universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Data Mining Methods in Financial Time-Series Forecasting“

verfasst von / submitted by

Valeriya Kolesnikova

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2017

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 920
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Quantitative Economics, Management and Finance
Betreut von / Supervisor:	Ao. Univ.-Prof. Dipl.-Ing. Dr. Erhard Reschenhofer

Abstract

Contemporary financial markets provide substantial monetary incentive for many agents to engage in sophisticated trading activity. Thus, massive scientific and engineering efforts towards developing time series prediction methods have been made. As a result, stock markets are highly efficient and extracting revenue from them is problematic. However, with recent development of modern machine learning techniques and emergence of powerful and accessible hardware it became possible to analyze vast amounts of stock data and search for complex price behavioural patterns and, thus, to challenge efficiency of the current financial markets.

In this work we apply three modern machine learning methods in order to analyze stock prices using historical information from the limit order book. We demonstrate that stock prices have behavioural patterns, and it allows us to predict the direction of large price fluctuations with accuracy, which is significantly better than performance of a random model. We also discuss whether our model can be used for extracting revenue from the financial markets and what are potential obstacles for that.

(German translation)

Zeitgenössische Finanzmärkte bieten einen beträchtlichen monetären Anreiz für viele Agenten, sich an anspruchsvollen Aktienhandelsaktivitäten zu beteiligen. Daher wurden massive wissenschaftliche und technische Anstrengungen unternommen, um Zeitreihenvorhersagemethoden zu entwickeln. Infolgedessen sind die Aktienmärkte sehr effizient und es ist problematisch, daraus Einnahmen zu erzielen. Mit der letzteren Entwicklung moderner maschineller Lernmethoden und dem Aufkommen leistungsfähiger und zugänglicher Hardware wurde es jedoch möglich, eine grosse Menge von Aktiendaten zu analysieren, nach komplexen Verhaltensmustern in Preisen zu suchen, und damit die Effizienz der aktuellen Finanzmärkte in Frage zu stellen.

In dieser Arbeit wenden wir drei moderne Methoden des maschinellen Lernens an, um Aktienkurse anhand historischer Informationen aus dem Orderbuch zu analysieren. Wir zeigen, dass Aktienkurse Verhaltensmuster haben, und es erlaubt uns, die Richtung von großen Preisschwankungen mit Genauigkeit, die signifikant besser ist als die Leistung eines Zufallsmodells, vorherzusagen. Wir diskutieren auch, ob unser Modell für die Erzielung von Einnahmen aus den Finanzmärkten genutzt werden kann und welche potenziellen Hindernisse dafür bestehen.

Contents

Chapter 1

Introduction

Since the formation of modern stock markets a large number of scientists and practitioners have been working on forecasting of financial time-series. In order to make predictions numerous factors are taken into account: various historical data, currently available information regarding a stock, an industry and economics in general, natural, political and social events, investors' sentiments and behavioral effects.

There are two main classical approaches to stock market prediction: fundamental analysis and technical analysis. Fundamental analysts use stock's underlying information such as company's financial statements and earnings, industry and macroeconomic indicators and a number of specific ratios, e.g. price to earnings, to predict future of the stock. In contrast to fundamental analysts technical analysts assume that economical information is already reflected in the prices; therefore, they look into the past of the stock. They analyze technical indicators, consisting mostly of price and volume relations, build charts and look for price patterns to predict following price trends.

There are other prediction approaches which are not typically included in these two groups, which use advanced computing techniques and rely on modern machine learning. Some of these methods search for complex patterns in the data, others focus on news and social media impact on stock market. According to Mitchell ([18]) machine learning can be defined in the following way: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T , as measured by P, improves with experience E." Using this definition we can formulate the following example for the problem of stock price prediction: task T is to predict if price moves up or down, experience E consists of historical prices, performance measure P is a number of correct predictions.

In this thesis we focus on the latter group of methods. The structure of the thesis is following: first, we provide a review on the literature related to the topic of stock market prediction. In a subsequent sections we formulate the stock prediction task as a machine learning problem and then present thorough experimental evaluation. We conclude the thesis with in-depth analysis of the key results and discuss limitations of our approach.

1.1 Literature Review

There are two dominant opinions in the field of forecasting market prices. The first one is based on the believe that stock prices follow random walk paths and there is no information in the past of the stock that can explain future fluctuations. There are two related theories: random walk hypothesis, which was supported by Burton G. Malkiel [17], and efficient-market hypothesis (EMH) formulated by Eugene Fama [7]. Malkiel stated that stock prices follow a random walk and provided evidence by showing that investing in an index fund outperforms investing in an average actively managed fund. Fama formulated three forms of EMH with regard to observable information set and provided empirical evidence for each form:

- *Weak* form: price of a stock reflects past price history.
- *Semi-strong* form: price of a stock reflects all past publicly available information.
- *Strong* form: price of a stock reflects all past information.

Weak form of EMH implies that technical analysis will not be able to predict excessive returns, though fundamental analysis can work in this case. Under semi-strong form of market efficiency neither technical nor fundamental analysis can produce excessive returns. Under strong form of market efficiency noone can earn excessive returns.

A large number of research papers are devoted to testing all three kinds of EMH. One of the obstacles which researchers encounter is a proper choice of asset pricing model used to forecast future returns. Expected returns have to be compared to observed returns and if observed returns are "abnormal" it can indicate both, market inefficiency or wrong pricing model, which does not account for some factors that could lead to abnormal returns. Such problem is called joint

hypothesis problem ([7]) and it states that proving market efficiency or inefficiency is inherently problematic.

Another group of researchers hold an opinion that stock prices (or returns) are, at least partially, predictable. Certain market anomalies serve as an evidence of market inefficiency. Momentum trading, or buying stocks performing well in the past and selling stocks performing badly in the past, results in excessive returns over certain periods of time [10]. There are some seasonal anomalies, such as January effect: the small companies' stocks have larger returns in January rather than in any other month [19]. Stocks with low price per share to earnings per share ratio perform better than stocks with high price to earnings ratio [3]. A relatively new field, behavioral finance, which is based on prospect theory by Kahnemann and Tversky [11], is often opposed to EMH. It argues that stock market anomalies are results of investors' irrationality and their psychological biases [2]. Fama doubts most of the known anomalies, suggesting that short-term anomalies tend to be eliminated with time, and reports that long-term anomalies disappear with reasonable changes, applied to methodology of estimating excessive returns [8]. In contrast to Malkiel, Lo and MacKinlay rejected random walk hypothesis for weekly stock market returns [15] and focused their research on this topic. In his later work Lo presented adaptive market hypothesis, combining ideas behind EMH and behavioral finance [14].

While theoretical debates continue, researchers propose new prediction methods and report significant results every year, and most of them rely on novel computing techniques. A number of papers is dedicated to prediction of future price directions using machine learning methods. Huang, Nakamori and Wang applied support vector machines (SVM) classifier with non-linear kernel to predict direction of NIKKEI 225 index weekly returns traded on Tokyo Stock Exchange. They used S&P 500 index and Japanese Yen to US dollar exchange rate as input variables and showed that SVM outperforms other examined classifiers. Another application of SVM was done by Kim ([12]), where he used SVM with non-linear kernels and 12 technical indicators as inputs to predict direction of daily KOSPI index and demonstrated that SVM performed better than artificial neural networks.

Kumar and Thenmozhi compared SVM with random forest, artificial neural networks and few other classifiers while predicting price changes of S&P CNX NIFTY index ([13]).They used technical indicators as input variables and prediction was done for daily closing prices. The results show that SVM with non-linear kernel outperforms all examined classifiers and it performs only slightly better than random forest classifier.

Ballings et all. [1] provide an extensive survey comparing ensemble methods, including random forests, to single classifier models, such as SVM, neural networks and k-nearest neighbor, applied to direction prediction of a stock price. The dataset consists of yearly data from more than 5000 European companies, input variables are extracted from companies' financial information and the prediction was done for one year ahead. The random forest classifier is shown to perform better than other methods and is followed by SVM.

Machine learning methods are also used to analyze external sources of information, such as news or social networks, to make a prediction regarding the future of stock markets. Bollen et al. [4] analyzed sentiment of the twitter feed and with the help of neural networks found it was correlated with performance of Dow Jones Industrial Average. Schumaker and Chen [20] used current prices of stocks and news articles as inputs to SVM in order to predict the reaction of the prices twenty minutes after the news release.

Chapter 2

Machine Learning for Predicting of Future Stock Prices

2.1 Problem Statement

In this work we focus on the problem of predicting large stock price deviations, which happen within short periods of time. We will formulate this task as a machine learning problem. Our main underlying assumption is that historical information can be used to foresee large price changes before they actually happen.

In order to define a price prediction task we introduce a notion of **price event**. Price event occurs when stock price changes by more than r in a short period of time T. Formally, let P_t be price of a stock at some moment of time t. We say that **price event**, **e**, occurs at a moment t_e if $\frac{|P_{t_e+T} - P_{t_e}|}{P_{t_e}} > r$.

Every price event is associated with a discrete label $y(e) \in \{-1, +1\}$, where "-1" indicates a negative price change and "$+1$" indicates a positive price change. Moreover, we assume that a suitable feature set, $\phi(e)$, is fixed. To construct feature set we use information about the stock from the limit order book. In particular, me utilize the last H moments of time, which were recorded strictly before event e has occurred. A schematic representation of a price event and feature representation vector is depicted in Figure 2.1. We discuss the particular choices for r , T and H, as well as granularity of time discretization later in the text.

On a high level, we aim to train a model, which predicts direction of a price change $y(e)$ from the feature set $\phi(e)$, which is derived from the past information from the limit order book. Our data-driven approach can be summarized in three main steps:

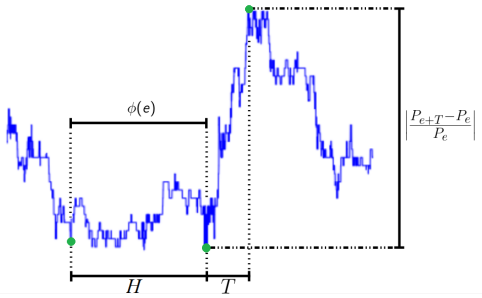


Figure 2.1: The schematic illustration of a price event. The horizontal interval marked with T represents the time period within which large price change appears. The horizontal interval marked with H represents a historical period of time, which is used for constructing a feature vector $\phi(e)$.

- Collect price events e, which form set E. Divide the set in three parts: training, validation and test sets.
- Compute events' feature representations $\phi(e)$ based on the past.
- Apply machine learning technique to fit a prediction model.
- Make a prediction using the fitted model.

This summary implies that the prediction is done on the set, containing only events. Unfortunately, such setting is impossible in real life where no information regarding the future of the stock is available. We can never definitely know if the price of the stock changes largely in the next few minutes. After we get significant prediction results on the set E, we try to make a prediction on the real data.

In a sequel of this work we present detailed explanation of the steps listed above. In particular, we evaluate performance of various machine learning techniques and discuss significance and implications of our results.

2.2 Data

The dataset we are using is extracted from LOBSTER, a limit order book tool. LOBSTER data is provided via two files: message and order book files. Figure 2.2 shows examples of the files.

The short description of the files as provided by [16]:

Message file

- Time: Seconds after midnight with decimal precision of at least milliseconds and up to nanoseconds depending on the period requested
- Event Type:

    1: Submission of a new limit order

    2: Cancellation (partial deletion of a limit order)

    3: Deletion (total deletion of a limit order)

    4: Execution of a visible limit order

    5: Execution of a hidden limit order

    7: Trading halt indicator (detailed information below)

- Order ID: Unique order reference number
- Size: Number of shares
- Price: Dollar price times 10000 (i.e. a stock price of $91.14 is given by 911400)
- Direction:

    -1: Sell limit order

    1: Buy limit order

    Note: Execution of a sell (buy) limit order corresponds to a buyer (seller) initiated trade, i.e. buy (sell) trade.

Order book file

- Ask Price 1: Level 1 ask price (best ask price)
- Ask Size 1: Level 1 ask volume (best ask volume)
- Bid Price 1: Level 1 bid price (best bid price)
- Bid Size 1: Level 1 bid volume (best bid volume)
- Ask Price 2: Level 2 ask price (second best ask price)
- Ask Size 2: Level 2 ask volume (second best ask volume)

    ...

    The term 'level' refers to occupied price levels. The difference between two levels in the LOBSTER output is not necessarily the minimum tick size.

LOBSTER

message file.

Time (sec)	Event Type	Order ID	Size	Price	Direction
⋮	⋮	⋮	⋮	⋮	⋮
34713.685155243	1	206833312	100	118600	-1
34714.133632201	3	206833312	100	118600	-1
⋮	⋮	⋮	⋮	⋮	⋮

order book file.

Ask Price 1	Ask Size 1	Bid Price 1	Bid Size 1	Ask Price 2	Ask Size 2	Bid Price 2	Bid Size 2	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1186600	9484	118500	8800	118700	22700	118400	14930	...
1186600	9384	118500	8800	118700	22700	118400	14930	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Figure 2.2: Message and order book file examples from LOBSTER website [16].

2.2.1 Data aggregation and Parameters

All experiments are done on `YHOO` stock since it has a long enough history of observations and is actively traded. Few other stocks are used to confirm consistency of the results.

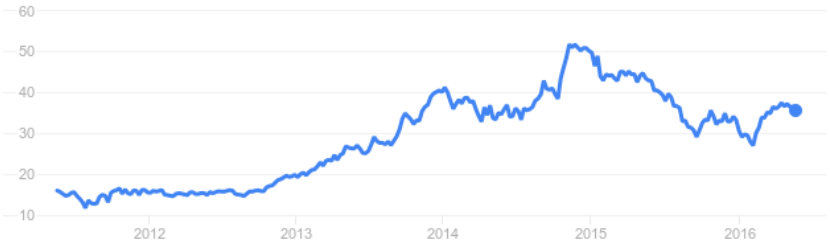


Figure 2.3: YHOO stock historical prices

The following disjoint consecutive time intervals are used:

Training set: 150 days, from 02.01.2013 to 06.08.2013

Validation set: 88 days, from 07.08.2013 to 10.12.2013

Test set: 75 days, from 11.12.2013 to 31.03.2014

Training set is used to fit parameters of the model, validation set is used to evaluate results and to choose hyperparameters of the model and test set is used to report results.

We try to solve our problem for relatively small periods of time. Initial values of the parameters are:

- $T = 2$ minutes.
- $r = 0.1\%$.
- Parameter H is chosen as a model parameter within model validation step.

We move along time series of stock prices and calculate absolute value of return for each window of size T and compare it with r. If absolute value of return is lower than r we move one second ahead and calculate absolute value of return for the new window and compare it with r again. Otherwise, we store feature vectors preceding this event and the label, $y(e)$, that corresponds to direction of the event.

We aggregate LOBSTER data by seconds averaging values within each second to reduce and to unify number of features.

Feature set, or feature representation, $\phi(e)$, then consists of feature vectors, each of length H, and can be considered as a two-dimensional vector itself.

With the specified choice of parameters T and r we can observe on average 75 price events per day and the following number of events per set:

Training set: $\sim$ 11000 events.

Validation set: $\sim$ 8000 events.

Test set: $\sim$ 7000 events

2.3 Prediction Models

The problem is a classification problem with two possible outcomes: positive or negative direction of the stock $y(e)$. We solve it with three models which use different approaches: support vector machine (SVM), random forest and k-nearest neighbors algorithm (kNN).

At first, all the models applied with a feature set containing only preceding prices. Other features are introduced in a separate section.

2.3.1 Support Vector Machine Model

SVM model is based on work of Vladimir Vapnik and Alexey Chervonenkis and was introduced by Bernhard Boser, Isabelle Guyon and Vladimir Vapnik in 1992

[5] in the current form, expanding the method to non-linear classifiers. We can consider observations we have to be points in multidimensional space where each point if defined by a vector of features $\phi(e)$. Each point also belongs to one of two classes given by $y(e)$. The linear SVM builds a hyperplane which separates different classes of points with the largest margin. The margin is a distance from the closest point of each class to the hyperplane. A graphical representation of the method in two-dimensional space is shown on Figure 2.4. Both lines separate two classes of points, but red line has the largest margin and would be built by SVM.

The problem then can be written in the following terms:

Objective:

$$\min_{w} \underbrace{\sum_{e} \max(0, 1 - y(e)\langle w, \phi(e)\rangle)}_{\text{Hinge loss}} + \underbrace{\frac{1}{2C}\|w\|^2}_{\text{Regularization}} \tag{2.1}$$

$$sign(\langle w, \phi(e)\rangle) = y(e). \tag{2.2}$$

where:

- w is a normal vector to a hyperplane. It is an outcome of the model. Once it is calculated, the sign of $\langle w, \phi(e)\rangle$ for a particular event e indicates the class of the event.
- C is a regularization constant. It shows a trade-off between the margin size and the correct fitting of the points to corresponding sides of the plane.

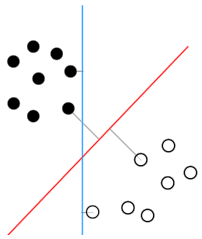


Figure 2.4: Example of SVM method in two-dimensional space in case of fully separable classes. The red hyperplane divides two classes of points (black and white) in 2-dimensional space. The blue line also separates two classes but it has smaller margin.

There are two parameters in the model:

- H, the length of feature window.
- C, the regularization constant.

We fit our model on training set for different values of the parameters and calculate the precision of the model, or the number of the correct predictions we can make with such parameters. Table 2.1 shows results on training and validation sets. The heatmap on Figure 2.5a helps to visualize how precision changes with respect to H and C on training set. It increases with both parameters, which is natural since we fit and make a prediction on the same set. After we fit the model on training set for each pair of H and C we make a prediction on validation set. The heatmap on Figure 2.5b shows a precision depending on the parameters on validation set. The area around $H = 120$ and $C = 32$ corresponds to the best results, so we check it closer with more granular values of the parameters. Heatmap on figure 2.6 shows the area at a closer range. Finally, we fix the parameters which give the best precision of $\sim 57.25\%$ on validation set:

- $H = 131$,
- $C = 44.7$.

With these values the precision on the test set is 57.8%.

Considering the test set contains 49.38% of positive price events and 50.62% of negative price events we use a goodness-of-fit test to justify the result. Since there are only two possible classes in the data, positive or negative, we can use Binomial test. Number of events in the test set is $n = 7363$, number of correct predictions is $q = 4256$. The null hypothesis is that the result we produced is insignificant, and the real probability of guessing the class is:

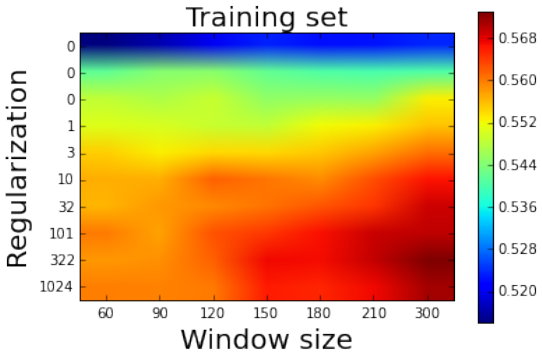
$H_0 : \ p = 0.5062.$

The test is two-sided. According to Binomial distribution the formula of probability of getting at least q successes in n trials is:

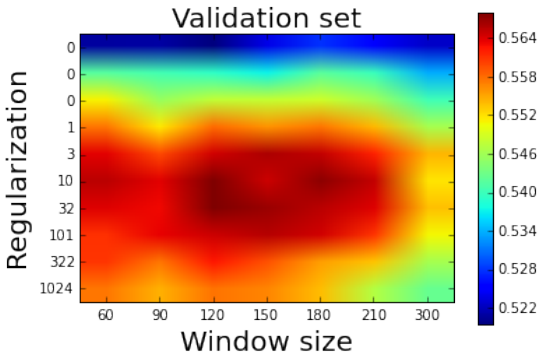
$$P(q; n, p) = \sum_{q=0}^{q} \binom{n}{q} \cdot p^q (1-p)^{n-q}$$

$$P(q; n, p) = 4.76 \cdot 10^{-35}$$

Since the calculated probability is much smaller than common significance levels of 1% or 5% we reject the null hypothesis. The result produced by SVM model is significant.



(a) Precision on training set.



(b) Precision on validation set.

Figure 2.5: Precision on training and validation sets with respect to modeling parameters H and C.

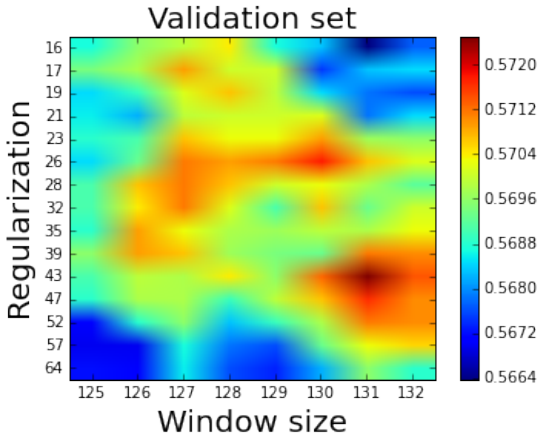


Figure 2.6: Precision on validation set with respect to granular modeling parameters H and C.

	C	0.03	0.1	0.31	1.0	3.17	10.1	32.0	101.6	322.6	1024
H											
60		51.42%	54.18%	54.86%	55.08%	55.48%	55.68%	55.64%	56.03%	55.83%	56.0%
90		51.69%	54.44%	54.73%	55.04%	55.28%	55.69%	55.83%	55.76%	55.88%	55.98%
120		52.02%	54.47%	54.92%	54.91%	55.41%	56.18%	55.92%	56.27%	56.18%	56.01%
150		52.35%	54.2%	54.51%	54.85%	55.42%	56.03%	56.07%	56.44%	56.74%	56.62%
180		52.23%	54.05%	54.56%	55.21%	55.54%	55.9%	56.24%	56.68%	56.71%	56.54%
210		52.26%	54.02%	54.53%	55.28%	55.77%	56.3%	56.45%	56.94%	56.97%	56.72%
300		52.38%	54.05%	55.27%	55.56%	56.08%	56.67%	56.92%	56.98%	57.32%	57.13%

(a) Training set

	C	0.03	0.1	0.31	1.0	3.17	10.1	32.0	101.6	322.6	1024
H											
60		52.1%	54.15%	55.11%	55.75%	56.39%	56.55%	56.4%	56.12%	56.1%	55.75%
90		52.11%	54.04%	54.53%	55.16%	55.98%	56.38%	56.32%	56.35%	55.76%	55.46%
120		51.95%	53.93%	54.78%	55.81%	56.46%	56.81%	56.81%	56.48%	56.24%	55.78%
150		52.39%	53.72%	54.8%	55.61%	56.62%	56.48%	56.69%	56.57%	55.94%	55.68%
180		52.79%	54.18%	54.86%	55.76%	56.51%	56.74%	56.56%	56.46%	55.55%	55.39%
210		52.52%	54.0%	54.59%	55.41%	56.19%	56.52%	56.4%	56.1%	55.35%	54.71%
300		52.27%	53.38%	54.03%	54.63%	55.43%	55.19%	55.37%	55.1%	54.65%	54.25%

(b) Validation set

Table 2.1: SVM model precision (results) on training and validation sets with respect to H and C.

2.3.2 Random Forest Model

Random forest model can be used for both, regression and classification problems. It was first proposed by Tin Kam Ho [9] in 1995 and was developed and generalized later by Leo Breiman [6]. The model was designed to resist to the problem of overfitting of the decision tree learning. Decision trees are known to learn too complicated patterns from training data and to overfit. As it follows from the name of the model it grows a forest consisting of decision trees. The decision is then taken based on the majority of votes among these trees for classification problem, and it is based on averaging results from the trees for regression problems. With random forest a decrease of a variance is traded for a small increase of a bias.

Each decision tree form the forest is fitted on a subset of the training set. The algorithm of generating subsets is called *Bagging (bootstrap aggregating)*. Given training set X and outputs Y, each of size n, a number of trees, or subsets, m, and a size of each subset l, a subset of size l is uniformly sampled m times with or without replacement from the set X. The term "with replacement" means that an element can be selected more than once and "without replacement" means that there can be no duplicates in a subset. The random choice of subsets gives uncorrelated trees. If all trees were built on the same set of data, even taking in account there is a randomness in building the trees, they would be correlated and would show similar results. In other words, they would be sensitive to the noise in training data, in the same way as a single tree is. Typically, the number of trees in the forest is from few hundreds to few thousands.

Another difference in building random forest's trees from building a single decision tree is that only some number of the features is used to make a split. The algorithm of selecting the features is called *random subspace method*, or *feature bagging*. It is doing basically the same as bagging, but using features instead of samples. It is possible that there are some features in the sample that can better than other features explain output variables in training set. Though, they can fail to explain the output outside training set. Choosing randomly a subset of features, we give a chance to less successful features to influence the outcome of the model. The recommended number of features to use in bagging is $\sqrt{f}$, where f is a number of features.

To sum up, there is a number of modeling parameters for this model. The most important are:

- m, a number of trees.

- l, a size of each subset.
- d, a maximal depth of each tree.
- f, a number of features.

If price history of length H is the only input variable, then we consider $f = H$ and, as recommended, we use $\sqrt{H}$ as number of features used for splitting. In our implementation of the model bagging is done with replacement and subset sizes are equal to the size of the whole training set: $l = n$. In this case all subsets are different. There are now three parameters, which we change to achieve better result: a price history size, a number of trees and a depth of trees.

Tables 2.2 and 2.3 show results on training and validation sets for different parameters H and m for depth of trees $d = 2$ and $d = 4$ correspondingly. In both cases the precision increases with respect to number of trees on training set which means the model fits better. The precision on validation set does not change much with respect to any parameter and is close to 50%. The percentage of positive events in validation set is 50.34% and the percentage of negative is 49.66%. Even without statistical test results do not look significant as it is possible to get such results by simply guessing the outcome based on the percentages of positive and negative results.

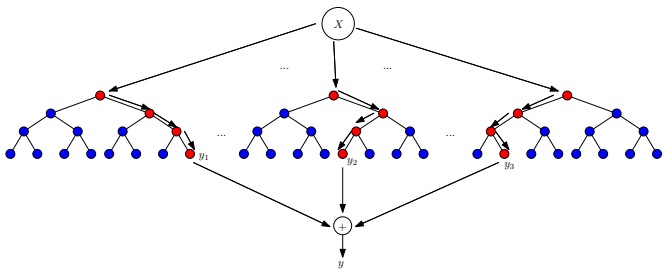


Figure 2.7: Each tree in the forest is built on a subset of training data using a subset of all features. The majority of votes(outputs) of all trees is then used as an output of the model in case of classification problem. In case of regression the average of trees' outputs is an output of the model.

	m	1	50	100	200	500	1000
H							
60		49.82%	51.14%	51.15%	51.93%	51.95%	51.51%
90		50.89%	51.94%	51.86%	52.29%	51.83%	52.36%
120		50.37%	51.73%	52.23%	51.68%	51.7%	51.5%
150		49.82%	51.58%	52.11%	51.74%	52.17,	51.95%
180		49.83%	51.55%	51.55%	52.09%	52.5%	51.5%
210		49.84%	52.22%	52.45%	51.29%	51.56%	51.77%
300		50.38%	52.47%	52.01%	52.02%	52.51%	51.48%

(a) Training set

	m	1	50	100	200	500	1000
H							
60		49.66%	49.67%	49.69%	49.67%	49.7 %	49.61%
90		50.41%	49.51%	49.42%	49.7%	49.58%	49.59%
120		50.34%	49.69%	49.6%	49.62%	49.61%	49.6%
150		50.33%	49.68%	49.74%	49.63%	49.73%	49.7%
180		50.32%	49.62%	49.7%	49.67%	49.75%	49.69%
210		49.66%	49.68%	49.64%	49.67%	49.77%	49.75%
300		50.33%	49.75%	49.76%	49.91%	49.75%	49.75%

(b) Validation set

Table 2.2: Random forest model results on training and validation sets with tree depth $d = 2$.

	m	1	50	100	200	500	1000
H							
60		50.84,	54.3%	55.12%	55.78%	55.09%	54.55%
90		50.46%	54.08%	54.97%	55.99%	55.63%	55.85%
120		50.71%	54.28%	55.05%	55.63%	55.48%	55.96%
150		50.01%	55.41%	55.57%	55.99%	56.25%	55.9%
180		50.96%	55.09%	54.98%	56.93%	56.56%	56.3%
210		51.2%	54.89%	56.88%	56.9%	55.92%	56.39%
300		50.65%	54.74%	55.45%	57.25%	56.68%	56.4%

(a) Training set

	m	1	50	100	200	500	1000
H							
60		49.74%	49.45%	49.72%	49.53%	49.38%	49.27%
90		49.67%	49.87%	49.58%	49.5%	49.6%	49.35%
120		49.67%	49.93%	49.77%	49.23%	49.45%	49.49%
150		49.67%	49.42%	49.61%	49.58%	49.45%	49.45%
180		49.66%	49.41%	49.47%	49.24%	49.56%	49.36%
210		49.73%	49.37%	49.7%	49.38%	49.43%	49.52%
300		49.69%	49.77%	49.78%	49.74%	49.44%	49.61%

(b) Validation set

Table 2.3: Random forest model results on training and validation set with tree depth $d = 4$.

2.3.3 k-nearest Neighbors Algorithm

kNN is a relatively simple method which is mainly used for classification problems.

If there is an unassigned point, which needs to be classified, the algorithm checks the classes of the closest points. It chooses a class based on the majority votes of the surrounding points. The graphical explanation of the algorithm is depicted on Figure 2.8. There are two classes: red and blue. The white circle point in the center is unassigned. If $k = 1$, then it will be assigned with the blue class; if $k = 3$, then it will be assigned with the red class; if $k = 5$, then it will be assigned with the blue class.

It is common to use Euclidean measure as a distance, however the choice of the distance measure usually depends on the problem. For instance, if feature variables are discrete, then other measure should be used.

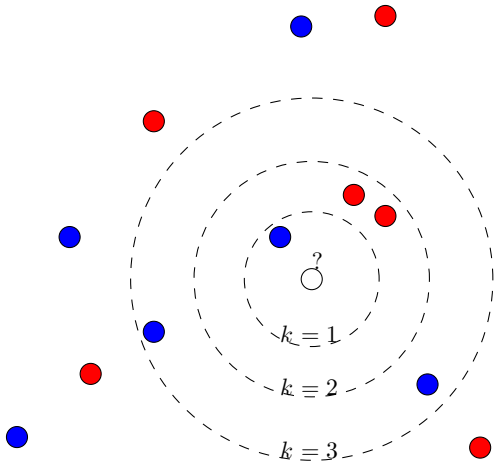


Figure 2.8: There are two classes: red and blue. The white circle point in the center is unassigned. If $k = 1$, then it will be assigned with the blue class; if $k = 3$, then it will be assigned with the red class; if $k = 5$, then it will be assigned with the blue class.

We take points from validation set and assign classes to them based on the samples from training set. We consider the distance to be Euclidean. Results are summarized in Table 2.4. The table also shows results on training set, which decrease with respect to k. It happens since the closest object to the sample is itself. Results on validation set do not change much with respect to k. They are lower than the results provided by SVM, and they fluctuate around 50%. As in case with random forest, the results are not significant taking into consideration

	k	3	7	10	20	30	100	1000
H								
60		75.57%	66.37%	62.62%	59.95%	57.87%	53.55%	51.51%
90		75.24%	66.12%	62.78%	59.39%	57.6 %	53.6%	51.48%
120		75.1 %	65.81%	61.97%	59.39%	57.51%	53.73%	51.67%
150		75.56%	65.98%	62.34%	58.79%	57.59%	54.02%	51.67%
180		74.88%	65.85%	62.5 %	58.93%	57.35%	54.22%	51.27%
210		75.38%	65.94%	62.46%	59.29%	56.82%	54.19%	51.25%
300		75.79%	65.56%	62.43%	58.65%	57.55%	54.24%	51.63%

(a) Training set

	k	3	7	10	20	30	100	1000
H								
60		49.86%	49.93%	49.91%	49.75%	49.96%	49.5 %	49.28%
90		50.03%	50.2 %	49.97%	50.03%	49.97%	49.55%	49.27%
120		50.19%	49.83%	50.12%	49.7 %	49.81%	49.62%	49.46%
150		50.05%	49.7 %	49.74%	49.27%	49.7 %	49.64%	49.57%
180		50.09%	49.62%	49.77%	49.88%	49.62%	49.59%	49.43%
210		49.95%	49.7 %	49.79%	50.02%	49.51%	49.5 %	49.45%
300		49.88%	49.45%	49.44%	49.5 %	49.55%	49.69%	49.21%

(b) Validation set

Table 2.4: kNN method results on training and validation sets with respect to k and H.

the percentage of positive and negative events in validation set.

2.4 Additional Analysis

2.4.1 Introducing Other Features

SVM model shows the best result among three models that were applied. To this point, only prices were used as features; however, there is some other information available in LOBSTER dataset, which can be used and which can possibly improve performance of the models. The following 13 features can be extracted:

- Spread,
- Number of submitted ask/bid orders,
- Volume of submitted ask/bid orders,
- Number of cancelled ask/bid orders,
- Volume of cancelled ask/bid orders,
- Number of executed ask/bid orders,
- Volume of executes ask/bid orders.

They are aggregated in a similar way as prices, by seconds, such that for a window of length H there is a vector of length H representing each feature. Combining price and each of additional features, we get 13 new feature sets. Results of running three models with new feature sets are presented in following sections.

SVM

We fix window length $H = 131$ which shows the best result in case when only a vector of preceding prices was used as an input. Figure 2.9 shows results for different values of regularization parameter C on different feature sets. The first picture shows the result for feature set containing only a vector of prices; the second shows the result for feature set containing a vector of prices and a vector of spreads; the third, a vector of prices and a vector of submitted ask orders, and so on. The blue line is a precision on training set and the green line is a precision on validation set. All graphs show worse precision on validation set if compared to the first graph. Hence, none of new features help the model to better explain the data.

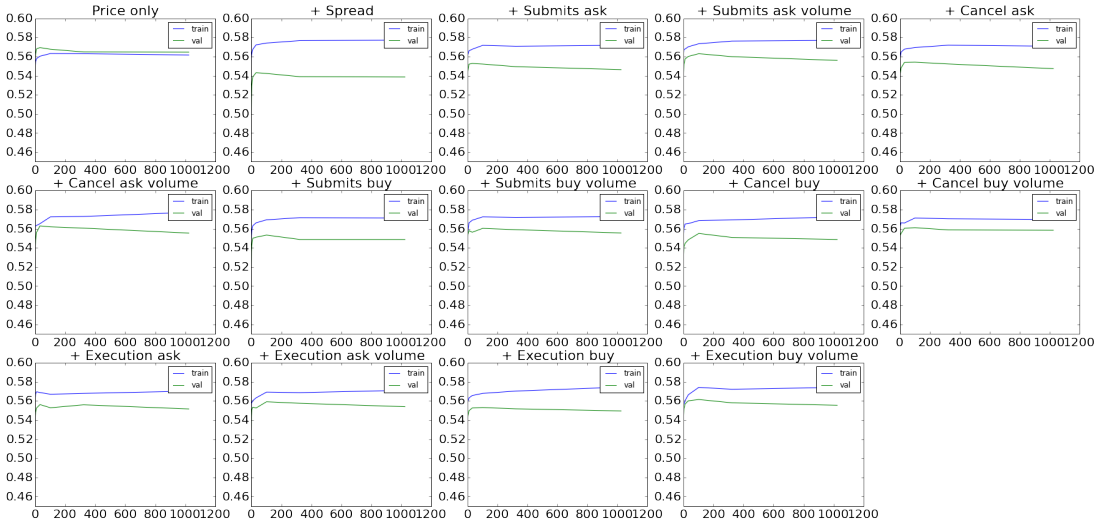


Figure 2.9: The graphs show precision of SVM on training (blue line) and validation (green line) sets with respect to regularization parameter C. The window size $H = 131$. The most top left graph shows results with feature set consisting only of price vector, the next graph to the left shows results with feature set consisting of price and spread vector, the next graph to the left shows results with feature set consisting of price and number of submitted ask orders, and so on.

Random Forest

Since Random Forest did not deliver particularly good result with any window size H, we fix the value $H = 131$ as it shows the best result in the original setting for SVM. Figure 2.10 demonstrates precision on training and validation sets with respect to number of trees, where each tree has maximum depth $d = 4$. None of the graphs show improvement in precision on validation set after new features were added to feature set.

kNN

For kNN model there is also no particularly good result in original setting with any H, so we fix the same $H = 131$. Figure 2.11 shows results on training and validation sets with respect to number of neighbors k. Again, the precision on validation set does not increase after new features added to feature set.

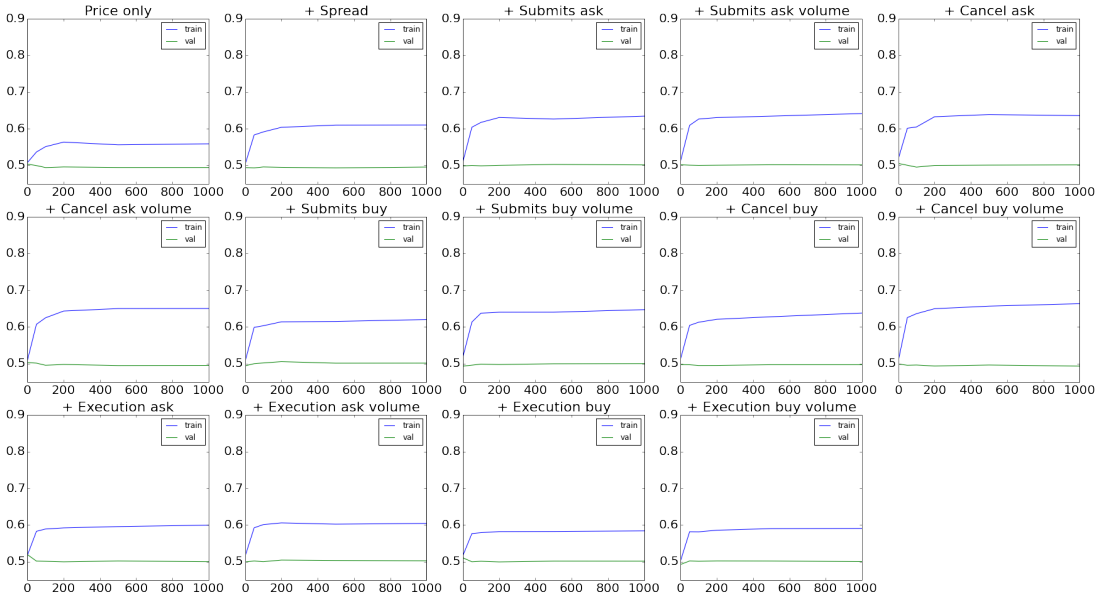


Figure 2.10: The graphs show precision of random forest method on training (blue line) and validation (green line) sets with respect to number of trees in the forest. The window size $H = 131$, maximum depth of each tree is $d = 4$. The most top left graph shows results with feature set consisting only of price vector, the next graph to the left shows results with feature set consisting of price and spread vector, and so on.

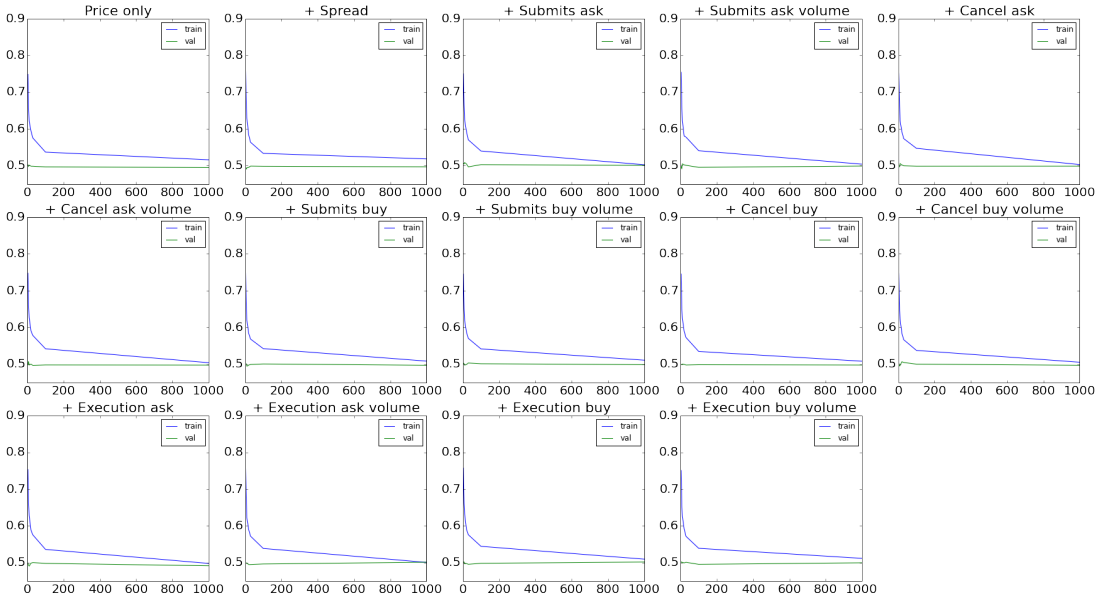


Figure 2.11: The graphs show precision of kNN method on training (blue line) and validation (green line) sets with respect to number of neighbors. The window size $H = 131$. The most top left graph shows results with feature set consisting only of price vector, the next graph to the left shows results with feature set consisting of price and spread vector, and so on.

2.4.2 Trading Bot

After fitting SVM model with $H = 131$ and $C = 44.7$ we find vector w, defining a hyperplane separating two classes of events. In the setting of the model we made a relaxing assumption that we know if a price event happens at some point, or not, but we don't know its class. We now use this assumption to create a simple trading bot which receives at each point of time information if the price event happens or not. Then it obeys following rules:

- If event will happen and $\langle w, \phi(e) \rangle) > 0$ buy stock now and sell it in 131 seconds.
- If event will happen and $\langle w, \phi(e) \rangle) < 0$ sell stock now and buy it in 131 seconds.

The result of running such bot on the test set is +669% return during 75 days with 4620 correct and 2743 incorrect predictions.

If a bot does not have information regarding the occurrence of a price event, it moves along time series and looks for the windows where it has high scores $\langle w, \phi(e) \rangle)$. Figure 2.12 shows distribution of scores. The rules are:

- If $\langle w, \phi(e) \rangle) > 0.1$ buy stock now and sell it in 131 seconds.
- If event will happen and $\langle w, \phi(e) \rangle) < -0.1$ sell stock now and buy it in 131 seconds.

The result of running such bot on test set is −35% return during 75 days with 6754 correct and 7234 incorrect predictions.

Such difference in two settings shows that the model can't make reliable predictions on real data where no future information is available.

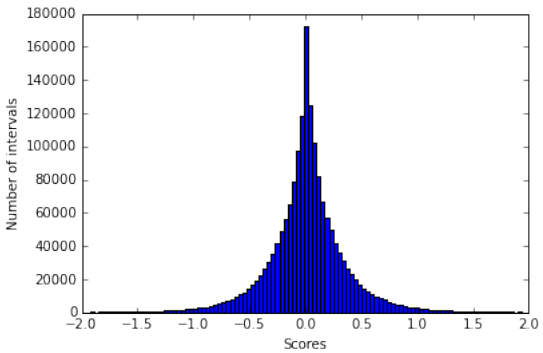


Figure 2.12: Distribution of SVM scores for all intervals of length 131 in test set.

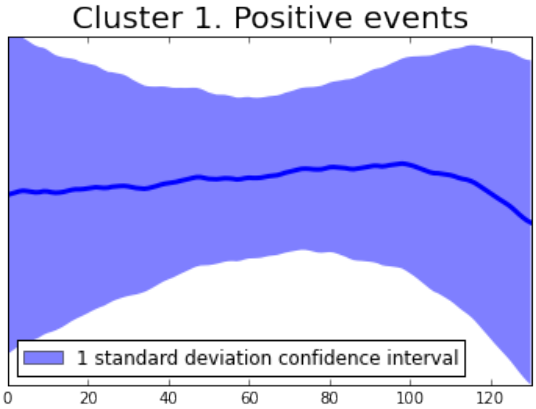
2.4.3 Interpretation of SVM Model Results

As in the previous section after fitting linear SVM model with $H = 131$ and $C = 44.7$ and computing w we move along time series and collect time intervals of length 131 where the model shows high scores. We group those windows in two clusters:

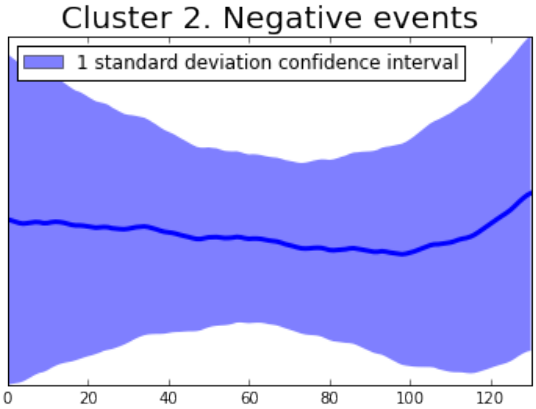
Cluster 1 contains time intervals showing score > 0.1.

Cluster 2 contains time intervals showing score < 0.1.

We then calculate mean and standard deviation of prices in time intervals in both clusters. The results are shown on Figure 2.13. The cluster containing time intervals with positive scores predicts positive events. The average path is depicted by the mean, and it slowly increases in the beginning and declines relatively fast closer to the end of the time interval. In contrast, cluster containing time intervals with negative scores predicts negative events. The cluster's average price path slowly decreases in the beginning and increases closer to the end of the time interval. Such behavior of the average price path of both clusters resembles mean reversion, a concept, and also a trading strategy, suggesting that a price reverts to its mean value over time. The means of the clusters can serve as a representation of the signals that the model learned and which it uses to make its predictions.



(a) Mean and standard deviation confidence interval of time intervals with positive score > 0.1.



(b) Mean and standard deviation confidence interval of time intervals with negative score < -0.1.

Figure 2.13: Two clusters formed by time intervals with absolute score > 0.1.

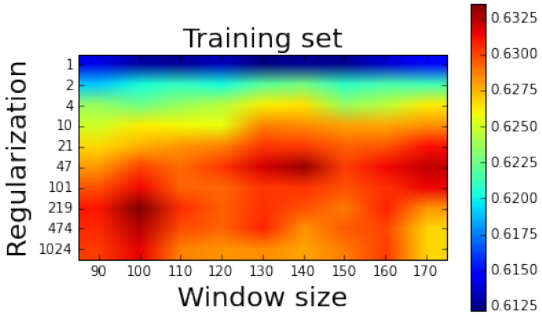
2.4.4 Other Stocks

To justify the findings we run the same models on two other datasets represented by stocks `AAPL` and `EBAY`. Same dates are used to split datasets into three parts: training, validation and test sets.

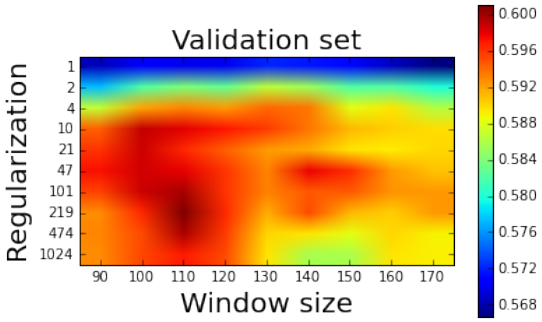
`AAPL`, SVM

Figure 2.14 shows precision heatmaps on training and validation sets with respect to the regularization parameter C and the history window H. The area around $H = 110$ and $C = 200$ shows the best result on validation set. Exploring the area closer, we get the best precision of 60.18% on validation set and 60.86% on test set with parameters:

- $H = 111$,
- $C = 173$.



(a) Precision on training set.



(b) Precision on validation set.

Figure 2.14: `AAPL` stock precision on training and validation sets with respect to modeling parameters H and C.

Test set contains 48.77% of positive price events and 51.23% of negative price events. Number of events in test set is $n = 4134$, number of correct predictions

is $q = 2516$. Binomial test is applied with the null hypothesis that the result we produced is insignificant, and the real probability of guessing the class is:

$$H_0 : \ p = 0.5062.$$

The probability of getting at least q successes in n trials is:

$$P(q; n, p) = 1.64 \cdot 10^{-35}$$

We reject the null hypothesis and consider the result of the method to be significant.

We check if other features can improve results for the window size $H = 111$. Figure 2.15 shows precision of the model with different feature sets with respect to the regularization parameter C on training and validation sets. The precision on validation set does not change if other features added to feature set.

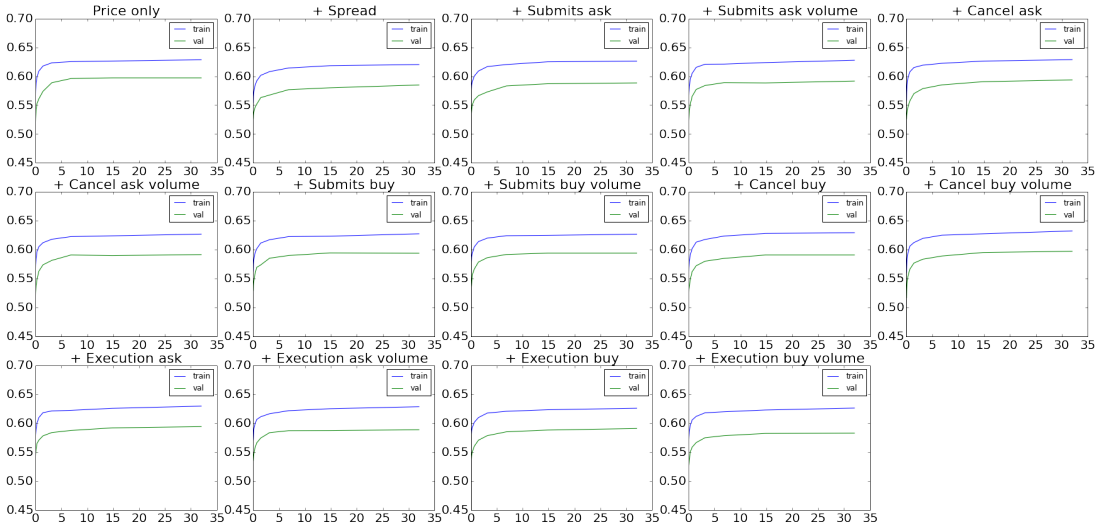


Figure 2.15: The graphs show precision of linear SVM on training (blue line) and validation (green line) sets of the stock `AAPL` with respect to the regularization parameter C. The window size $H = 111$. The most top left graph shows results with feature set consisting only of price vector, the next graph to the left shows results with feature set consisting of price and spread vector, and so on.

`AAPL`, Random Forest

Table 2.5 and Table 2.6 show precisions on training and validation sets with respect to history window H, number of trees in the forest m, and for depth of trees $d = 2$ and $d = 4$ respectively. The percentage of positive events in validation

set is 50.33%. As in the case with YHOO, results are not substantial on validation set.

Figure 2.16 shows precision of random forest model with additional features for the window $H = 111$ and maximal depth of the trees $d = 4$ with respect to number of trees m. And again, none of additional features improves the results.

	m	1	50	100	200	500	1000
H							
60		50.98%	51.65%	52.05%	52.14%	51.9%	52.34%
90		51.31%	52.06%	52.09%	52.65%	52.7%	52.49%
120		50.98%	52.6%	52.83%	52.42%	52.51%	52.4%
150		50.92%	52.19%	52.23%	52.14%	52.31%	51.95%
180		51.4%	51.94%	52.12%	52.13%	51.89%	51.96%
210		50.98%	52.05%	52.12%	51.94%	52.23%	51.89%
300		50.57%	51.44%	52.0%	52.08%	51.77%	51.64%

(a) Training set

	m	1	50	100	200	500	1000
H							
60		50.33%	50.21%	50.19%	50.26%	50.19%	50.17%
90		50.37%	50.4%	50.12%	50.22%	50.26%	50.3%
120		50.12%	50.64%	50.22%	50.17%	50.33%	50.3%
150		50.34%	50.54%	50.27%	50.21%	50.43%	50.29%
180		50.32%	50.25%	50.23%	50.3%	50.3%	50.29%
210		50.41%	50.32%	50.45%	50.36%	50.32%	50.34%
300		50.42%	50.39%	50.31%	50.31%	50.31%	50.28%

(b) Validation set

Table 2.5: AAPL random forest model precision on training and validation sets with tree depth $d = 2$.

AAPL, kNN

The results for kNN model are provided by Table 2.7. The precision on validation set is close to 50% and percentage of positive events in the validation set is 50.33%, so the results of the model are not substantial.

Figure 2.17 shows precision of the model with $H = 111$. Results do not improve with introduction of additional features.

	m	1	50	100	200	500	1000
H							
60		51.67%	55.56%	55.7%	56.55%	55.88%	57.36%
90		51.43%	56.38%	55.44%	56.4%	56.69%	56.74%
120		51.04%	55.34%	56.13%	56.5%	57.68%	57.38%
150		51.28%	55.48%	56.01%	57.24%	57.53%	56.76%
180		51.37%	54.87%	56.53%	57.2%	56.86%	58.14%
210		50.99%	55.43%	56.45%	56.28%	57.04%	57.14%
300		51.82%	56.2%	56.41%	56.55%	57.82%	57.31%

(a) Training set

	m	1	50	100	200	500	1000
H							
60		50.33%	50.39%	50.62%	49.99%	50.06%	50.28%
90		50.4%	50.62%	50.26%	50.49%	50.04%	50.17%
120		50.51%	50.15%	50.78%	50.56%	50.19%	50.26%
150		50.34%	51.29%	50.18%	50.3%	50.11%	50.25%
180		50.04%	50.3%	50.61%	50.09%	50.09%	50.16%
210		50.47%	51.06%	50.09%	50.66%	50.29%	50.16%
300		50.33%	50.3%	50.26%	50.33%	50.12%	50.33%

(b) Validation set

Table 2.6: `AAPL` random forest model precision on training and validation sets with tree depth $d = 4$.

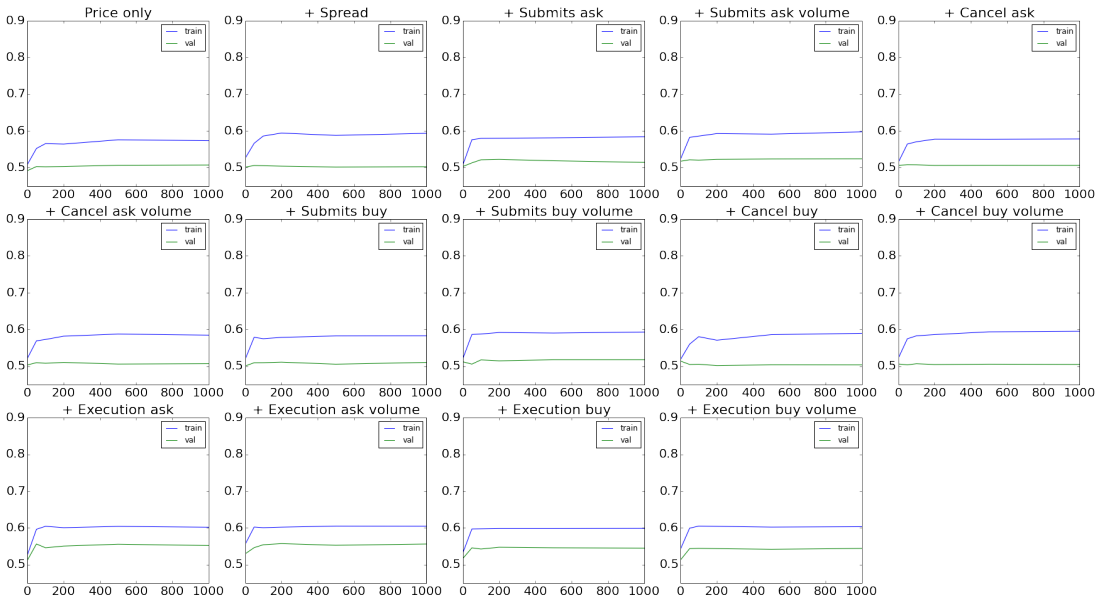


Figure 2.16: The graphs show precision of random forest on training (blue line) and validation (green line) sets of the stock `AAPL` with respect to number of trees. The window size $H = 111$ and maximal depth of trees $d = 4$. The most top left graph shows results with feature set consisting only of price vector, the next graph to the left shows results with feature set consisting of price and spread vector, and so on.

	k	1	3	7	10	20	30	100	1000
H									
60		100%	75.91%	66.97%	63.25%	59.97%	58.66%	54.55%	50.72%
90		100%	75.77%	66.62%	63.52%	59.37%	58.2%	54.72%	50.84%
120		100%	75.61%	66.2%	63.35%	59.49%	58.0%	54.41%	51.11%
150		100%	75.99%	66.36%	63.17%	59.31%	57.7%	54.65%	51.17%
180		100%	75.72%	66.08%	62.98%	59.55%	57.93%	54.89%	51.21%
210		100%	75.4%	66.12%	62.56%	59.45%	57.65%	54.52%	51.24%
300		100%	74.39%	65.06%	61.76%	58.44%	57.23%	54.1%	51.59%

(a) Training set

	k	1	3	7	10	20	30	100	1000
H									
60		51.14%	50.22%	51.12%	50.92%	51.51%	50.69%	51.23%	49.1%
90		49.72%	49.87%	50.92%	50.08%	51.05%	50.76%	51.42%	49.13%
120		50.1%	50.44%	51.03%	50.71%	51.1%	50.81%	51.19%	49.19%
150		50.84%	50.73%	50.82%	50.84%	50.79%	51.04%	51.04%	49.09%
180		50.04%	50.34%	51.45%	51.0%	50.47%	50.82%	51.67%	49.1%
210		50.5%	50.45%	51.15%	50.56%	50.68%	51.09%	50.95%	49.19%
300		50.3%	51.41%	51.32%	51.32%	50.73%	51.01%	50.76%	49.17%

(b) Validation set

Table 2.7: `AAPL` kNN model precision on training and validation sets with respect to H and k.

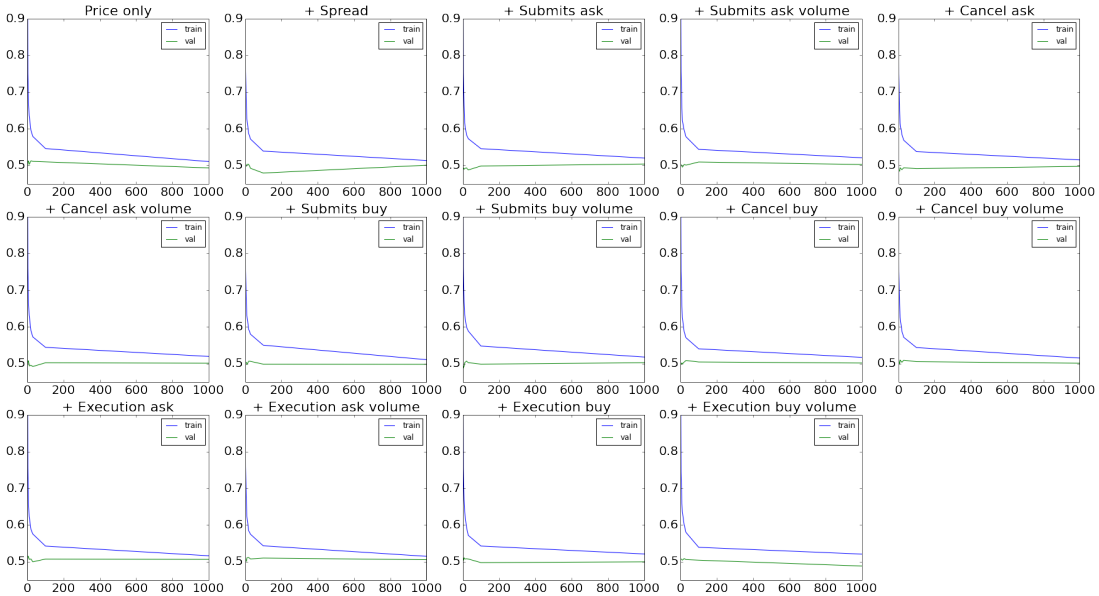


Figure 2.17: The graphs show precision of kNN on training (blue line) and validation (green line) sets with respect to number of neighbors k. The window size $H = 111$. The most top left graph shows results with feature set consisting only of price vector, the next graph to the left shows results with feature set consisting of price and spread vector, and so on.

EBAY, SVM

Figure 2.18 shows precision heatmaps on training and validation sets with respect to the regularization parameter C and the history window H. The area around $H = 120$ and $C = 10$ shows the best result. Further exploration of the area gives the best precision on validation set of 60.28% and precision on test set of 57.86% with the parameters:

- $H = 119$,
- $C = 24$.

The test set contains 50.69% of positive price events and 49.31% of negative price events. Number of events in the test set is $n = 5693$, number of correct predictions is $q = 3276$. Binomial test is applied with the null hypothesis that the result we produced is insignificant, and the real probability of guessing the class is:

$$H_0 : \ p = 0.5069.$$

The probability of getting at least q successes in n trials is:

$$P(q; n, p) = 1.45 \cdot 10^{-35}$$

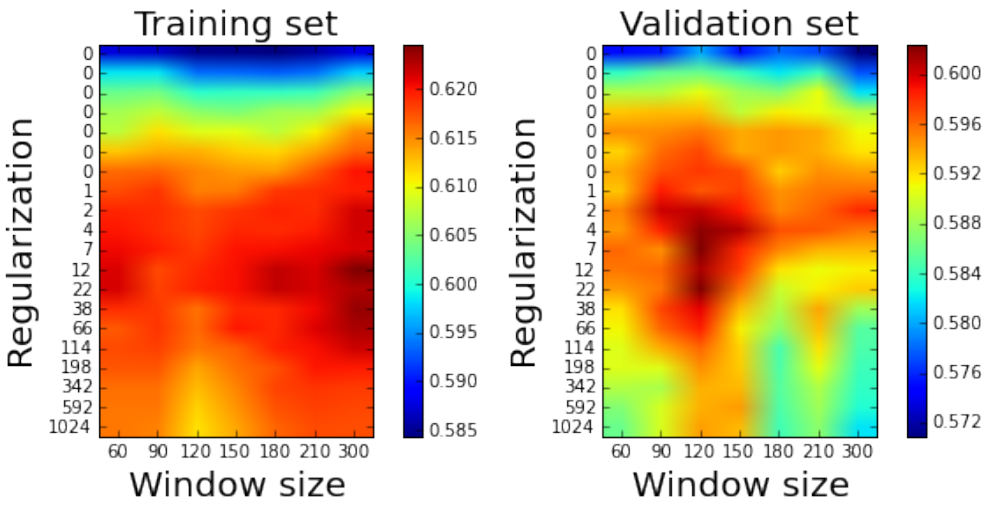
We reject the null hypothesis and consider the result of the method to be significant.

Figure 2.19 shows that none of additional features improves results on the validation set. The precision is shown for the parameter $H = 119$ with respect to the regularization parameter C.

EBAY, Random Forest

Tables 2.8 and 2.9 show precision on training and validation sets with respect to the history window H, the number of trees in the forest m, and for the maximal depth of trees $d = 2$ and $d = 4$ respectively. The percentage of positive events in the validation set is 49.35%. The results on validation set are not substantial.

Figure 2.16 shows precision with additional features for the window $H = 119$ and the maximal depth of trees $d = 4$ with respect to number of the trees. Additional features do not improve results.



(a) Precision on training set. (b) Precision on validation set.

Figure 2.18: `EBAY` stock precision on training and validation set with respect to modeling parameters H and C.

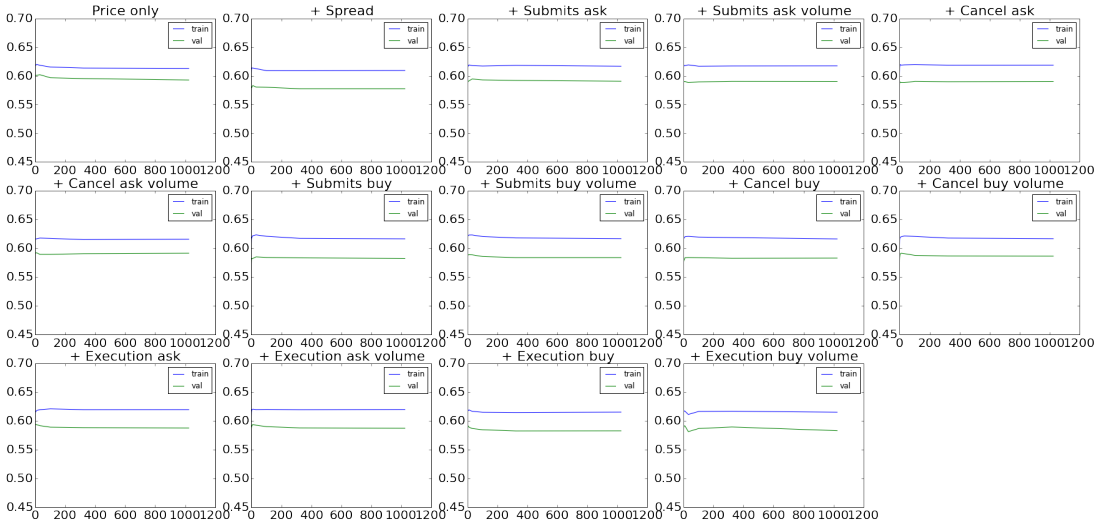


Figure 2.19: The graphs show precision of linear SVM on training (blue line) and validation (green line) sets of stock `EBAY` with respect to the regularization parameter C. The window size $H = 119$. The most top left graph shows results with feature set consisting only of price vector, the next graph to the left shows results with feature set consisting of price and spread vector, and so on.

`EBAY`, kNN

Table 2.10 provides results for kNN model. The precision on validation set is close to 50% and the percentage of the positive events in validation set is 49.35%.

Figure 2.21 shows precision of the model with different feature sets and the window size is $H = 119$. Additional features do not improve results.

	m	1	50	100	200	500	1000
H							
60		50.31%	52.76%	52.98%	53.55%	52.51%	52.63%
90		50.48%	52.58%	52.8 %	53.09%	53.5 %	52.39%
120		50.58%	52.6 %	52.64%	52.97%	52.46%	52.56%
150		50.5 %	52.08%	52.77%	52.28%	52.06%	52.39%
180		50.45%	52.42%	52.7 %	52.71%	52.32%	52.24%
210		50.74%	52.89%	52.15%	52.26%	52.77%	52.09%
300		50.67%	52.85%	52.53%	52.5 %	52.76%	52.36%

(a) Training set

	m	1	50	100	200	500	1000
H							
60		50.69%	50.1 %	51.11%	50.67%	51.28%	51.12%
90		50.78%	51.03%	50.35%	51.17%	51.18%	51.11%
120		49.43%	50.24%	50.84%	50.39%	51.22%	51.25%
150		50.53%	51.21%	51.3 %	51.1 %	51.04%	51.25%
180		49.43%	50.73%	51.27%	51.21%	51.08%	51.21%
210		50.82%	50.76%	51.32%	51.02%	50.81%	51.02%
300		50.77%	50.8 %	50.78%	50.81%	51.12%	51.15%

(b) Validation set

Table 2.8: `EBAY` random forest model precision on training and validation sets with depth of trees $d = 2$.

	m	1	50	100	200	500	1000
H							
60		50.98%	55.46%	57.66%	57.78%	58.21%	59.48%
90		51.08%	55.41%	57.11%	58.9%	58.29%	58.6%
120		50.94%	55.28%	57.39%	58.38%	59.65%	60.09%
150		50.76%	55.4 %	56.52%	59.82%	58.3 %	58.2%
180		51.23%	54.44%	57.48%	58.41%	58.38%	59.55%
210		50.86%	56.18%	57.68%	59.63%	57.99%	59.45%
300		50.73%	56.1 %	56.34%	57.62%	60.4 %	59.07%

(a) Training set

	m	1	50	100	200	500	1000
H							
60		50.92%	51.76%	51.39%	51.4%	51.7 %	51.39%
90		50.78%	51.2%	50.84%	51.63%	51.59%	51.42%
120		50.66%	49.9%	50.92%	51.2%	51.57%	51.59%
150		49.49%	50.34%	51.13%	50.79%	51.3 %	51.52%
180		49.57%	50.77%	51.94%	50.98%	51.22%	51.07%
210		49.35%	51.73%	50.88%	51.38%	50.85%	50.99%
300		49.39%	51.11%	50.1 %	50.75%	50.74%	50.91%

(b) Validation set

Table 2.9: `EBAY` random forest model precision on training and validation sets with depth of trees $d = 4$.

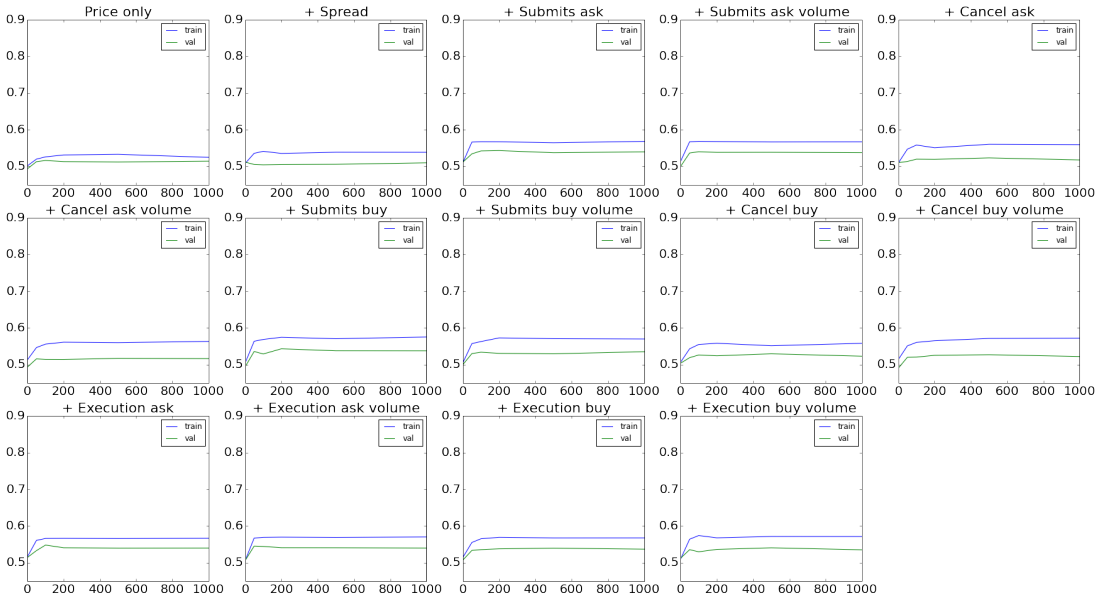


Figure 2.20: The graphs show precision of random forest model on training (blue line) and validation (green line) sets of stock `EBAY` with respect to the number of trees. The window size $H = 111$ and the maximal depth of trees $d = 4$. The most top left graph shows results with feature set consisting only of price vector, the next graph to the left shows results with feature set consisting of price and spread vector, and so on.

	k	1	3	7	10	20	30	100	1000
H									
60		99.98%	75.89%	66.91%	64.46%	60.66%	58.92%	55.56%	50.96%
90		99.98%	75.79%	66.81%	63.8 %	60.38%	58.33%	54.21%	50.86%
120		99.98%	75.91%	66.44%	63.01%	60.27%	58.57%	53.76%	50.78%
150		99.98%	75.19%	66.45%	63.28%	59.82%	57.97%	53.72%	50.91%
180		99.98%	75.02%	66.12%	63.54%	59.06%	57.26%	53.74%	51.01%
210		99.98%	75.3 %	66.5 %	63.08%	59.3 %	57.76%	53.56%	50.87%
300		99.98%	75.67%	65.64%	62.81%	59.3 %	57.45%	53.33%	50.98%

(a) Training set

	k	1	3	7	10	20	30	100	1000
H									
60		52.1%	51.85%	52.97%	52.86%	51.73%	51.88%	51.51%	49.76%
90		51.84%	51.57%	51.03%	51.46%	50.78%	51.42%	51.05%	50.15%
120		51.29%	51.49%	51.42%	51.76%	51.74%	50.92%	50.46%	50.09%
150		50.98%	51.05%	50.62%	50.48%	50.65%	50.88%	51.05%	50.39%
180		51.13%	51.61%	50.88%	51.25%	50.74%	51.07%	50.9 %	50.25%
210		49.8 %	50.29%	50.99%	50.85%	50.57%	51.12%	50.28%	50.31%
300		50.26%	49.85%	49.9 %	50.05%	50.57%	51.01%	50.47%	50.13%

(b) Validation set

Table 2.10: `EBAY` kNN model precision on training and validation sets with respect to H and k.

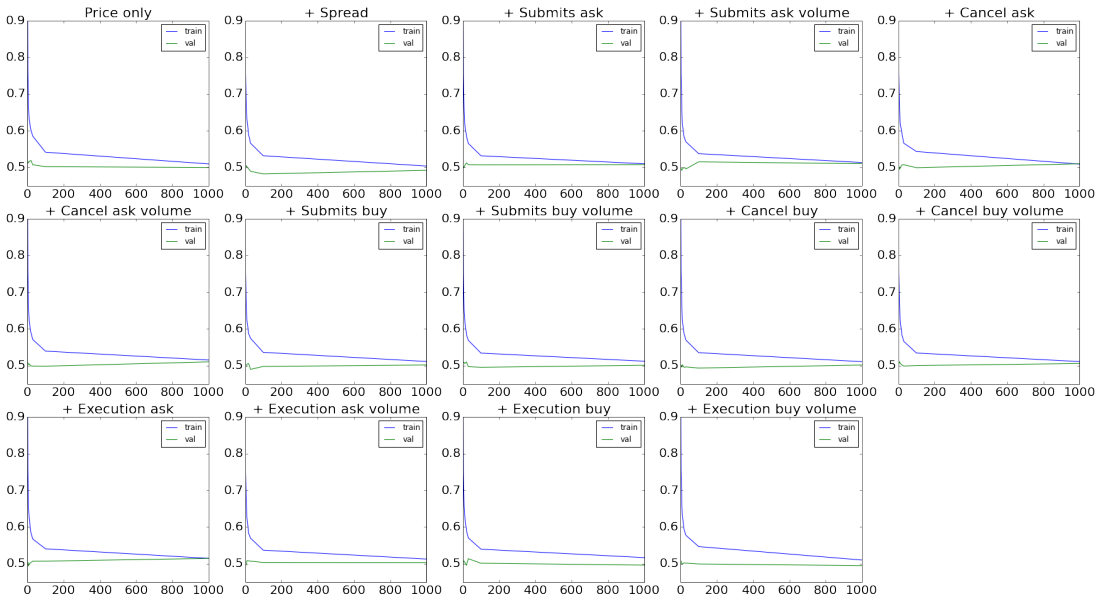


Figure 2.21: The graphs show precision of kNN on training (blue line) and validation (green line) sets of the stock `EBAY` with respect to the number of neighbors k. The window size $H = 111$. The most top left graph shows results with feature set consisting only of price vector, the next graph to the left shows results with feature set consisting of price and spread vector, and so on.

2.5 Conclusion

Three different machine learning models were applied to predict if price of a stock is going to increase or decrease in the following two minutes. The main assumption of this problem is that it is known that the following price change is relatively large and that it is definitely going to happen in the next two minutes. Only one of the proposed models, linear SVM, shows significant prediction precision on the test set. This model, however, fails to predict a direction when it does not have information if a large price change is going to happen. The model uses only past prices as input variables and it does not deliver better results with additional variables, e.g. number or volume of submitted ask/bid orders. The signal which the model learns from the past prices resembles mean reversion behavior.

To improve results one other features can be added to the model. Potentially relations or ratios between some historical indicators can be meaningful features. To reduce number of possible combinations of features one can use technical analysis indicators.

References

[1] Michel Ballings et al. “Evaluating Multiple Classifiers for Stock Price Direction Prediction”. In: *Expert Syst. Appl.* 42.20 (Nov. 2015), pp. 7046–7056.

[2] Nicholas Barberis, Ming Huang, and Tano Santos. “Prospect Theory and Asset Prices”. In: *The Quarterly Journal of Economics* 116.1 (2001), pp. 1–53.

[3] S Basu. “Investment Performance of Common Stocks in Relation to Their Price-Earnings Ratios: A Test of the Efficient Market Hypothesis”. In: *Journal of Finance* 32.3 (1977), pp. 663–82.

[4] Johan Bollen, Huina Mao, and Xiaojun Zeng. “Twitter mood predicts the stock market”. In: *Journal of Computational Science* 2.1 (2011), pp. 1–8.

[5] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. “A Training Algorithm for Optimal Margin Classifiers”. In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT ’92. Pittsburgh, Pennsylvania, USA: ACM, 1992, pp. 144–152.

[6] Leo Breiman. “Random Forests”. In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32.

[7] Eugene Fama. “Efficient Capital Markets: A Review of Theory and Empirical Work”. In: *Journal of Finance* 25.2 (1970), pp. 383–417.

[8] Eugene F. Fama. “Market efficiency, long-term returns, and behavioral finance”. In: *Journal of Financial Economics* 49.3 (1998), pp. 283–306.

[9] Tin Kam Ho. “The Random Subspace Method for Constructing Decision Forests”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 20.8 (Aug. 1998), pp. 832–844.

[10] Narasimhan Jegadeesh and Sheridan Titman. “Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency”. In: *Journal of Finance* 48.1 (1993), pp. 65–91.

[11] Daniel Kahneman and Amos Tversky. “Prospect Theory: An Analysis of Decision under Risk”. In: *Econometrica* 47.2 (1979), pp. 263–91.

[12] Kyoung-jae Kim. “Financial time series forecasting using support vector machines”. In: *Neurocomputing* 55.1 (2003). Support Vector Machines, pp. 307–319.

[13] Manish Kumar and M Thenmozhi. “Forecasting stock index movement: A comparison of support vector machines and random forest”. In: (2006).

[14] Andrew W. Lo. *The Adaptive Markets Hypothesis: Market Efficiency from an Evolutionary Perspective*. 2004.

[15] Andrew W. Lo and A. Craig MacKinlay. *Stock Market Prices Do Not Follow Random Walks: Evidence From a Simple Specification Test*. Working Paper 2168. National Bureau of Economic Research, Feb. 1987.

[16] *LOBSTER: Limit Order Book System*. URL: `https://lobsterdata.com/`.

[17] Burton Gordon Malkiel. *A Random Walk Down Wall Street [By] Burton G. Malkiel*. Norton, 1973.

[18] Thomas M. Mitchell. *Machine Learning*. 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

[19] Michael S. Rozeff and William Kinney. “Capital market seasonality: The case of stock returns”. In: *Journal of Financial Economics* 3.4 (1976), pp. 379–402.

[20] Robert P. Schumaker and Hsinchun Chen. “Textual Analysis of Stock Market Prediction Using Breaking Financial News: The AZFin Text System”. In: *ACM Trans. Inf. Syst.* 27.2 (Mar. 2009), 12:1–12:19. ISSN: 1046-8188.