing the new article and possibly reject the previous forecast. The evaluation does not take this into consideration, hence not reflecting the real behavior of a trader.

The headlines of the six articles in Figure 14.5 are presented below. Note that article number 5 did not invoke a forecast, since it was published when the market was closed.

1. "Atlas Copco: Köper bolag inom kompressorteknik USA, OMS 240 MKR"
   "Atlas Copco: Buys compressor technology company, USA, turnover 240 MSEK"
2. "Atlas Copco: Köper svenska GIA underjordsmaskiner m OMS 230 MKR"
   "Atlas Copco: Buys Swedish GIA underground machines with turnover 230 MSEK"
3. "Atlas Copco: Moodys höjer utsikter till positiva (stabila)"
   "Atlas Copco: Moodys raises outlook to positive (stable)"
4. "Atlas Copco: Moodys höjer utsikter till positiva (stabila) (ny)"
   Supplementary article to number 3.
5. "Atlas Copco: Köper tillgångar i kinesiskt bolag"
   "Atlas Copco: Buys assets in Chinese company"
6. "Atlas Copco: Tar över FSG Borrutrust från distributör Colombia"
   "Atlas Copco: Takes control over sales of drilling equipment from distributor in Colombia"

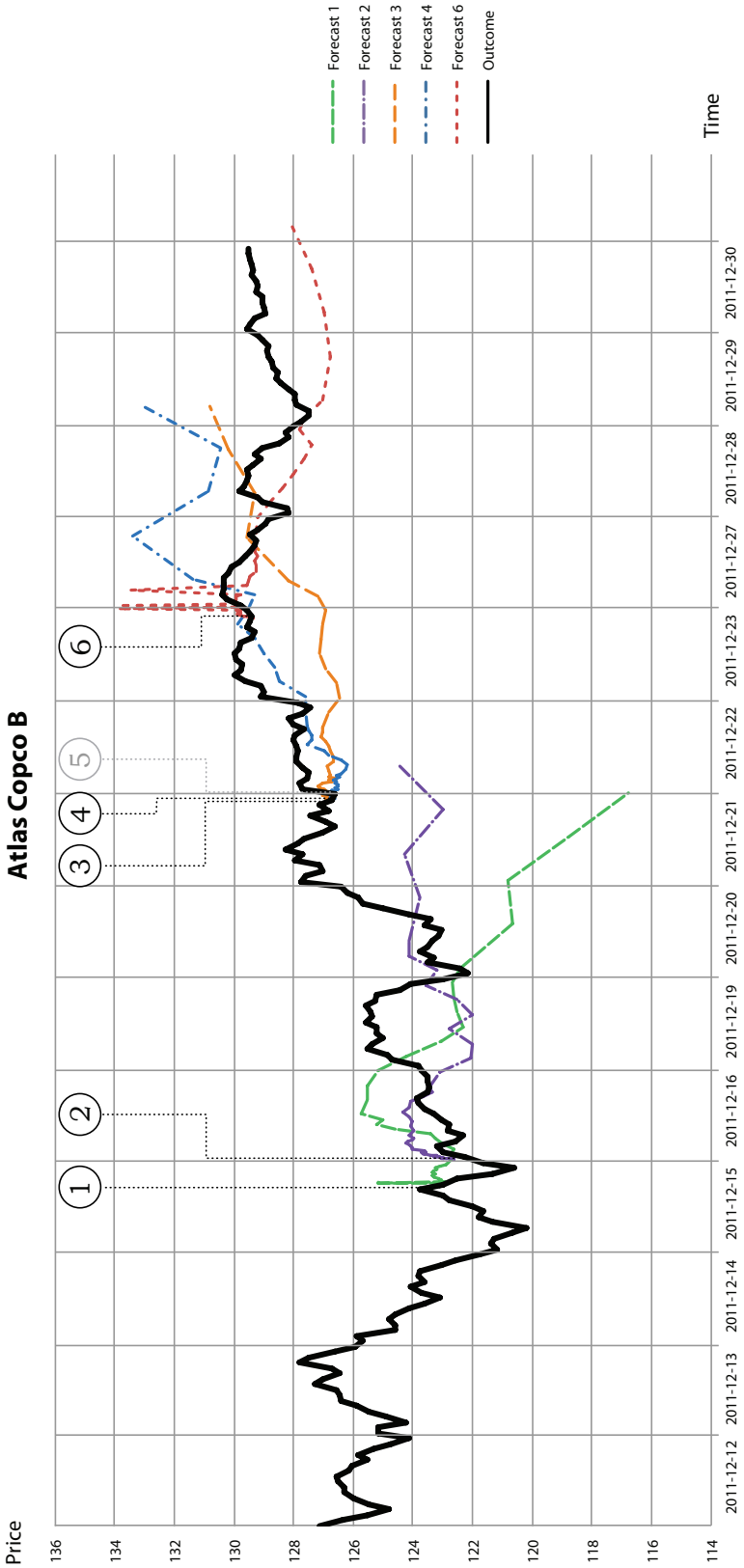


Figure 14.5: The stock price of Atlas Copco B (black continuous line) during the latter half of December 2011. The other, colored, lines are aggregated forecasts based on articles from Nyhetsbyrån Direkt. The articles of SIX News were removed from the figure for pedagogical reasons.

14.2.3 Trends

The forecasts showed a bias towards negative forecasts, due to the trend in the training data. When the outcome at the same time showed a positive trend, the systematic error was evident. These two trends are apparent in Figure 14.3 (before and after the turn of the month July to August, respectively). In some of the previous research, the trend was extracted from the time-series data before training. The current trend when forecasting was then added to the forecast before it is finally presented. If trained with enough data, the wkNN should be able to reflect and differentiate between the different trends and no such feature filtering should be needed. However, this would demand that all patterns had been seen before in all trends. Trend neutral sequences would require less training data to distinguish the patterns. Also, the forecasts would show better dynamics since any trend or baseline could be added afterwards; that is, in the forecast aggregation. However, this not straightforward since the current trend through t_0 needs to be estimated. It must be valid until the end of the forecast, which is yet unseen.

14.3 Further research

14.3.1 Price Pattern Analysis

The seemingly systematic time varying impaired performance could possibly—at least parts of—be derived from what sequences being trained on; the price pattern analysis training is not news-driven. It is reasonable to believe that at least some reaction occurs on the stock market when a news article is published (and later readjustment due to overreaction). The timing of these patterns are key when later generating news-driven forecasts. Thus, instead of sampling t_0 every twenty minutes, it would be interesting to see if the performance changed if the training was news-driven just like the forecasting.

In n-space, the distances can become huge. From time to another, it is reasonable that a prototype vector will be significantly closer to its closest neighbor than to its second or third. This would cause specific observations to influence the forecast to an excessive extent. Feature space analysis could remedy the problem. Is it dispersed or clustered? Density? It is obvious that the stretch in the dimensions corresponding to distant sample points is much wider than in the ones closer to t_0, but how much does this affect the distance? How much are the short-term dimensions suppressed? What if the dimensions were normalized individually?

Feature space evaluation might conclude that inverse Euclidian distance is not a suitable weight. Mahalanobis distance could be a better alternative, since it is scale-invariant and takes correlation into consideration. Maybe the weights should be non-linear. Maybe fixed weights should be applied instead (for the first, second and third neighbor and so on). Since these are weights in a weighted sum, finding them could be a part of the training.

14.3.2 Vanir

Since the aggregator of the Vanir system was no success, it could be of interest to see research on alternative aggregation methods.

A forecast has in general lower dimensionality than the original data it is based on. Hence, aggregating two already compiled forecasts from two separate analysis modules could mean that important information has already been obscured. It could be better to aggregate all data in one model directly. This means that the forecaster can take all data into consideration when producing the forecast. However, having two forecasters, different and more appropriate machine learning techniques could be used for each forecasting problem. It would be interesting to evaluate the same feature extraction setup as in Vanir, but with one machine learning model.

As described in Section 14.2.2, during evaluation it would be better to use linear sample distance. Furthermore, to reflect the behavior of a trader the evaluation of a certain forecast should stop as soon as there is a newer forecast available (of the same company). That is, key values like hit rate and such should be calculated until the newer forecast becomes available.

Bibliography

Paul Abbondante. 2010. Trading Volume and Stock Indices: A Test of Technical Analysis. *American Journal of Economics and Business Administration*, 2(3): 287–292. URL http://thescipub.com/pdf/10.3844/ajebasp.2010.287.292.

Lucy F. Ackert, Bryan K. Church, and Richard Deaves. 2003. Emotion and financial markets. *Economic Review*, (Q2):33–41. URL http://www.frbatlanta.org/filelegacydocs/ackert_q203.pdf.

Lawrence Blume, David Easley, and Maureen O'Hara. 1994. Market Statistics and Technical Analysis: The Role of Volume. *Journal of Finance*, 49(1):153–81. URL http://ideas.repec.org/a/bla/jfinan/v49y1994i1p153-81.html.

Fredrik Cedervall and Johannes Elgh. 2011. Snabba Cash: En modell för kortsiktig aktiehandel. URL http://www2.fek.su.se/uppsats/uppsats/2010/Kandidat15/216/snabba_cash_kandidat_HT10.pdf.

Chee Yong Chai and Shakirah Mohd Taib. 2010. *Machine Learning and Systems Engineering*, chapter 37: Hybrid Stock Investment Strategy Decision Support System: Integration of Data Mining, Artificial Intelligence and Decision Support. Lecture Notes in Electrical Engineering. Springer. ISBN 9789048194186. URL http://books.google.com/books?id=BRZm4DRptMQC.

Pei-Chann Chang, Chen-Hao Liu, Chin-Yuan Fan, Jun-Lin Lin, and Chih-Ming Lai. 2009. An ensemble of neural networks for stock trading decision making. In *Proceedings of the Intelligent computing 5th international conference on Emerging intelligent computing technology and applications*, ICIC'09, pages 1–10, Berlin, Heidelberg. Springer-Verlag. ISBN 3-642-04019-5, 978-3-642-04019-1. URL http://dl.acm.org/citation.cfm?id=1788154.1788156.

Anderson da Silva Soares, Maria Stela Veludo De Paiva, and Clarimar José Coelho. 2007. Technical and Fundamental Analysis for the Forecast of Financial Scrip Quotation: An Approach Employing Artificial Neural Networks and Wavelet Transform. In *Proceedings of the 4th international symposium on Neural Networks: Advances in Neural Networks, Part III*, ISNN '07, pages 1024–1032, Berlin, Heidelberg. Springer-Verlag. ISBN 978-3-540-72394-3. URL http://dx.doi.org/10.1007/978-3-540-72395-0_125.

Shangkun Deng, Takashi Mitsubuchi, Kei Shioda, Tatsuro Shimada, and Akito Sakurai. 2011. Combining Technical Analysis with Sentiment Analysis for Stock Price Prediction. In *DASC*, pages 800–807. IEEE.

Johannes Elgh. 2012. Machine learning news analysis for stock trading. Master's thesis, KTH Royal Institute of Technology.

David Enke and Suraphan Thawornwong. 2005. The use of data mining and neural networks for forecasting stock market returns. *Expert Syst. Appl.*, 29(4):927–940. ISSN 0957-4174. URL `http://dx.doi.org/10.1016/j.eswa.2005.06.024`.

Eugene F Fama. 1970. Efficient Capital Markets: A Review of Theory and Empirical Work. *Journal of Finance*, 25(2):383–417. URL `http://ideas.repec.org/a/bla/jfinan/v25y1970i2p383-417.html`.

Yi Feng, Ronggang Yu, and Peter Stone. 2004. Two Stock-Trading Agents: Market Making and Technical Analysis. In Peyman Faratin, David C. Parkes, Juan A. Rodriguez-Aguilar, and William E. Walsh, editors, *Agent Mediated Electronic Commerce V: Designing Mechanisms and Systems*, volume 3048 of *Lecture Notes in Artificial Intelligence*, pages 18–36. Springer Verlag. URL `http://www.cs.utexas.edu/users/ai-lab/pub-view.php?PubID=126608`.

Thomas Hellström. 1998. *A Random Walk through the Stock Market*. PhD thesis, Umeå University.

K. Senthamarai Kannan, P. Sailapathi Sekar, M. Mohamed Sathik, and P. Arumugam. 2010. Financial Stock Market Forecast using Data Mining Techniques. *Lecture Notes in Engineering and Computer Science*, 2180(1):555–559.

Jordan Kotick. 2003. The Fundamentals of Technical Analysis: Patterns and Predictions. *Alchemist*, 30:14–16. URL `http://www.lbma.org.uk/assets/alch30_techtrading.pdf`.

K. Kozak, M. Kozak, and K. Stapor. 2005. Weighted k-Nearest-Neighbor Techniques for High Throughput Screening Data. *Life Sciences*, 1(3):155–160.

Jin Li and Edward P. K. Tsang. 1999. Improving technical analysis predictions: An application of genetic programming. In *Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference*, pages 108–112. AAAI Press. ISBN 1-57735-080-4. URL `http://dl.acm.org/citation.cfm?id=646812.707186`.

Wei Liu and Sanjay Chawla. 2011. Class confidence weighted kNN algorithms for imbalanced data sets. In *Proceedings of the 15th Pacific-Asia conference on Advances in knowledge discovery and data mining - Volume Part II*, PAKDD'11, pages 345–356, Berlin, Heidelberg. Springer-Verlag. ISBN 978-3-642-20846-1. URL `http://dl.acm.org/citation.cfm?id=2022850.2022879`.

Andrew W. Lo and A. Craig MacKinlay. 1988. Stock market prices do not follow random walks: evidence from a simple specification test. *Review of Financial Studies*, 1(1):41–66. URL `http://rfs.oxfordjournals.org/content/1/1/41.abstract`.

Marilyn McDonald. 2007. *Forex Simplified: Behind the Scenes of Currency Trading*. Marketplace Books. ISBN 9781592803163.

Cheol-Ho Park and Scott H. Irwin. 2007. What Do We Know About the Profitability of Technical Analysis? *Journal of Economic Surveys*, 21(4):786–826. URL `http://doi.wiley.com/10.1111/j.1467-6419.2007.00519.x`.

Aurobinda Prasad. 2005. Technical Analysis. URL `http://www.karvycomtrade.com/downloads/Technical%20Analysis.pdf`.

Reza Raei, Shapour Mohammadi, and Mohammad Mehdi Tajik. 2011. An Intelligent technical analysis using neural network. *Management Science Letters*, 1(3): 355–362.

Barbara Rockefeller. 2011. *Technical Analysis For Dummies*. For Dummies. John Wiley & Sons. ISBN 9780470888001. URL `http://books.google.se/books?id=UfJnGpGTD5QC`.

Andy Tan, Hiok Chai Quek, and Kin Choong Yow. 2008. Maximizing winning trades using a novel RSPOP fuzzy neural network intelligent stock trading system. *Applied Intelligence*, 29(2):116–128. ISSN 0924-669X. URL `http://dx.doi.org/10.1007/s10489-007-0055-1`.

Gary Gang Tian, Guang Hua Wan, and Mingyuan Guo. 2002. Market Efficiency and the Returns to Simple Technical Trading Rules: New Evidence from U.S. Equity Market and Chinese Equity Markets. *Asia-Pacific Financial Markets*, 9 (3–4):241–258.

Appendix A

w*k*NN results for different *k*-values

		Outcome			
		UP	DOWN	ZERO	Σ
Forecast	UP	60 212	57 008	6 660	123 880
	DOWN	64 483	61 646	6 618	132 747
	ZERO	2 445	2 390	870	5 705
	Σ	127 140	121 044	14 148	262 332

(a) k=5

		Outcome			
		UP	DOWN	ZERO	Σ
Forecast	UP	59 182	56 173	6 572	121 927
	DOWN	64 979	61 896	6 507	133 382
	ZERO	2 979	2 975	1 069	7 023
	Σ	127 140	121 044	14 148	262 332

(b) k=10

		Outcome			
		UP	DOWN	ZERO	Σ
Forecast	UP	58 633	54 379	6 363	119 375
	DOWN	64 612	62 882	6 440	133 934
	ZERO	3 895	3 783	1 345	9 023
	Σ	127 140	121 044	14 148	262 332

(c) k=20

		Outcome			
		UP	DOWN	ZERO	Σ
Forecast	UP	57 590	53 894	6 203	117 687
	DOWN	65 050	62 702	6 239	133 991
	ZERO	4 500	4 448	1 706	10 654
	Σ	127 140	121 044	14 148	262 332

(d) k=30

Figure A.1: wkNN confusion matrices.

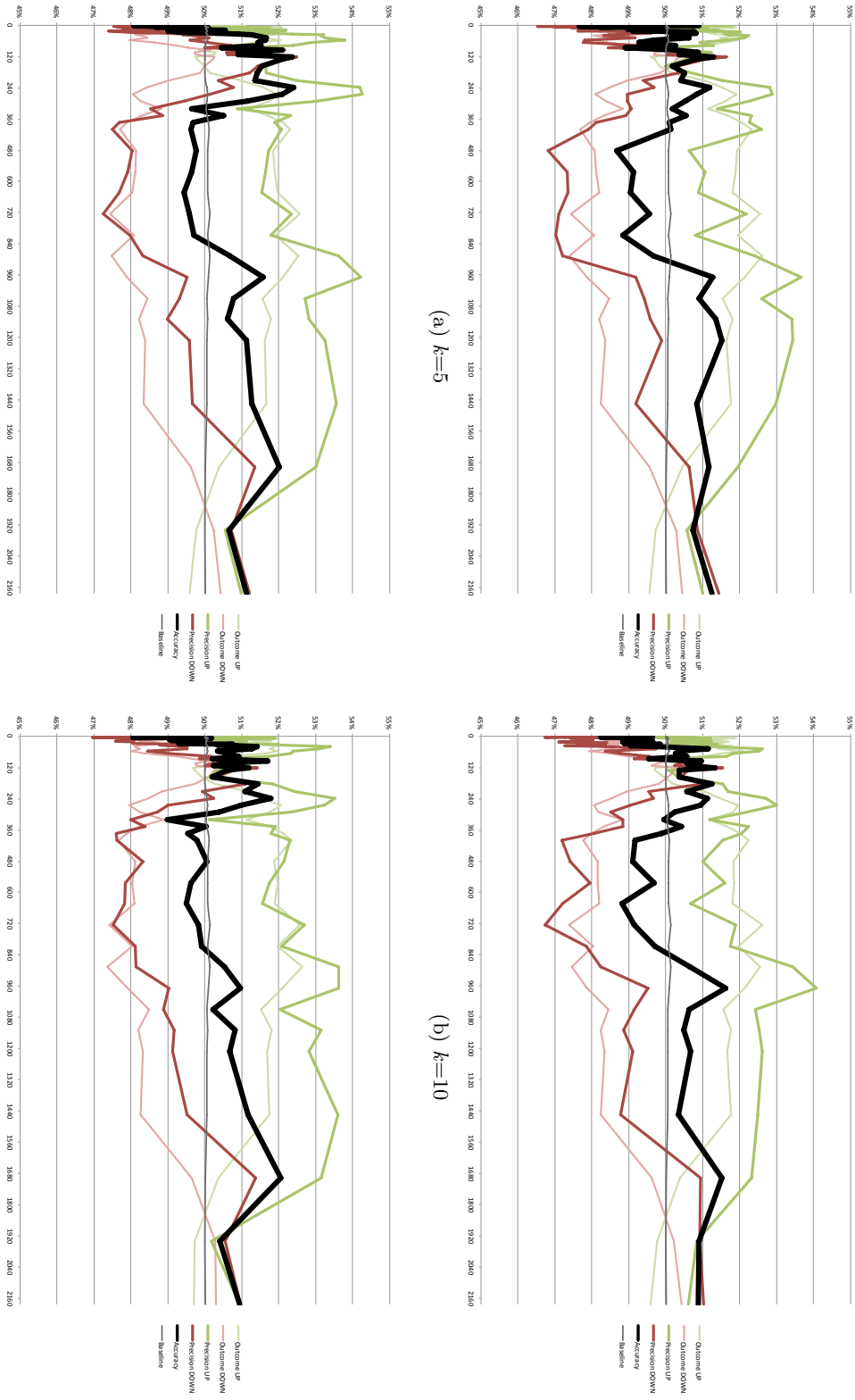
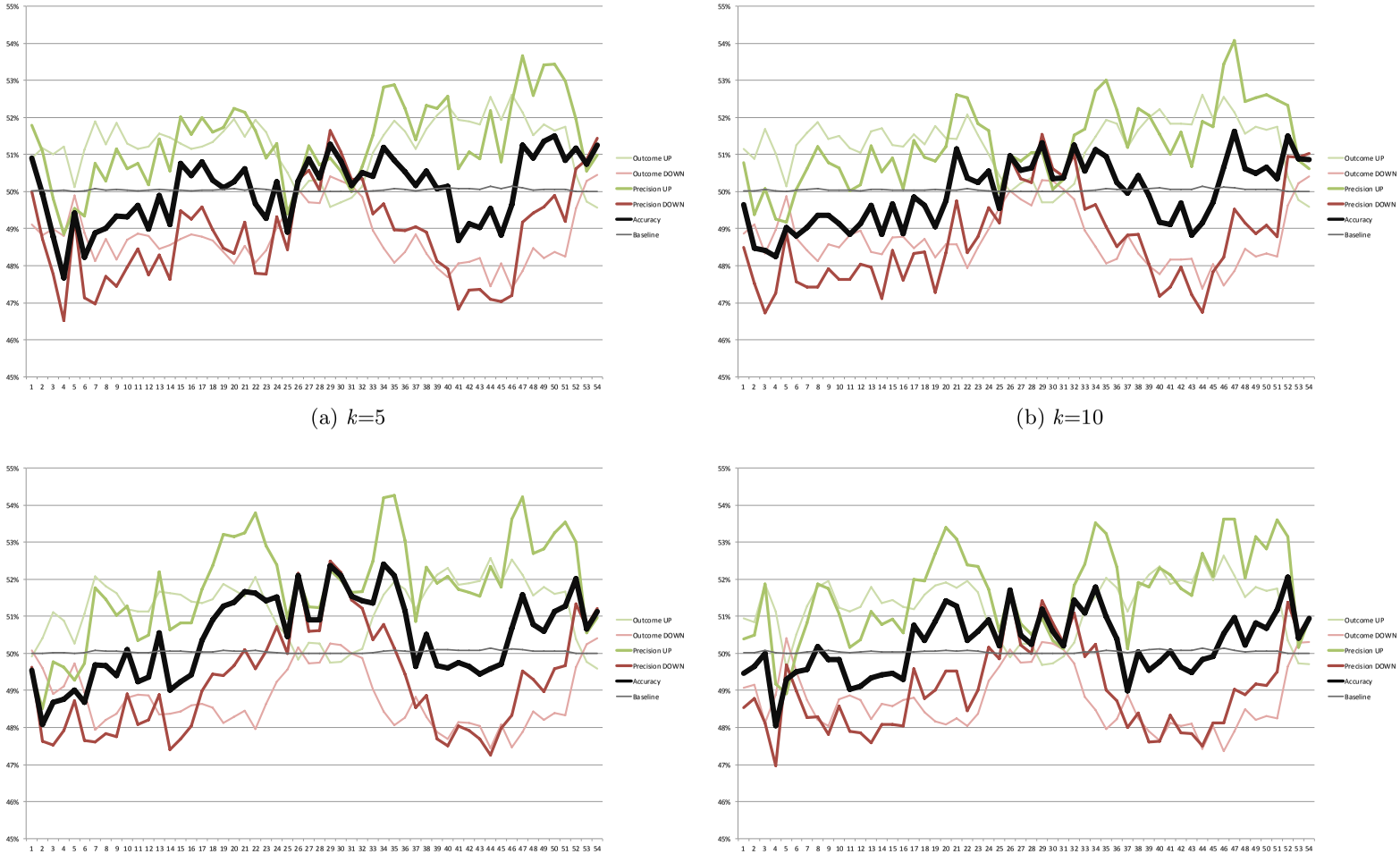


Figure A.2: wkNN hit rate over time, with time scale in minutes.



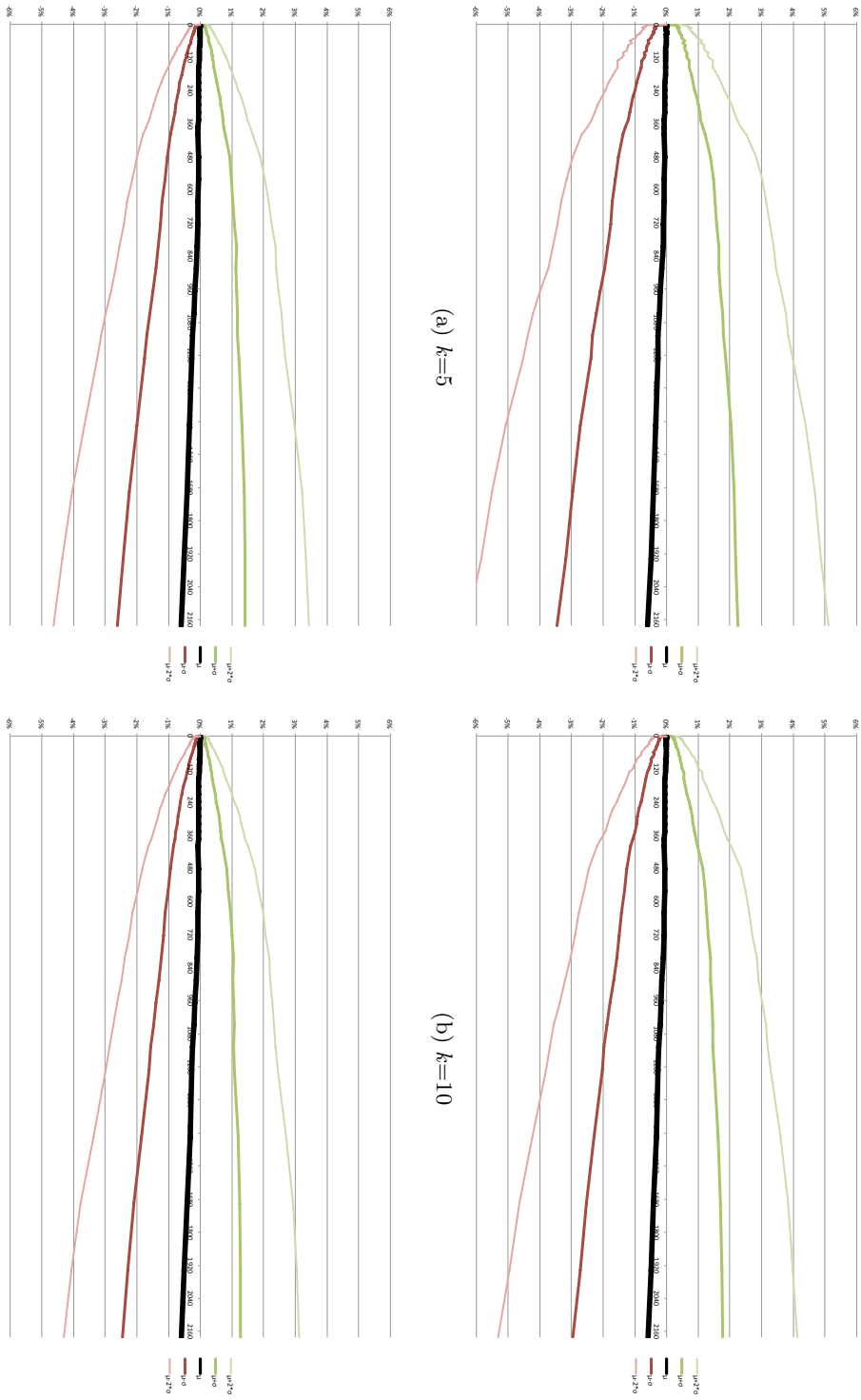
(a) k=5

(b) k=10

(c) k=20

(d) k=30

Figure A.3: wkNN hit rate over time, with time step scale.



(a) k=5 (b) k=10

(c) k=20 (d) k=30

Figure A.4: wkNN forecast distribution over time, with time scale in minutes.

TRITA-CSC-E 2012:088
ISRN-KTH/CSC/E--12/088-SE
ISSN-1653-5715

www.kth.se