Systems Identification of an Impossible System:  
An Empirical Study of Models for Stock Market Prediction

Semester Thesis

May 15, 2007

Georg Schneider  
scgeorg@student.ethz.ch

Supervisor: Prof. Ivo Sbalzarini
Abstract

This work describes the development of machine learning models and their application to automatic trading algorithms. The goal is to predict future prices of selected stock based only on past data. Primary focus is the construction and selection of useful features. We then perform model selection concentrating on neural network methods. A simple alternative method for prediction is also presented. These models are then employed for decision making of automatic trading algorithms with the goal of using the gained information as good as possible. We show that selected algorithms perform well on previously unseen data with the restriction the generalization onto other dataset is not successful. In our tests the presented methods perform up to two times better than the market average.
# Table of Contents

1. Introduction ................................................................................................................. 4  
   1.1. Motivation ................................................................................................................ 4  
   1.2. Goal ........................................................................................................................... 4  
   1.3. Scope of Work ......................................................................................................... 4  
2. Overview ......................................................................................................................... 5  
   2.1. Tool Evaluation ....................................................................................................... 5  
   2.2. Data Selection ....................................................................................................... 5  
   2.3. Feature Extraction ................................................................................................. 7  
3. Building Predictors ......................................................................................................... 8  
   3.1. Determining Optimal Time Window Size ............................................................... 8  
   3.2. Analyzing Feature Quality .................................................................................... 9  
   3.3. Optimizing the Network Architecture ................................................................. 12  
   3.4. Cross-validating Predictors .................................................................................. 14  
   3.5. Alternative Prediction Methods .......................................................................... 17  
4. Using Predictions for Trading ...................................................................................... 19  
   4.1. Developing Trading Strategies ............................................................................. 19  
   4.2. Assessing the Methods ....................................................................................... 20  
   4.3. Cross-validating Trading Strategies ..................................................................... 23  
5. Conclusions ..................................................................................................................... 25  
   5.1. Results ...................................................................................................................... 25  
   5.2. Further work .......................................................................................................... 25  
   5.3. Personal Experience .............................................................................................. 26  
6. References ...................................................................................................................... 27
1. Introduction

1.1. Motivation

There are two aspects regarding the use of machine learning techniques. First, people try to mimic human behaviour when it comes to tasks like recognizing objects in an image. On the other hand, these methods are often used for analysing and processing amounts of data, which are too large for humans to be feasible. The goal then is to identify hidden coherences or patterns which can give information about the underlying domain.

When looking for a topic for my semester thesis the problem of stock marker prediction immediately caught my interest. It is a problem that is thought to be hard, or even impossible to solve. Yet, just a tiny lead in information compared to other market participants would lead to a valuable trading advantage. Being interested in the field of machine learning, this project constituted a good starting point for gaining experience in this area. I especially liked the idea of employing methods, which I had studied in courses before, on a real-world problem. The experimental approach which provided much freedom in choice of techniques and tools was also appealing to me.

1.2. Goal

The purpose of this work is to assess the performance of various prediction models on different market data. I focused on forecasts of the real stock price rather than just classification of the directional movement. Predicting only the direction of change is a more feasible task but it lacks information about the amount of change of the price. As the final goal is to evaluate the trained model using various trading strategies, this information is important for decision making of the algorithm. As a result, different models are assessed with respect to prediction accuracy and usefulness for trading algorithms.

1.3. Scope of Work

The thesis is composed of six chapters. In this chapter, the motivation, purpose and scope of the thesis are described. The second one gives an overview over the applied tools as well as the datasets and features that were used to train the various models. In the third chapter conducted experiments are described investigating various model setups and their performance on the different datasets. Also, information quality of these predictions with respect to trading is examined. Chapter four contains an evaluation of different trading strategies, which make use of the information obtained from the made predictions. The behaviour of the various algorithms is observed and described in detail. In chapter five overall results are summarized and conclusions are drawn. Possible further work based on this thesis is proposed and personal experiences are reported. The last chapter then contains references to all used resources.
2. Overview

2.1. Tool Evaluation

Several possibilities have been considered for conducting the learning experiments. Implementing the used techniques with the help of machine learning libraries (like libSVM [1]) from scratch wasn’t considered an option, as the focus of the work should lie on experimenting rather than on programming. Another possibility was to use the well-known Spider package for Matlab© ([2]) developed at the Max Planck Institute for Biological Cybernetics in Tübingen / Germany. The reason for not using this toolbox is that it is rather complicated to get started with and programming in Matlab© doesn’t offer the desired flexibility and easy of use. Two other frameworks have been looked at. The VFML tools ([3]) from the University of Washington / USA are rather focused on high performance algorithms and are only available from the command line. The Weka toolkit ([4]) developed at the University of Waikato / New Zealand offers wide range of algorithms and is accessible via a graphical user interface. My decision finally fell for another application. The YALE environment ([5]), an open-source framework maintained by the Rapid-I company, provides an easy-to-use interface and besides wrapping the Weka algorithms it provides additional techniques for processing, manipulating and visualizing data. It also implements sophisticated methods for meta-optimization of feature and parameter selection. Experiments are represented by a tree-like structure and are very intuitive to setup.

2.2. Data Selection

To be able to validate the generalization capability of the models selected stocks and indices should be of different nature. Technology related financial products, for example, have been under heavy influence by the dot com bubble around the year 2000, whereas other industries haven’t been at all.

The source of all market data used is the Yahoo© finance web page ([6]). They offer free historical price data on a daily basis and have a broad selection of stocks and indices available. The data is downloadable in an easily processable format.

Stocks from different companies in various industry sectors are investigated in this thesis. Three big companies and a small one have been chosen: the IBM Corporation (technology), the Novartis company (drug manufacturer) and Kraft Foods Inc. (Foods) represent major players in the respective stock market. The Amazon Biotech Inc. from the biotechnology sector serves as an example for a young and rather small company. Two compound indices are also investigated: the Dow Jones industrial average and the NASDAQ composite index. The details of the datasets are described hereafter.

**IBM Dataset**
- Company: International Business Machines Corporation
- Market: New York stock exchange
- Sector: Technology
- Industry: Diversified Computer Systems
NVS Dataset
Company: Novartis AG
Market: New York stock exchange
Sector: Healthcare
Industry: Drug Manufacturers - Major
Data source: Yahoo© Finance
Examples: 2373
Features: 5

KFT Dataset
Company: Kraft Foods Inc.
Market: New York stock exchange
Sector: Consumer Goods
Industry: Food – Major Diversified
Data source: Yahoo© Finance
Examples: 1212
Features: 5

AMZB Dataset
Company: Amazon Biotech Inc.
Market: New York stock exchange
Industry: Biotechnology
Sector: Health Care
Data source: Yahoo© Finance
Examples: 490
Features: 5

DJI Dataset
Index: Dow Jones industrial average
Market: New York stock exchange
Data source: Yahoo© Finance
Examples: 19491
Features: 5

IXIC Dataset
Index: NASDAQ Composite index
Market: NASDAQ
Data source: Yahoo© Finance
Examples: 5449
Features: 5
2.3. Feature Extraction

The raw data from Yahoo© contains opening and closing, as well as daily high and low prices. The total trading volume is also included for each day. To extend the feature space, we want to extract additional properties from the data. With financial instruments this can be done by calculating common market indicators, which are also used by traders for decision making. The description of these indicators is explained hereafter and their interpretation for trading purposes is discussed later in the report.

**Simple Moving Average** (SMA): This is the average closing price over a certain period of time. Usual time ranges are 40 and 200 days, but I also included the SMA over ten and 100 days.

**Exponential Moving Average** (EMA): The EMA is also the average closing price, but included days in the past are weighted exponentially decreasing. This indicator is calculated for twelve and 26 days.

**Moving Average Convergence / Divergence** (MACD): The MACD is simply the difference between a short and a long period EMA. Typically, the twelve and 26 day EMAs are used for it. Often the MACD is interpreted together with the so-called signal line, which is the 9 days EMA of the MACD.

**Volatility** (VOL): Volatility is usually referred to as the standard deviation of the change of price for a certain time horizon. It is calculated for ten, 40, 100 and 200 days.

**Money Flow Index** (MFI): The money flow for a day is calculated by multiplying the daily trading volume by the typical price (average of high, low and closing price). If the typical price is higher than on the previous day, one speaks of positive money flow, and vice versa. The money flow index then is the percentage of positive money flow to the total money flow for a certain period of time. It is calculated for ten, 40, 100 and 200 days.

**Rate of Change** (ROC): The momentum is defined as the difference between today’s closing price and that of N days before. The rate of change is the ratio between the momentum and the closing price N days ago. It is also calculated for the usual periods (10, 40, 100 and 200).

**Stochastic oscillators** (%K, %D): These indicators generally compare today’s closing price to the total range of the price over a certain period. The %K oscillator, for example, is defined by the ratio of the difference between today’s closing price and the lowest closing price of the last N days to the difference between the highest and the lowest closing price of the last N days. The %D oscillator is the SMA over three days of the %K indicator. I calculated these respectively for ten, 40, 100 and 200 days.

**Relative Strength Index** (RSI): The RSI represents the ratio of average upward movements of the closing price to the average overall movement over a specified period of time. The EMA is used for averaging the movements. For convenience, I calculated this index for twelve and 26 days, as these EMAs are already available.
3. Building Predictors

The stock of the IBM Corporation serves as an entry point for building and testing initial models. This is mainly because prices are available from 1962 on, so there is much data to train the predictors. Also, IBM is a big company and its stock isn’t that volatile than stocks of smaller companies. The biggest challenge with this dataset is to make good predictions on the part of the test data, where the price rises to levels not present in the training data (the period of the dot com bubble around the year 2000). Good accuracy here would mean that the model generalizes well to previously unseen data. The conducted experiments will concentrate on neural networks as predictors as they are known to perform well for such tasks.

3.1. Determining Optimal Time Window Size

The first experiment will look at a simple setup using only past closing prices for prediction. The time window of these will be varied to see how much past data should be used to train the classifier. The performance of this predictor will allow us later to estimate the benefit of adding additional features in the future.

The size of these windows is varied from three up to 25 days. To be able to compare the models on the same data we fix the number of examples in the test set to 5000. This leaves us around 6100 training examples depending on the time frame used. The closing price of the following trading day is used as prediction target. The whole dataset is normalized prior to learning to the range \([0, 1]\).

We employ a Multilayer Perceptron with one hidden layer containing five hidden units with a sigmoid activation function. The network is trained with the backpropagation algorithm over 200 epochs using a learning rate of 0.3 and a momentum parameter of 0.2.

Assessment of the different models is done in two ways. On the one hand we calculate the average relative error

\[
e = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{\text{price}_{\text{true}} - \text{price}_{\text{predicted}}}{\text{price}_{\text{true}}} \right|
\]

on the test set for measuring prediction quality (\(n\) being the number of test examples). As we want to use the models as a basis for trading algorithms later we also define a directional hit rate measure. A hit is obtained if the predicted movement \((\text{prediction}_{\text{tomorrow}} - \text{prediction}_{\text{today}})\) has the same sign as the true movement of the price. We then calculate the ratio of correct directional predictions on the test set. As the models generally fail to predict situation where there is no movement of the price at all we don’t include these cases in the calculation. The two measures are connected in that, if the error is zero, the hit rate is 100%. Otherwise there is no direct relationship between them.

Unfortunately, the toolkit doesn’t provide a way to incorporate user-defined performance measures. Therefore we can only train the predictors with respect to built-in measures like the relative error. The hit rate will just serve as an indicator for usefulness for trading purposes.

Figure 1 shows the two performance measures for different window sizes.
We observe here that the two performance measures don’t necessarily correlate with each other. A window size of five is optimal with respect to the error, whereas the hit rate is highest with a size of 15. As we don’t know yet, which criterion is more important for decision making of the trading algorithms later, we pursue further the best models of both quality measures.

### 3.2. Analyzing Feature Quality

After having investigated the window sizes we now assess all features (including those described in section 2.3) with respect to their usefulness for prediction. Because of the computational complexity of the applied algorithms we first restrict ourselves on the unwindowed features of a single day only.

The methods try to optimize feature sets iteratively by improving individual sets using evolutionary strategies. A feature set is represented as a weight vector. A feature weighted one is considered selected, whereas a weight of zero means that the feature isn’t selected. At each step of the algorithm a population of features sets is maintained, out of which the best performing feature sets are mutated (see next paragraph) in order to find better ones. Performance of a feature set is determined as follows: the features are used to train a simple neural network using a training set of 500 examples. The resulting relative error of this network on a test set (again of 500 examples) is then used to rank the feature set. The algorithms are considered converged if there is no improvement of the best found feature set over five generations.
The following mutation schemes are employed: the first method used is called forward selection. Starting from the empty set features leading to the biggest error reduction when included are successively added until convergence (as described in the previous paragraph) is reached. The resulting feature set is called `for_sel`. An adapted version of this approach, called forward weighting, adds features by applying a certain non-zero weight to it (feature set `for_wei`). The weights are restricted to the values 0, 0.25, 0.5, 0.75 and 1 in order to still have a discrete search space. The opposite procedures are called backward selection (`back_sel`), respectively backward weighting (`back_wei`). There, one starts from the full feature set and successively removes features (respectively decreases weights). Obviously, these methods are likely to get stuck in a local optimum, so we employ another more sophisticated technique, called genetic feature selection / weighting (feature sets `evo_sel` / `evo_wei`). The selection algorithm mutates feature sets by selecting / deselecting features with a probability of 1 / (number of features). The weighting approach changes the weight of each feature according to a Gaussian distribution. The process is stopped when convergence is reached.

For comparison, we also generate a few more feature sets manually. Those include features that have a high correlation with the target and thus appear to be useful. The full feature set is also included. In figure 2 the selected features (respectively features with high weights) are listed for the different methods. Manually created set are prefixed `man` here.

<table>
<thead>
<tr>
<th>index</th>
<th>feature set name</th>
<th>included features / highly weighted features</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>man_1</td>
<td>all features</td>
</tr>
<tr>
<td>2</td>
<td>man_2</td>
<td>volume, open, high, low and close</td>
</tr>
<tr>
<td>3</td>
<td>man_3</td>
<td>man_2 + moving averages (all SMA and EMA)</td>
</tr>
<tr>
<td>4</td>
<td>man_4</td>
<td>man_3 + MACD, SIGNAL</td>
</tr>
<tr>
<td>5</td>
<td>man_5</td>
<td>man_4 + VOL(all)</td>
</tr>
<tr>
<td>6</td>
<td>man_6</td>
<td>man_2 + VOL(all) + ROC(all)</td>
</tr>
<tr>
<td>7</td>
<td>for_sel</td>
<td>volume, high, close, MACD, VOL(all), ROC-10</td>
</tr>
<tr>
<td>8</td>
<td>for_wei</td>
<td>high, close, ROC-40</td>
</tr>
<tr>
<td>9</td>
<td>back_sel</td>
<td>all except SMA-100, MFI-10 and %D-100</td>
</tr>
<tr>
<td>10</td>
<td>back_wei</td>
<td>all except EMA-26 and %K-200</td>
</tr>
<tr>
<td>11</td>
<td>evo_sel</td>
<td>volume, open, low, close, SMA-40, SMA-100, EMA-26, MACD, VOL-40, MFI-10, MFI-100, ROC-100, ROC-200, %K-40, %K-100, %K-200 and %D-200</td>
</tr>
<tr>
<td>12</td>
<td>evo_wei</td>
<td>volume, close, SMA-10, SMA-40, EMA-26, MACD, VOL-200, MFI-10, MFI-40, MFI-200, ROC-40, ROC-200, %K-10, %D-10</td>
</tr>
</tbody>
</table>

Fig. 2: Generated feature sets. Manually created set are prefixed `man_`, the others are generated by feature selection algorithms.

We notice here, that heuristic feature searches indicate that the closing price is most relevant for making predictions as it is included in all sets. Also, volume and volatility seem to be beneficial to performance.

As a next step the presented feature sets are used to train the very same neural network as in section 3.1 on the whole dataset. We again fix the size of the test set to 5000 examples and train the predictor with the remaining data. Figure 3 compares the various feature sets with respect to the two performance measures introduced in the previous section.
Fig. 3: Comparison of different feature sets. The lowest prediction error of 2.07% is obtained using forward selection (for_sel), whereas a manually created feature set (man_5) results in the highest hit rate (50.51%).

We again see that a low prediction error doesn’t necessarily mean a high hit rate. Two different sets perform best on the two quality measures. Generally, one observes that the automatic selection methods generate models with relatively low hit rates, whereas the manually selected set do well on this measure. The reason could be that the heuristic algorithms assess their intermediate feature sets only with respect to the relative prediction error, totally neglecting the hit rate criterion. This leads to low generalization error, but hit rates can nevertheless be quite low.

A general observation is that using too many features generally hurts performance. Using all available features leads to the worst average error and a low hit rate. Backward methods fail to eliminate useless features because they get stuck early in a local minimum. Evolutionary techniques do better but also fail to find really good subsets. Forward selection and weighting works best with respect to prediction error, although very little features are selected and hit rates are quite low. The reason of failure of the heuristic methods could be that they work on a small part of the dataset containing only 1000 examples. Features sets that perform well on this part don’t necessarily lead to good performance on the whole dataset.

From this experiment we further investigate the following feature sets: man_2 with a high hit rate and a low relative error, man_5 with the highest hit rate and forSel with the lowest prediction error. From these feature sets we now create windows of several days, according to our results from the first experiment. Figure 4 shows the resulting performance values for the different setups.
Fig. 4: Combination of best feature sets with optimal window sizes. Lowest error is obtained from for_sel with a window size of 5. Hit rate is best for man_5 and a window size of 5.

Compared to the previous results we observe that windowing the feature sets over several days leads to much better hit rates. The downside is that the relative error is also increasing dramatically. A possible reason for this is that windowing the features over several days creates much more features than in the previous experiments. The man_5 feature set with a window size of 15, for example, has 255 features. It could be that the architecture of the neural network we use is too simple to be able to handle such big feature sets. Therefore, we try to find an optimal network structure in the next experiment.

### 3.3. Optimizing the Network Architecture

As said before we assume that the high relative prediction error in the previous experiment results from a too simple network structure. Thus, we now try out different architectures and check, if the performance can be improved by that. From the previous experiment we retain the feature set with the lowest relative error as well as the two achieving the highest hit rates. The first one is the for_sel feature set with a window size of five (called forsel_w5 now), which now has 45 features. Then we use man_2 with a window of 15 days (man2_w15), now having 75 features. Finally we also try out man_5 windowed over five days (man5_w5), which now contains 85 features.

Networks with one and two hidden layers, containing different numbers of hidden nodes, are tested on the same data as in the previous experiments. Figure 5 shows the average relative prediction error for the three feature sets with different network structures.
Fig. 5: Average relative error for different network structures. Highest accuracy (3.5%) is obtained with the forsel_w5 feature set using a network with a single hidden layer containing 15 hidden nodes.

Figure 6 shows the hit rates for the various network structures.

Fig. 6: Hit rates for different network structures. On average the most promising candidates with respect to hit rate is a two layer network containing 15 respectively five hidden nodes.
Generally, increasing the complexity of the network not only helps reducing the relative error but also increases the hit rate of the predictor. But if the chosen architecture is too complex both quality measures suffer from it. We can observe that the forsel_w5 feature set has the lowest error but performs bad regarding the hit rate. The other two feature sets behave different. Their resulting error is high, but they also obtain higher rates than the setups from the previous sections. Two structures out of the tested ones are most promising: the single-layer network with 15 hidden nodes leads to a small average error on all feature sets and also results in the highest obtained hit rate (51.95%). The two-layer network with 15 and five hidden nodes doesn’t lead to low relative errors (7.07% on average) but results in the highest average hit rate of 51.5% averaged over the three datasets. These structures will be kept and investigated further in the next experiment. We want to see, how the best setups found so far perform on other datasets.

### 3.4. Cross-validating Predictors

Having found promising predictor setup for the *IBM* stock, it is now time to validate these on other datasets. The goal is to see if certain models generalize well or if we simply created a predictor suited for this single dataset. The following models are chosen according to the results from the previous experiment: for all three feature sets we train a single-layer network with 15 hidden nodes (suffix _s1) and a two-layer network containing 15 and five hidden nodes (suffix _s2). These models are rebuilt for all datasets presented in section 2.2. An overview over their performance is shown in figure 7 and 8, where the average relative error and the hit rate measure are plotted for the different datasets.

![Comparison of models for different datasets](image)

*Fig. 7: Average relative error of the models for different datasets. Errors are high for datasets containing very few examples (AMZB, KFT). For the other datasets performance is comparable to our previous experiments with the IBM dataset.*
We see here that on the bigger datasets the model error is low, comparable to the results from the IBM dataset. The best relative error on the test set (2.39%) is obtained with the NVS dataset using the man2_w15_s2 model. For the AMZB and KFT dataset errors are high, probably due to small training sets. Averaged over all datasets the forsel_w5_s1 model again performs best with respect to the relative error on the test sets.

Looking at the hit rates the picture is different. In general the rates are below 50%, which is rather bad. Only the IXIC dataset presents promising results having an average hit rate of 51.79%. This may be due to the similar shape and characteristics compared to the IBM dataset. The AMZB dataset shows an exceptional hit rate with the man2_w15_s1 model, which is rather surprising. It turns out to be coincidental, as several random runs of the training don’t confirm this result leading to much lower hit rates.

As an example for a good model (low error and high hit rate) figure 9 shows the true and the predicted price for the test examples of the IXIC test dataset. What looks good on the big scale shows some deficiencies if we zoom in on a small representative part of the plot (figure 10).
Fig. 9: Comparison of true and predicted price. The predicted curve resembles a smoothed version of the true price.

Fig. 10: Zoomed in version of figure 9 (first 200 examples). As already seen on the whole plot spikes at high frequencies are ignored by the prediction, which only reproduces a directional trend of the true price.
The obvious problem with this model is its late reaction to trend changes of the price. Hence we can’t rely on the daily forecasts of this model. Despite this deficiency one can observe that the general trend of the true price curve is followed well by the predictions. We can try to use this kind of information for making trading decisions, which is done in chapter 4.

3.5. Alternative Prediction Methods

After assessing different neural network models on the datasets we will now try out another technique, which is not based on machine learning methods. Let’s assume the data contains periodically repeating up- and downward trends. The idea is then to do a Fourier transform of the closing price series from the training set to calculate its frequency components. From this the inverse is calculated using only the some of the frequencies. This gives us a simple sinusoidal model of the base oscillation of the price in the training set. Applying this model to the test set allows us to do predictions of upward and downward trends according to the movement of the sinus curve.

First, we want to analyse the frequency components of our training data. For the calculation we use the same spilt of the datasets as before. As an example figure 11 shows the power spectrum of the training examples from the *IBM* dataset.

![Power spectrum of the training dataset](image)

Fig. 11: Power spectrum of the *IBM* training data. The power decreases fast with increasing frequency.

We can see that only the lowest frequencies contribute significantly to the signal. Thus, it is reasonable to only use these to fit the sinus curve. Tests on the various datasets reveal that fixing k to three leads to most reasonable models. Higher values would result in too much oscillation not representing well the actual price curves. Figure 12 shows the hit rate measure of the model built for each of the datasets.
The plot shows that the model in fact can lead to good directional predictions. On the AMZB dataset the approach completely fails. This is because in this dataset the price constantly falls and the sinus model can’t imitate this behaviour, regardless of the choice of the k parameter. Figure 13 shows, as an example, the stock price together with the resulting sinus curve for the IXIC dataset.

One can see that the frequency-based approach has the potential to correctly predict the direction of price movement. We will see if we can use these models to make trading decision in the next chapter.
4. Using Predictions for Trading

In this chapter we use the models built in the previous chapters for decision making process of trading algorithms. The goal is to use the information provided by the predictions as good as possible to maximize the profit made on the test period. The developed strategies are presented in the following.

4.1. Developing Trading Strategies

*rand* (random trading): For comparison with the other predictors we first define a rule-based trading algorithm which makes decisions according to the technical indicators presented in section 2.3. The information that these rules are extracted from comes from [7]. The first trading rule is to buy if the MACD line crosses up through the signal line and to sell if it crosses down through it. A MFI of over 80 indicates that a stock is overbought and if it’s lower than 20 this title is considered oversold. Thus, we here decide to sell if the stock is overbought and vice versa. If the ROC line crosses up through zero, a change in direction from downwards to upwards is signalled. We decide to buy at this point and to sell if it crosses down through zero. The %K indicator signals a buying decision, if it crosses up through the %D line. Vice versa, we sell if is crosses down through it. For the RSI the same rule applies as for the MFI. We decide to sell if the indicator signals that the stock is overbought (values above 80) and buy in the opposite situation (values below 20). Now the five rules are evaluated for each trading day. We then calculate a decision sum for each day as follows: for a buying signal of a rule the sum is increased by one and, accordingly, a selling signal decreases the sum by one. If for a day the decision sum exceed 2 a buying decision is made. Correspondingly, a sum of less than -2 induces a selling decision.

*nul* (no trading): This strategy doesn’t trade at all. It simply buys shares with all available money in the beginning and keeps them until the end. This corresponds to the profit made from the average price movement over the test period.

*trend* (trend): Here the predictions from the last three days are considered. If they are constantly rising, we assume to be in an upward trend and make the decision to buy a certain amount of shares. If the predictions are falling over that period we decide to sell the same amount of shares.

*inv* (inverse trend): The opposite reasoning as with the trend strategy is also possible. If predictions are constantly falling, we assume that the price will soon be rising again. Thus, we make the decision to buy a certain amount of shares. Accordingly, we sell if predictions indicate an upward trend.

*tp* (turning point): This strategy looks for three successive movements in the same direction followed by a predicted movement in the opposite direction. This situation is interpreted as a turning point and, according to the direction, a trading decision is made.

*mix_tp* (mixed turning point): This method is essentially the same as the turning_point, except that we look at two successive movements of the true price followed by two predicted movement in the opposite direction.
This strategy looks at the relative size of the predicted movement for tomorrow:

\[
\text{rel} \_ \text{pred} \_ \text{move} = \frac{|\text{prediction}_{\text{tomorrow}} - \text{prediction}_{\text{today}}|}{\text{prediction}_{\text{today}}}
\]

If it exceeds a certain threshold a decision is made. Depending on the direction of movement an amount of shares are bought or sold. The thresholds for buying (\(b\) parameter) and selling decisions (\(s\) parameter) can be varied independently.

Here the same criterion for decision making is applied than with the \(p \_ b \_ s\) approach. The only difference is that the amount of shares to buy or sell is also proportional to the size of the predicted movement. If a buying decision is made (\(\text{rel} \_ \text{pred} \_ \text{move} \) exceeds \(b\) parameter) a fraction of \(\text{rel} \_ \text{pred} \_ \text{move} \times 10\) percent of the liquid assets is invested in shares.

\[
\text{buy} \_ \text{amount} = \frac{\text{liquid} \_ \text{assets} \times \text{rel} \_ \text{pred} \_ \text{move} \times 10}{\text{price}_{\text{today}}}
\]

Vice versa, if a selling decision is made a fraction of again \(\text{rel} \_ \text{pred} \_ \text{move} \times 10\) percent of the shares is sold and thus converted into liquid assets.

\[
\text{sell} \_ \text{amount} = \text{available} \_ \text{shares} \times \text{rel} \_ \text{pred} \_ \text{move} \times 10
\]

This strategy is employed for the frequency-based sinus models. Buying decisions are made at the peaks of the sinus wave and vice versa. A fixed amount of shares is traded every time.

The prices of the different datasets reside on quite different scales. Therefore all strategies, except those trading with proportional amount of shares, always buy as much as they can afford and sell everything on a selling decision. To penalize strategies that conduct too many trades a transaction fee is introduced for every trade, which is one percent of the traded volume and 50$ minimum.

### 4.2. Assessing the Methods

In the following the most promising models and strategies are chosen by assessing them on the IBM test data. As a performance measure for trading the total value of assets (TVA) is calculated. This is comprised of the current liquid assets plus the current value of the shares one possesses.

\[
\text{TVA} = \text{current} \_ \text{balance} + \text{available} \_ \text{shares} \times \text{price}_{\text{today}}
\]

Figure 14 shows the final value of this measure at the last day of the test dataset for different models using the strategies presented in the previous section.
The strategies using information about the size of the price movement clearly outperform other methods. The plot shows that the models with high hit rates from the previous chapter perform much better than those with a low average prediction error. Also, the strategies using information on the predicted size of price movement (pa_0.02_0.015 and p_0.02_0.015) are much more successful than the other, simpler ones. Figure 15 shows an example of the behaviour of the pa_0.02_0.015 strategy together with the man5_w5_s1 model.
The plot shows that the strategy reacts reasonable. It buys on upward and sells on downward trends. The only deficiency that can be noticed here is that the selling decisions are probably made too late, when the stock has already fallen for too long.

So we’ll now have a closer look at the \( b \) and \( s \) parameters responsible for decision making with the \( p_{b,s} \) and \( pa_{b,s} \) strategies. In figure 16 the influence of them is shown for the \( pa_{b,s} \) strategy by drawing three interesting measures. The model \( man5\_w5\_s2 \), which performed best in the previous experiment, is employed now. The top plot shows the original stock price together with the current TVA of the strategy with different values for the \( b \) and \( s \) parameters. The middle graph plots the percentage of liquid assets over time. The bottom plot finally shows the accumulated transaction fees.

![Figure 16: Influence of the threshold parameters on the \( pa_{b,s} \) strategy. The algorithm represented by the green line is frequently conducting trades, whereas the other ones rely on much less decisions.](image)

It is interesting to notice that the \( pa_{0.01,0.005} \) strategy accumulates a lot of transaction fees as it conducts lots of trades with low volume. It is also the worst performing strategy. All
three strategies start to buy shares more or less at the same point before the stock starts to rise.
The reason for the good performance of the other two strategies is that they convert their assets more and more into shares towards the end of the period. After testing various parameter settings two appear most promising. These are the \( pa_0.015_0.01 \) and \( pa_0.02_0.015 \) strategies, which are encoded as \( pa1 \) and \( pa2 \) in the plot in the next section. Accordingly, the \( p_0.015_0.01 \) and \( p_0.02_0.015 \) methods are named \( p1 \) and \( p2 \).

4.3. Cross-validating Trading Strategies

Finally, we want to check performance on the other datasets. It turns out that the models behave very different on the various datasets. For example do the successful setups for the \( IBM \) dataset don’t perform well on the \( DJI \) data. They completely fail to make trading decisions as predictions do not behave similar. Thus, we have to find the best model from the available ones for every dataset. For this reason the trading performance is averaged over all strategies to choose the model to be applied for a specific dataset. The final TVA after the test period is shown in figure 17 together with the \( trad \) and \( rand \) strategy for every dataset. This gives us a good indication on the usability of the developed decision making strategies. As a base line the \( null \) strategy is drawn. This represents the profit made from the average stock movement over the test period and only models performing better than average are interesting to us.

First we will look at the various dataset. For the \( IBM \) test set the proportional trading strategies (\( pa1, pa2, p1, p2 \)) perform better than the market average. The best strategy (\( p2 \)) even manages to outperform the average by a factor of 2. On the \( DJI \) data, all trading strategies fail to reach the average profit. The reason is that average profit is already very high because the stock is constantly rising. It is hard for trading strategies to additionally make profit out of this stock. On the \( IXIC \) and \( KFT \) data the mixed turning point strategy performs best making slightly more profit than the market average. The results on the \( NVS \) dataset are disappointing, as no strategy comes even close to the average performance. On the \( AMZB \) test set, which is a downward moving stock, some methods perform better than the average. This is not surprising as in this case not trading any stock would already perform quite well.

Regarding the prediction-based trading strategies, the \( mix\_tp, p\_b\_s \) and \( pa\_b\_s \) methods look promising. They manage to make better than average profit on four of the datasets. More fine-tuning of these could probably also make them work on the other datasets.

The reference approaches produce unsatisfying results. The frequency-based (\( tp\_sin \)) and the random (\( rand \)) approach fail except for the \( AMZB \) data. Here they profit from conducting few trades and thus retaining much liquid assets until the end. The results from the frequency-based method indicate that the datasets don’t follow a common frequency pattern. The indicator-based technique (\( trad \)) performs below average. This can be attributed to the rather primitive rules derived from the indicators.

Generally we can conclude that the models don’t generalize well on other datasets. The good hit rates on the \( IBM \) dataset are not reproduced on the other stocks. A possible reason is that the characteristics of the various stocks are too different and the models need to be adapted for each of them. Nevertheless, using trading strategies with different properties leads to good results even on datasets were the models don’t fit well.
Fig. 17: Overview of the performance of the best models using various trading strategies. Compared to the null strategy other approaches are successful on four of the dataset.
5. Conclusions

5.1. Results

In this work we demonstrate how to build step by step a trading algorithm for specific market data. This is done in an iterative way by feature selection (section 3.2), model selection (section 3.3) and finally developing trading strategies based on the trained models (chapter 4). The goal was to build models from the IBM dataset that are general enough to also perform well on other market data. For that reason feature and model selection were not redone for each dataset, but the models found from the IBM experiments were directly retrained and tested on the other datasets.

The results indicate that the techniques used for feature and model selection of the IBM dataset perform well on this dataset only. Partially successful was one method on the IXIC data, which contains a peak in the test set similar to the IBM dataset. This leads to the assumption that the proposed techniques only manages to beat the average market movement on upward moving stocks, which show an above average peak in the test set. On the other datasets, that don’t have such characteristics, the algorithms don’t perform well in general.

Another important point regarding the application of predictors for trading purposes is that a low prediction error doesn’t necessarily give useful information for decision making. A measure like the directional hit rate is more reliable for this purpose. Due to limitations of the toolkit the models couldn’t be optimized with respect to this performance measure (as mentioned in section 3.1). One should also notice that the information contained in the predicted prices (and their movements) can lead to more sophisticated (and successful) trading decisions, which is shown by the good performance of the \( p_b_s \) and \( pa_b_s \) strategies.

Automatic feature selection is only partially successful in our experiments. This may be due to non-exhaustive feature space searches that were limited by computational power. A clear relation between the number of features and the needed network complexity could be identified. This indicates that feature sets and models should be selected together in a coupled approach.

5.2. Further work

There are several points where further investigations can be done.

With feature selection one could try to construct new features out of the technical indicators presented in section 2.3. As these often are interpreted together or within a certain trend this knowledge could be used to functionally combined them to form new, more useful features.

As mentioned in the previous section feature and model selection should be coupled. This way the model is chosen according to the nature of a feature set instead of being selected independently. As the final goal was to find profitable trading strategies the selection of trading algorithms could also be coupled into the model finding process.

The problem of poor performance on datasets not involved in the model building process could be addressed. A cross-validation of predictors on various dataset during the feature and model selection could give an improvement here.
One could also transform the problem into a classification task by targeting only the prediction of the direction of movement of a price. This way one can employ well known classifiers like, e.g., support vector machines. Instead of making daily predictions one could then also try to forecast trends (e.g. by setting the prediction target to the directional movement over one week).

Another interesting point would be the examination of market data of other types, like commodities, exchange rates or other derivative products.

5.3. Personal Experience

The work gave me a deep insight into the difficulties of predicting a seemingly random time series. I learned a lot about the impact of feature selection on the quality of the resulting models. The problem of clever parameter search for model selection was very interesting to tackle. The relation between a model and a related trading strategy was previously unknown to me, as I assumed that having an accurate predictor is sufficient to make good trading decisions. During my work I realized that this is not necessarily the case and more considerations have to be made about how to use the information from a trained model. Concentrating on neural networks, I understood much better their handling and behaviour. I also got quite familiar with the YALE toolkit and learned about its possibilities as well as its limitations. Finally, I have a clearer picture of the field of machine learning in general after having been confronted with a real world problem for the first time.

I want to thank my supervisor Prof. Ivo Sbalzarini for his support and valuable assistance during the course of the project.
6. References


5. http://yale.sf.net/
