A Random Walk On Wall Street

Predicting Stocks Using Machine Learning and Time Series Models

Project Report

Group: Nikos Bosse and Felix Süttmann

Georg-August-University Göttingen
Statistisches Praktikum

GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Faculty of Business and Economics
Georg-August-University Göttingen
https://www.uni-goettingen.de/

Title:
A Random Walk On Wall Street

Theme:
Predicting Stocks Using Machine Learning
and Time Series Models

Project Period:
Summer Semester 2019

Project Group:
-

Participant(s):
Nikos Bosse
Felix Süttmann

Supervisor(s):
Prof. Thomas Kneib
Jan Röder

Copies: 1

Page Numbers: 52

Date of Completion:
September 15, 2019

Contents

1. Introduction

The internet is full with recipes, videos and forum posts promising easy ways to forecast future stocks. Articles on medium.com promise to teach the reader the secrets to successfully predicting future stocks with the help of time series, AI and machine learning. On the other side are people who believe in the efficient market hypothesis and laugh about the attempt to ever beat the market. If the efficient market hypothesis is true, all known information is almost immediately priced in and what movement remains is completely unpredictable. On the other hand, others regard the mere existence and the size of the finance industry as evidence that some people are indeed better than others in predicting the market. In this report we will try to use statistical analysis to predict movements in stock prices. We are ultimately expecting to fail. If it were so simple to successfully and reliably predict stock prices given only a couple of months time - the world would indeed be a very different place. We are therefore in the following exploring different prediction methods and trading strategies mainly with the aim of learning about their features, possibilities and limitations. In Chapter 2 we will explore the data that forms the basis of our analysis: stock market data, analyst reports and externally provided news sentiment scores. Chapter 3 will use a machine learning approach to predict stock movement. In Chapter 4 we will present and apply various time series models to the stock data. Chapter 5 will detail a number of commonly applied trading strategies and compare their effectiveness. Lastly Chanpter 6 will shortly summarize our findings.

2. The Data

The following chapter gives an overview over the data on which we based our analysis. The firms we selected will be introduced as well as the Ravenpack Sentiment Data and the Analyst Reports. Lastly the stock data will be explored in detail and appropriate transformations of the data will be motivated.

2.1 Stock Data

This section presents the stock data and the necessary transformations applied to it. The data comprises financial data for 10 selected companies from the NASDAQ stock index. The choice was determined by our supervisor Jan Röder as those were the companies for which he could provide us with additional external data. Table 2.1 shows the companies and their associated stock tickers. Note that those are very large and well known companies that are traded very frequently. We therefore should expect the markets for those stocks to work very efficiently, leaving little (if any) room for the kind of market inefficiencies we would like to exploit.

Companies Analyzed

	companyname	ticker
1	3M Co	MMM
2	American Express Co	AXP
3	General Electric Co	GE
4	Intel Corp	INTC
5	Johnson & Johnson	JNJ
6	Procter & Gamble Co	PG
7	United Technologies Corp	UTX
8	Verizon Communications Inc	VZ
9	Visa Inc	V
10	Walt Disney Co	DIS

Table 2.1: List of company name and ticker of the companies we analyzed for this report.

The stock prices were downloaded from Yahoo Finance (finance.yahoo.com). Table 2.2 provides a small representative overview over the raw financial data provided. Figure 2.1 shows the adjusted closing prices ('Adj Close', adjusted for e.g. dividends) of all assets.
Time series analysis usually assumes that data are at least weakly stationary. Weak stationarity implies that the mean of the time series is constant over time and that the covariance between two observations y_t and y_{t+h} depends only on h, not on t (see e.g. Shumway and Stoffer (2011)). From Figure 2.1 it can be clearly seen that most of the stocks exhibit a strong trend. Also the variance of most stocks increases steadily with time over the observed period (illustrated in Figure 2.2 where the standard deviation of the time series is shown). The data are therefore clearly not stationary.

Stock Data

Date	Open	High	Low	Close	Adj Close	Volume	ticker
2012-01-03	83.76	84.44	83.36	83.49	68.41	3380100	MMM
2012-01-04	83.13	84.26	83.11	84.18	68.98	3007400	MMM
2012-01-05	83.53	83.87	82.70	83.80	68.67	3116400	MMM
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2015-11-27	29.11	29.21	29.03	29.19	25.97	34469600	GE
2015-11-30	29.16	29.28	28.79	28.79	25.61	82905200	GE
2015-12-01	28.84	29.09	28.72	29.01	25.80	56414600	GE
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2017-12-27	108.42	108.55	107.46	107.64	105.31	5624000	DIS
2017-12-28	108.00	108.05	107.06	107.77	105.43	3477700	DIS
2017-12-29	108.05	108.34	107.51	107.51	105.18	4538400	DIS

Table 2.2

Stock Prices

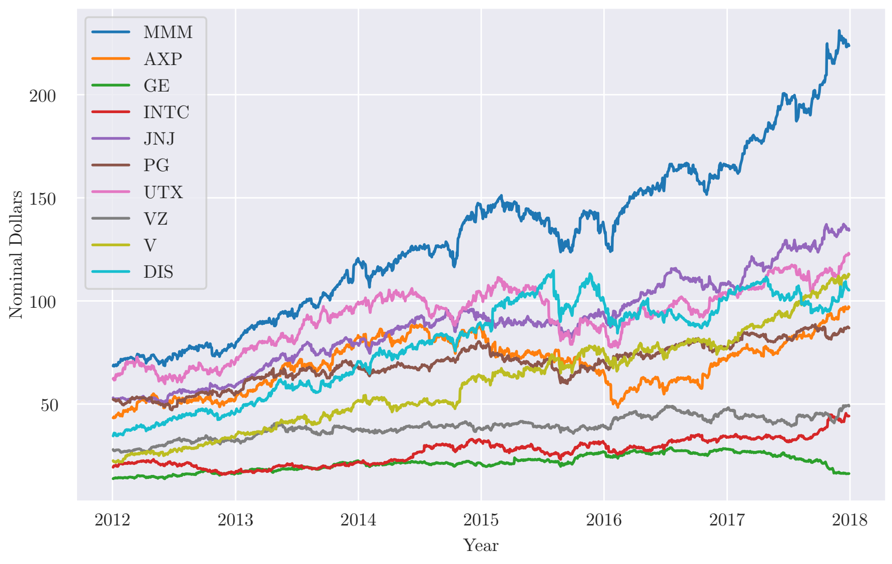


Figure 2.1: Time series of the adjusted closing prices of all 10 stocks looked at in this paper

2.1.1 Data Transformation

In order to obtain weakly stationary time series the data needs to be transformed. To ensure stationarity, economists usually work with either returns or log-returns, albeit the nomenclature is sometimes used confusingly. Daily returns can be calculated as

$$r_t^{(1)} = \frac{r_t}{r_{t-1}} \qquad \text{or as} \qquad r_t^{(2)} = \frac{r_t - r_{t-1}}{r_{t-1}} = r_t^{(1)} - 1.$$

Usually economists use $r_t^{(2)}$ and call them either 'returns' or more commonly 'log-returns', even though no logging takes place. in this report we will call $r_t^{(1)}$ 'returns' and $\log r_t^{(1)}$ 'log-returns' to increase conceptual clarity. However, calling either $\log r_t^{(1)}$ or $r_t^{(2)}$ 'log-returns' makes little difference in practice, as the difference is negligible as $\log(r_t^{(1)}) \approx r_t^{(1)} - 1 = r_t^{(2)}$ for values of $r_t^{(1)}$ close to 1. Log-returns are computationally convenient and numerically stable. Using log-returns is also suitable to make the time series stationary. To see why, we can look a different route of transforming the data: First we take the log of the time series to transform any exponential trend into a linear one and to stabilizes the

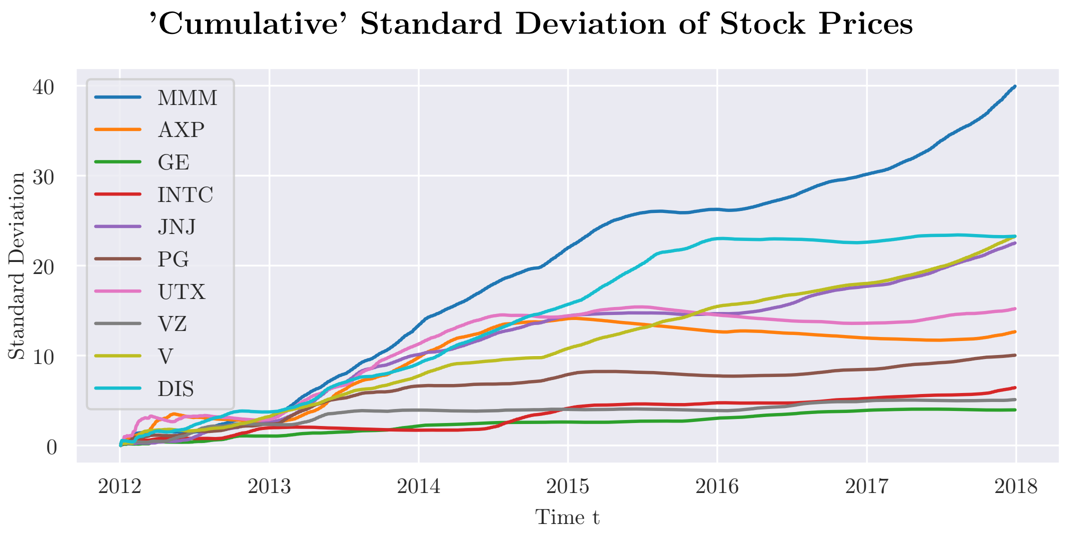


Figure 2.2: Standard deviation for the time series of stock prices. The value of the graph at point t is calculated as the standard deviation of all recorded values of the respective stocks up to that point t.

variance. Removing the linear trend altogether can then be achieved by differencing the time series (see Shumway and Stoffer (2011), p.61). We arrive again at the log-returns as $\log r_t^{(1)} = \log \frac{y_t}{y_{t-1}} = \log y_t - \log y_{t-1}$.

Autocorrelation and Partial Autocorrelation

To better understand the effect of the transformation, we need to look at the autocorrelation and partial autocorrelation of the time series. The autocorrelation at lag j is defined as the correlation between an observation at time t with the observation at time t - j. For a stationary series, the autocorrelation does not depend on t, but only on the number of periods that lie between one observation y_t and another observation y_{t+h}. The autocorrelation function (ACF) can then formally be expressed as

$$\text{ACF}(h) = corr(y_t, y_{t+h}). \tag{2.1}$$

The partial autocorrelation between an observation y_t and another observation y_{t+1} is the correlation between y_t and y_{t+h} that is not already explained by a linear dependence on the observations in between y_t and y_{t+h}. Formally the partial autocorrelation function (PACF) can be defined as

$$\text{PACF}(h) = corr(y_t - \hat{y}_t, y_{t+h} - \hat{y}_{t+h}), \tag{2.2}$$

where $\hat{y}_{t+h} = \beta_1 y_{t+h-1} + \beta_2 y_{t+h-2} + ... + \beta_{h-1} y_{t+1}$

and $\hat{y}_t = \beta_1 y_{t+1} + \beta_2 y_{t+2} + ... + \beta_{h-1} y_{t+h-1}$

are the linear combinations $\{y_{t+1}, ..., y_{t+h-1}\}$ that minimize the mean squared error of a regression of y_{t+h}, and y_t respectively, on $\{y_{t+1}, ..., y_{t+h-1}\}$. Both $y_t - \hat{y}_t$ and $y_{t+h} - \hat{y}_{t+h}$ are uncorrelated with $\{y_{t+1}, ..., y_{t+h-1}\}$. Representative of all, Figure 2.3 shows the ACF and PACF for the closing prices and log-returns of the stock of Procter&Gamble (PG). One can see in the top two plots that the ACF of the original time series very slowly decays and that the partial autocorrelation at lag 1 is very large and then cuts off. This again is a sign that differencing was an appropriate transformation (see Shumway and Stoffer 2011, p.

145) to make the series stationary. After the transformations (bottom plots) the partial autocorrelation at lag 1 has vanished and the autocorrelation has mostly dropped to insignificance. We can also see that we have not induced any negative autocorrelation, the data therefore is not overdifferenced. Figure 2.4 shows the log-returns of Procter&Gamble

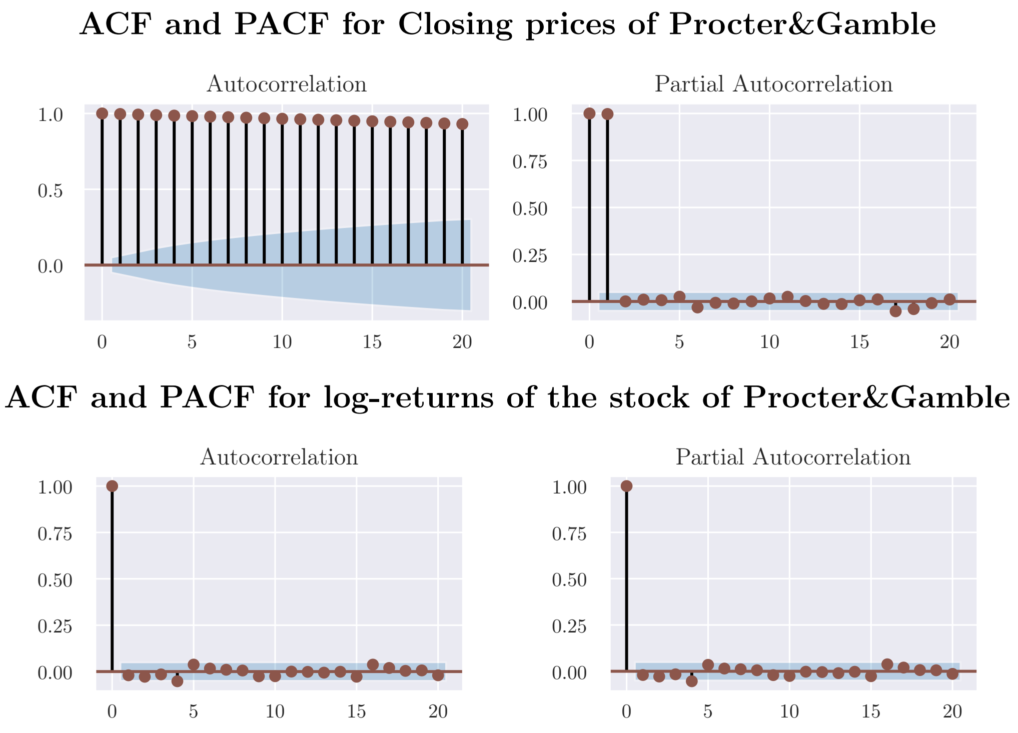


Figure 2.3: Autocorrelation function (ACF) and partial autocorrelation function (PACF) for the Adjusted Closing Prices (top) and log-returns (bottom) of Procter&Gamble. The ACF and PACF for the other stocks look very similar and are therefore not shown here.

(again, representative of all). The trend has disappeared and the data visually looks similar to white noise. Also the means of the time series are very close to zero (as shown in Table 2.3). In addition we have performed a formal test whether our data is stationary or not, the augmented Dickey-Fuller test (ADF) (Said and Dickey, 1984). The null hypothesis of the ADF is the presence of a unit root (implying non-stationarity), the alternative hypothesis then represents (weak) stationarity of the time series. P-values of the ADF applied to the log-returns of all 10 time series were smaller than 10^{-12} even after correcting for multiple testing, so we can safely assume the time series are stationary.

Means of the log-returns for all Stocks

	MMM	AXP	GE	INTC	JNJ	PG	UTX	VZ	V	DIS
mean	0.000786	0.000534	0.000110	0.000548	0.000616	0.000337	0.000450	0.000372	0.001068	0.000739

Table 2.3

Distribution of the Log-Returns and Conditional Heteroskedasticity

The log-returns are, however, not normally distributed. By looking at Figure 2.5 we can clearly see that the distribution has fat tails: extreme events appear more often than would be expected if the data was normally distributed. The time series are also not homoscedastic. While stationarity implies that the unconditional variance is constant

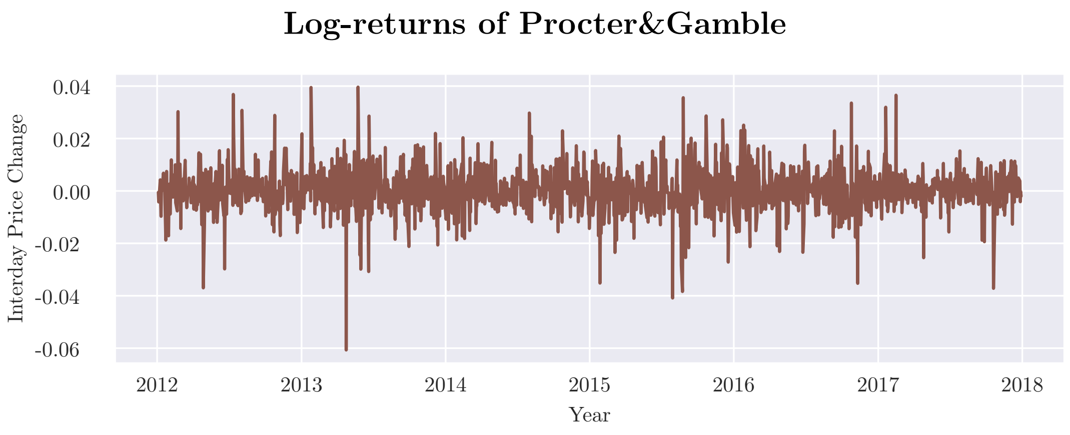


Figure 2.4: Log-returns of Procter&Gamble. Visually the data looks similar to white noise. The log-returns of the other stocks look virtually identical and are therefore not shown.

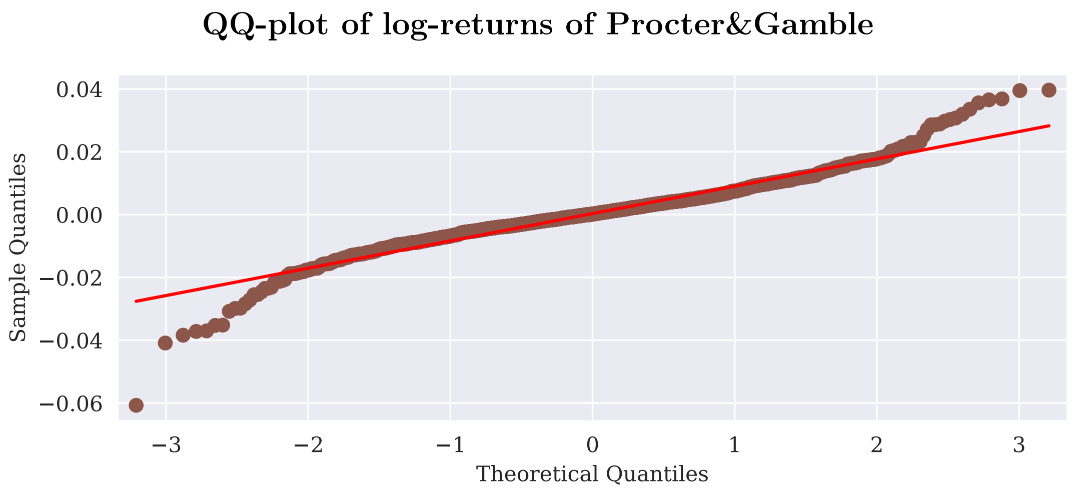


Figure 2.5: QQ-Plot for log-returns of Procter&Gamble. QQ-plots for the other stocks look virtually identical and are therefore not shown.

over time, the variance of the time series may fluctuate conditional on past observations (see (Engle, 1982)). Indeed, this so called conditional heteroskedasticity is quite common in financial data. The pattern can be observed in Figure 2.6 which shows the squared log-returns of Procter&Gamble as an approximation of the time series' variance. Figure 2.7 shows the ACF and PACF of the squared log-returns of three selected stocks. One can see that all of them exhibit at least some autocorellation. In Chapter 4, we will try to model this volatility by assuming a time series model for the conditional variance.

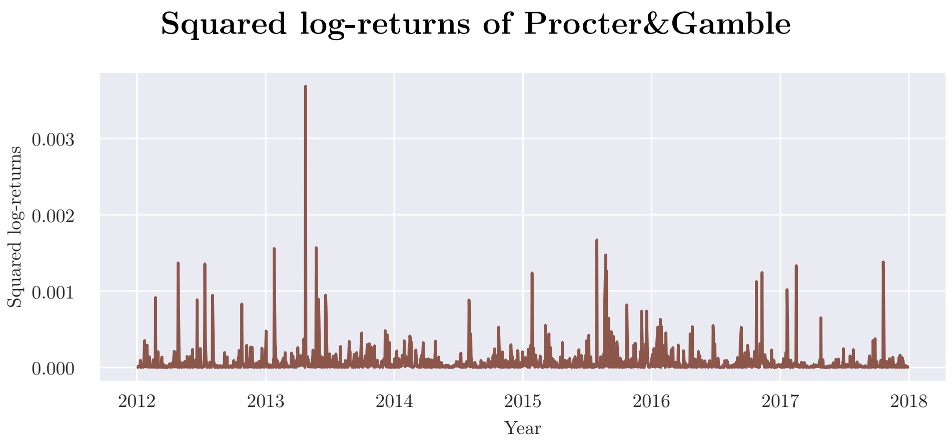


Figure 2.6: Plot of squared log returns of Procter&Gamble. This serves as an approximation of the variance of the log returns, as $Var(x) = E[(x - E(x))]^2$ and the mean of the log returns is close to zero. The pattern looks rather similar for all stocks, therefore only one is shown.

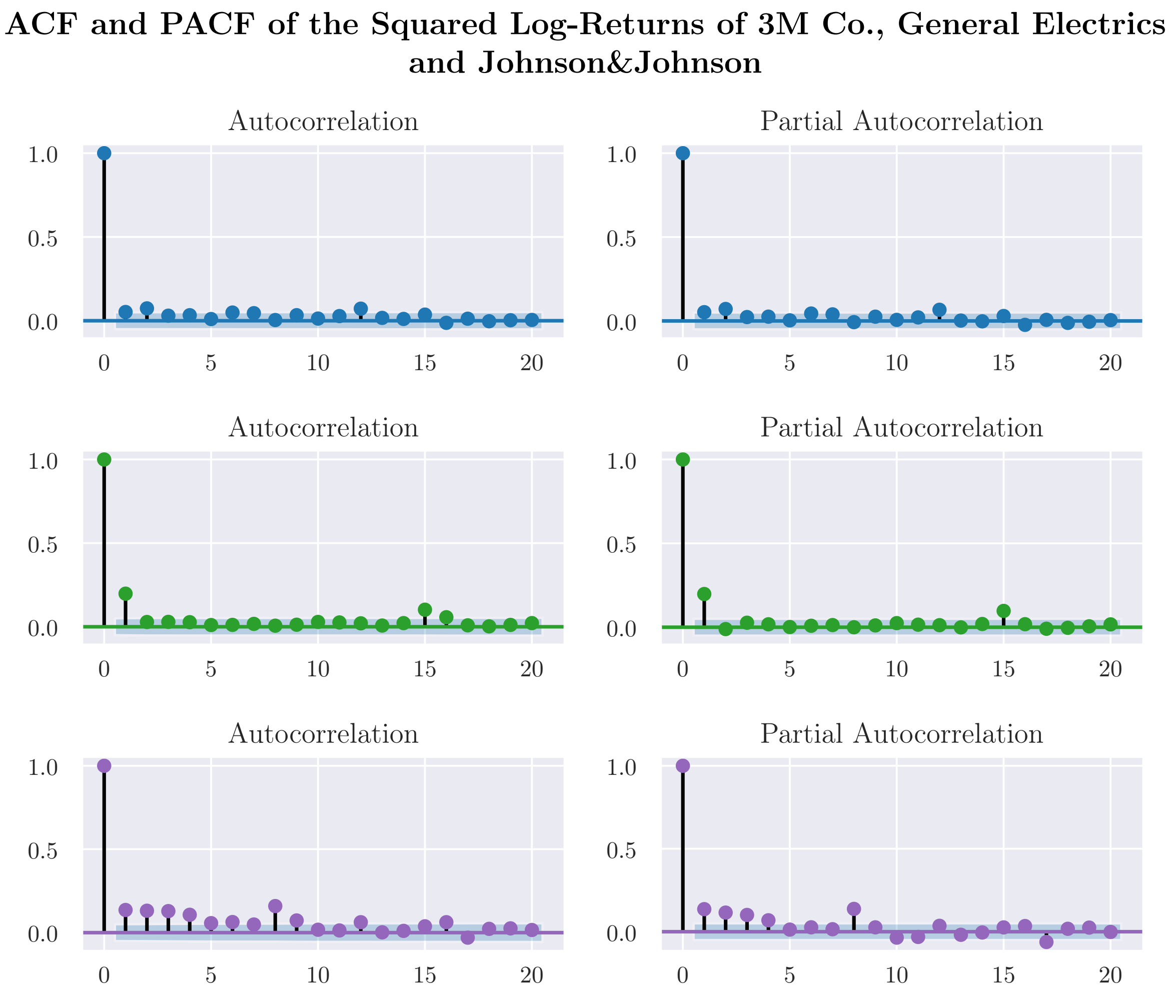


Figure 2.7: ACF and PACF of squared log-returns of 3M Co., General Electrics and Johnson&Johnson. We see that some of the squared log-returns exhibit indeed autocorrelation, while others do less so. Strong autocorrelation implies that there is information about the future in the time series that can be modeled.

2.2 Analyst Reports

As an addition to the financial time series, we were given information extracted from 17153 analyst reports provided by Thomson One. The reports are part of Thomson One's so called "company research" data. They were usually written by analysts employed by brokers like "Deutsche Bank Research" who have specialized in specific industries. Mostly, the reports contain general research documents like SWOT analyses or sector reports. Typically, they also include e.g. earnings per share prognoses or Buy-Hold-Sell-Recommendations. The reports were pre-processed before the data was given to us. Structuring elements like tables and graphs were discarded and the remaining text was broken up and saved in a `.csv`-file. This file formed the basis for our text analysis and the sentiment scores we have generated ourselves, see Chapter 3.

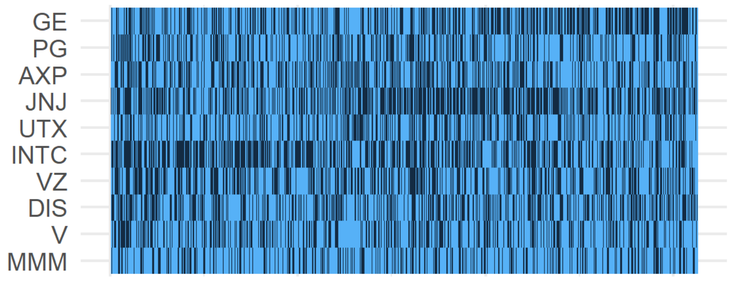


Figure 2.8: Missing Data (blue) on a daily base, weekends excluded, for selected stocks

The reports span the period from 2012-01-02 to 2017-12-28, but unfortunately there a lot of days without any reports (illustrated in Figure 2.8). Also the reports follow a seasonal structure: in Figure 2.9 one can see that starting with a gap around Christmas reports are released with quarterly peaks.

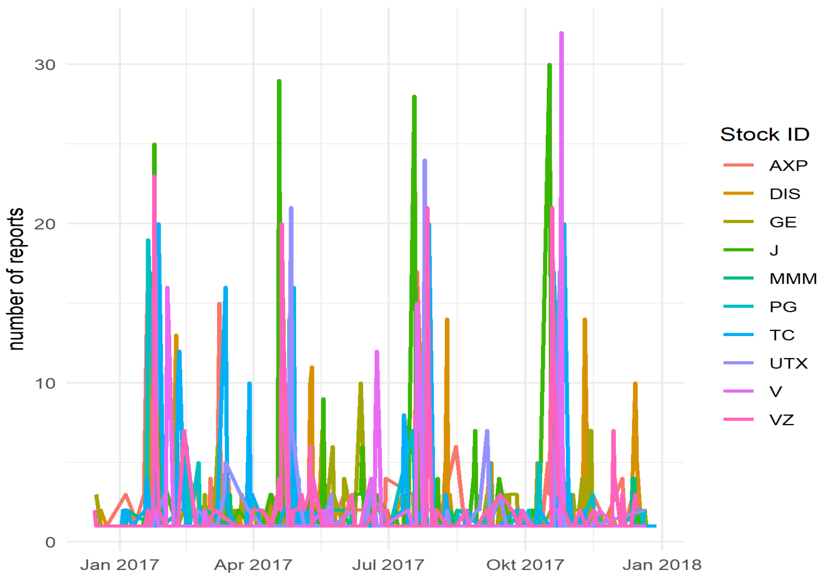


Figure 2.9: Clustering of analyst reports around certain dates

2.3 Ravenpack Sentiment Data

As a second external data source for our analysis, we were given already pre-computed sentiment data by our supervisor. The data comes from the financial data provider RavenPack. According to their website (www.ravenpack.com), RavenPack is a leading provider of big data analytics for financial services. Our data set comprises different scores that were computed based on the news recorded from various sources. The data comprises 1021520 observations that span the time frame from beginning of 2012 to the end of 2017. Information is provided with intra day precision. As such, news and events are recorded at their time of appearance and not just on a daily basis. Table 2.4 shows an overview over the RavenPack data. The first variable provided is an indicator for the estimated positive or negative sentiment score called 'ess', ranging from 0 to 100, where 50 is neutral. This goes along with an estimated novelty score called 'ens' (0-100) describing how 'new' the news are and a 'relevance' (0-100) variable that shows how closely the information is related to the company analzyed. Additionally, two variables contain a 90 day rolling summary of events. One displays the percentage of positive events over a 90 day rolling window ('aes'). The other, 'aev' is the sum of events over the past 90 days. Some values are missing and for some of the trading days no information is provided at all.

Overview Over the RavenPack Data

timestamp_utc	ticker	relevance	ess	ens	aes	aev
2012-01-01	MMM	99	NaN	NaN	58	73
2012-01-01	MMM	99	NaN	NaN	58	73
2012-01-01	MMM	99	NaN	NaN	58	73
⋮	⋮	⋮	⋮	⋮	⋮	⋮
2013-02-26	INTC	100	61.0	56.0	75	105
2013-02-26	INTC	100	61.0	100.0	75	105
2013-02-26	INTC	100	61.0	100.0	75	105
⋮	⋮	⋮	⋮	⋮	⋮	⋮
2017-12-31	JNJ	100	37.0	100.0	60	549
2017-12-31	JNJ	100	62.0	100.0	60	550
2017-12-31	JNJ	100	49.0	75.0	60	551

Table 2.4: Some sample rows of the RavenPack data.

3. Stock Prediction Using Sentiment Analysis

The value of stocks is affected by external events. But information that moves the markets is also captured for example in news articles, online posts on social media or detailed financial reports. One can try to exploit the information presented in the news before it is fully absorbed by the markets. To do so, text data is processed and sentiments are extracted to infer whether the text conveys rather positive or negative information (see Haddi et al. (2013)). Using this information one can either try to predict stocks directly or to forecast future volatility (see Robertson et al. (2007)). This chapter will explore ways to leverage information from news using machine learning to predict the movement of stocks. It will first extract sentiments from text data using two unsupervised learning methods. Secondly, these sentiments will be used for a binary classification of upwards or downwards movement of the ten stocks using XGBoost.
As we did not have access to good news data, we implemented our approach on the available analyst reports. In contrast to current news, the analyst reports are released much less frequently and focus more on past performance of stocks than new information. As such they are not able to incorporate current events as quickly as the news. The reports also cluster around certain dates with long stretches of no or very few reports in between (Figure 2.9). This makes it unlikely that they are valuable for trading strategies.

3.1 Overview of Existing Methods for Sentiment Extraction

There exist several methods to obtain sentiment scores from text data. Most commonly procedures rely on a library of previously known positive and negative words or groups of words called 4-grams. The simplest method, called bag-of-words methods, simply counts the number of positive and negative words in the text. Other approaches extract parts of the text around the location of specific words and then use Support Vector Machines or Naive Bayes Classifier to generate sentiment scores (see Westerski (2007) for further references). Many of these more advanced sentiment classification techniques are supervised learning methods. As such they need a labelled data set for initial training. They are therefore not applicable to our 17153 analyst reports, because these are unstructured and not labelled. Also, the reports have a very specific format and language, therefore other pretrained models or other labelled training data sets could not be used. Other strategies for financial data rely on the availability of intra day trading and news data: By looking at the movement or volatility of the period close after the news release one can estimate approximate sentiment scores (Robertson et al., 2007). As the our stock data is only inter day we could not apply this method.

3.2 Sentiment Extraction - Theoretical Background

Two methods to estimate sentiment scores in an unsupervised way will be explored here. The first one is a library-based approach, the second one is the so called Joint Sentiment Topic Model.

3.2.1 Sentiment Library Method

Our first method for extracting sentiments relies on a library that categorizes words as positive or negative. We can apply this library to the analyst report and obtain the number of positive (P) and the number of negative words (N) in the report. A sentiment or polarity score can then be calculated for each report from the relative frequencies of positive and negative words:

$$Polarity = \frac{P - N}{P + N} \tag{3.1}$$

Some more advanced libraries even measure the positiveness of each word with a numeric score (and not only 1 and -1), but we could not find any library appropriate for our purposes. Instead we used a library from Loughran and McDonald (2016) as simple word list. An excerpt from the list can be found in Table 3.1.

	positive $n = 218$	negative $n = 1282$
1	acclaim	abandonment
2	accomplishment	abdication
3	advantage	abuse
4	assure	acquittal
5	attractiveness	catastrophe
6	delightful	criticize
7	diligent	degrade
8	impress	harsh
9	⋮	⋮

Table 3.1: Excerpt from the sentiment dictionary by Loughran and McDonald (2016)

3.2.2 Joint Sentiment Topic Model

As a second method for extracting sentiments, we use the Joint Sentiment Topic Model by Lin and He (2009). First, the user provides a number of topics to be estimated. Given the desired number of topics, the JST approach then automatically creates topics and assigns probabilities to the individual words of being associated to the different topics. Similar to, e.g. an exploratory factor analysis, one cannot always know why some words are grouped more closely together than others. The JST method heaviliy relies on the ideas of Blei et al. (2003) and uses a Bayesian hierarchical model, called Latent Dirichlet Allocation (LDA) for the topic detection. JST, however, goes one step further by also assigning sentiment scores to words within specific topics.

To better understand the procedure we will first have a closer look at LDA and then discuss the additional features of the JST. The general idea of LDA is that each document can be described as coming from a distribution of topics. A financial report could for example be classified as being 50% about politics, 30% about environment and 20% about the chemical industry. To be more precise however, the topics are not assigned explicit names or meanings by the model. The classification would therefore rather look like 50% topic 1, 30% topic 2 etc. Each topic can itself be described by a distribution of words (Blei et al., 2003) that indicates how likely a word is to appear in a specific topic. This assumes that the words themselves are independent of each other which is in reality not the case, but for large data sets this assumption seems to be acceptable. The LDA has two Dirichlet priors containing hyperparameters α and β. A high α indicates that each document is likely to contain a mixture of many topics, while a low α indicates that one document is strongly associated with one or few topics. Similarly a high β implies that the words contained in each topic have a high overlap, while a low β means that words are strongly associated with only one topic. LDA backtracks from the document level to identify topics that are likely to have generated this specific corpus of words. The underlying optimisation of the model is done using a Gibbs-Sampler. In the first step, each word is assigned a probability to belong to each of the T topics. Those probabilities of course sum up to one. each document also gets a probability to contain each of the T topics. The algorithm then iterates over the words, topics and documents, updating the underlying probabilities. In the Joint Sentiment Topic model the LDA is extended by adding a layer in between the document and the topic layer that determines a sentiment (Lin and He, 2009). The new sentiment layer can be associated with documents, followed by topics and then words. For a detailed mathematical definition see Lin and He (2009). While the method is very good at estimating the association of words to the main topics that were present in the training data, it is ill prepared to deal with information that is very specific or completely new. If something happens, e.g. a data breach at Visa, that has never before occurred in the training data, the model cannot categorize it. The model is also not good in dealing with words that are too specific to really fit inside any of the pre-specified number of categories identified at the initial training stage. It may, however, be really good at capturing the general mood that is associated with words found across a large number of documents.

3.3 Cleaning of the Text Data

In order to obtain useful results from our models, the data first has to be cleaned. The entire set of analyst reports contains 157380 unique words, characters and symbols. Many of them, however, do not really convey meaning or relevant information. To get reliable sentiment scores, the text data therefore has to be cleaned and preprocessed in order to remove noise and reduce the dimensionality of the unsupervised learning problem (Haddi et al., 2013). The preprocessing was done in four steps using `R` (R Core Team, 2017) and the `R` package `tidytext` (Silge and Robinson, 2016). At first, words where converted to lowercase and all words where saved as separate strings. Stop words (like 'the', 'in', 'at') were removed by using a custom stop word library. There exist a lot of readily available stop word libraries, both from the *tidytext* package and from other sources. One problem, however, with many libraries is that they also contain many adjectives that might be important for the extraction of reasonable sentiments. We therefore created our own library of stop words by modifying the existing `tidytext` stop word library. This first step of removing stop words reduces the total number of words from 67M initially to 43M (see Figure 3.1). In the second step all special characters, links to websites, hyper-references, numbers, words with numbers and punctuation marks are removed as well. The result reduced the number of unique words and items by about 50% from 157206 to 82525.

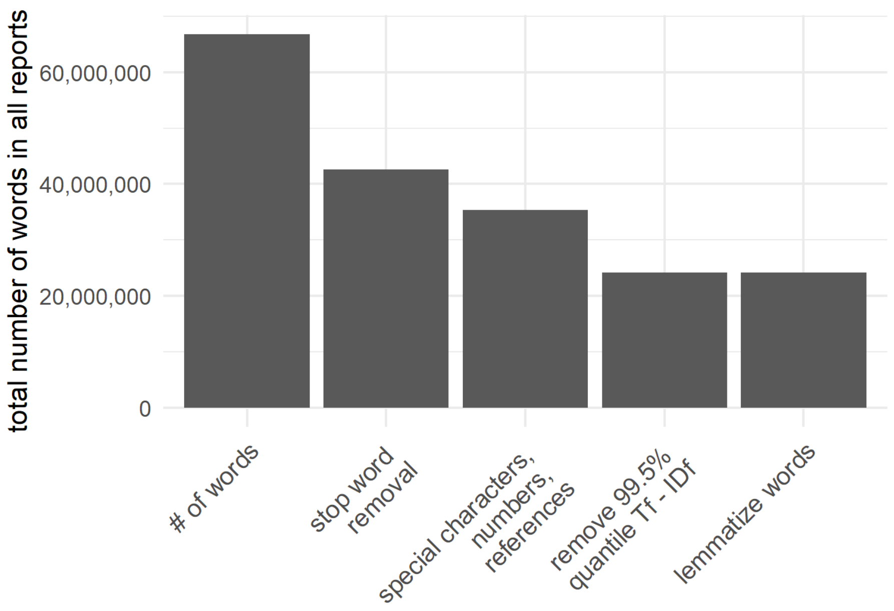


Figure 3.1: Reduction in total number of words due to preprocessing the data.

As explained earlier, the JST model is not good at dealing with words and topics which occur very frequently in one document, but not in any of the others. Those words that are too specific therefore have to be identified and removed. On way to do so is to look at the so called 'Term Frequency Inverse Document Frequency' (TF-IDF), which puts the feature frequency (FF) (also called term frequency (TF)) in relation to the inverse document frequency (IDF). A high Term Frequency Inverse Document Frequency indicates that a word appears very often in one document but is very rare in others. The TF-IDF of a specific word in a certain document is calculated as: (Na et al., 2004).

$$\text{TF-IDF} = \text{FF} * \log \frac{\text{N}}{\text{DF}}, \tag{3.2}$$

where FF is the absolute number of occurrences of that word in the document, N represents the total number of documents in the text coprus, and DF is the number of documents in the corpus that contain the word. The result is a number between 0 and 1 for each word and document. For a large number of words, as in our case, TF-IDF values are usually quite low. For topic modelling like LDA the lower 10% of words judged by TF-IDF are often removed, because they contain relatively frequent words over all documents. In our case however, the words to be removed contained relatively many adjectives relevant to sentiment extraction. We therefore did not remove the lower 10% of words. We did, however, remove the highest ranked 0.05% of words in every document as those are words very specific to this document, but very rare in others. Removing the 99.5 percent upper quantile of TF-IDF words seems indeed to improve the performance. An additional 10293 unique words were removed. The fourth and last step consists of lemmatizing the words using the *textstem* package (Rinker, 2018). Lemmatizing words means reducing them to their inflectional forms. The words 'are, is' are for example reduced to the word 'be'. After these four cleaning steps the number of individual words and items was reduced from 157380 to 64228 by about 60 Percent and of the total number of words only 36

Percent are left. The summary statistics for the total number of words in each report in Table 3.2 show the result more clearly.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
original reports	16	1913	3230	3834	4785	21502
cleaned reports	0	729	1232	1400	1803	7218

Table 3.2: Summary statistics for the number of words in each report. While originally the median report contained 3230 word we see it is now only a third of that.

3.4 Sentiment Extraction - Results

In the following, we will present the results for the sentiment extraction performed with the sentiment library method and with JST. Figure 3.2 shows the distribution of the sentiment scores we obtained after using the document feature matrix (Benoit et al., 2018) with the sentiment library in R. The polarity score was then calculated like in equation 3.1. The histogram shows an approximate normal distribution around zero. This indicates that the sentiment extraction has worked and that the the imbalance in the number of positive ($n = 218$) and negative ($n = 1282$)) words does not affect the results negatively.

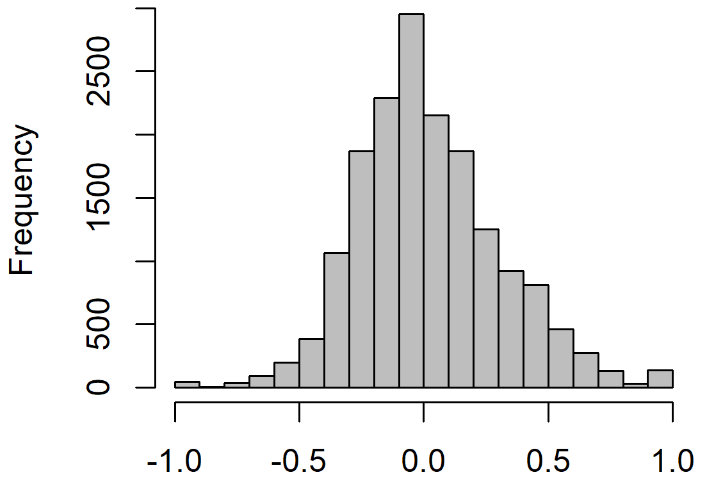


Figure 3.2: Histogram of the sentiment scores computed by the sentiment library

The output of the JST models were generated using the `rJST` package in `R` (Boiten, 2018). To improve the estimation, we have also added the sentiment dictionary used previously for the library method as prior information. From the model we obtained a set of posterior probabilities that link words, topics and documents together with sentiment scores. Based on a training data set that is large and diverse enough, those posterior probabilities can then be used to determine the topics and sentiment in a new document from the words used in them.

Because we do not have any labelled training data we cannot validate the results of the model externally, e.g. with the accuracy scores used in Lin and He (2009). Instead we have to examine whether the association of words, topics, and sentiments proposed by the model look reasonable. We have always used two sentiments and tried different values for the number of topics to be estimated. We achieved the most reasonable results with 30 topics. Using the custom stop library as previously described and removing the top 0.05% words as indicated by TF-IDF also improved the results. Due to the fact that we only have analyst reports for ten different stocks the results of the JST should still be regarded critically. Table 3.3 gives an overview over the words with the highest posterior probability for sentiment one on the topics one, two, three and thirty (see Table A.1) in the appendix for sentiment two. It is still visible that the documents come from a business context, but there are few words that directly indicate a sentiment like 'good', 'growth' and 'high'. This makes it hard to tell if the calculated sentiment one means 'positive', 'negative' or something completely different.

	topic1sent1	topic2sent1	topic3sent1	...	topic30sent1
1	good	year	customer		price
2	look	margin	year		volatility
3	company	will	service		day
4	just	good	will		stock
5	question	growth	business		financial
6	mean	market	believe		average
7	make	expect	continue		expect
8	want	service	datum		year
9	right	continue	growth		next
10	business	high	line		group
11	talk	next	fix		express
12	people	single	also		report
13	time	still	price		month
14	content	also	industry		rate
15	without	digit	time		person

Table 3.3: Words for sentiment one with highest posterior probability per topic

Another way to evaluate the model is to take a closer look at the posterior probabilities for the sentiments of all documents. The histogram in Figure 3.3 displays a split between a documents associated with sentiment one and others associated with sentiment 2. Note that with only two sentiments, the probabilities for a single document of belonging to sentiment one and to sentiment two add up to one. One could therefore obtain a histogram of sentiment two by mirroring this histogram at 0.5. The calculated sentiments clearly seem to lean towards the left, which indicates that sentiment two was more prominent in the data.
The results of the unsupervised Joint Sentiment Models are difficult to evaluate and we have to be at least wary of the results we obtained. Problematic in this case could also be, that the JST views the documents as bag of words and therefore also ignores the ordering of words. Business language contains a lot of fixed phrases that are not considered here. Nonetheless they might be an informative feature for the estimation of stock price movements in the next section.

3.4.1 XGBoost on Sentiment Data

Based on the RavenPack data and our own custom sentiment scores we tried to make predictions for the movement of the time series of different stocks. Inspired by different

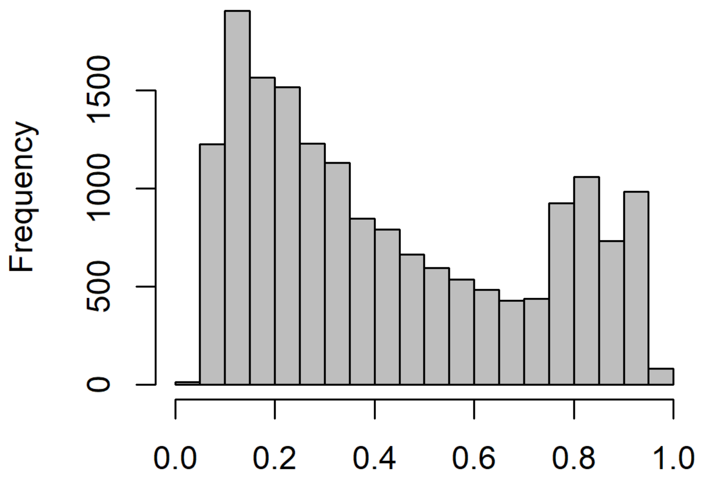


Figure 3.3: Histogram of the sentiment scores computed by the Joined Sentiment Topic model

Kaggle competitions and (Li et al., 2018) we decided to try this using XGBoost for binary classification of upwards of downwards movement (which largely ignores the special time dependent structure of the data). XGBoost is a gradient tree boosting method developed by Chen and Guestrin (2016) and has several beneficial properties. Next to the computational properties it scales well for large data sets and can handle missing values internally. A possible competitor would be LightGBM by Ke et al. (2017) from Microsoft. The general idea behind tree boosting is that they split data leaf-wise with the best fit, whereas other boosting algorithms split the tree level-wise. Splits are done only at the position where the loss can be reduced the most. The model creates an ensemble of a predefined number of classification and regression trees (CART) that complement each other. This tree ensemble method was originally developed by Friedman (2001). XGBoost which stands for "Extreme Gradient Boosting" can be seen as an extension of this framework.

After applying the two previously described sentiment extraction methods, we obtained one sentiment score from each model. For days where there existed multiple reports those scores we computed the average score and the standard deviation. The number of reports per day was also saved in another variable. In addition to the scores we included the log-returns of the previous day (logR yest) and current day (logR today) as well as the trading volume at that day (see Table 3.4). With this we are trying to predict a binary target, namely if the log-returns will be positive or negative the following day. Observations without corresponding analyst reports are omitted. From the RavenPack sentiment data we tried all the given variables together with "logR yest", "logR today" and "Volume". The RavenPack data has records for almost every day but some days have missing values when there was no sentiment recorded. Missing values are handled internally by XGBoost (Chen and Guestrin, 2016). The model learns automatically where to go when a value is missing. For a large enough amount of data this is similar to imputing a value, but based on a reduction on training loss. The data was split into a training, validation and testing split by cutting the time series at predefined dates. Around 80 Percent where used for training and validation. Testing is done out of sample on values after 2016-12-10. For

utc_date	logR today	Volume	logR yest.	sent1	sentBoW	sent1 sd	...
2012-01-03	0.0000	45M	0.0000	0.2790	-0.6190	0.0000	...
2012-01-04	0.0230	48M	0.0000	0.7019	0.2318	0.1843	...
2012-01-05	0.0116	49M	0.0230	0.1413	0.0000	0.0069	...
2012-01-06	-0.0060	36M	0.0116	0.4749	0.1205	0.0257	...
2012-01-09	0.0090	47M	-0.0060	0.6006	-0.6842	0.4526	...
2012-01-11	0.0084	57M	0.0090	0.3607	-0.0296	0.2511	...
2012-01-12	-0.0020	44M	0.0084	0.4279	0.0668	0.1181	...
2012-01-13	-0.0239	63M	-0.0020	0.4603	-0.0076	0.2330	...

Table 3.4: Excerpt from the data frame used for predictions

each stock, an independent model was fit and refitted after each prediction. After testing different hyper parameters we settled on a specification with 200 trees, a maximum depth of three, a learning rate of 0.05 and $\eta = 0.1$. Figure 3.4 displays a random tree from the XGBoost tree ensemble for the RavenPack sentiment data.

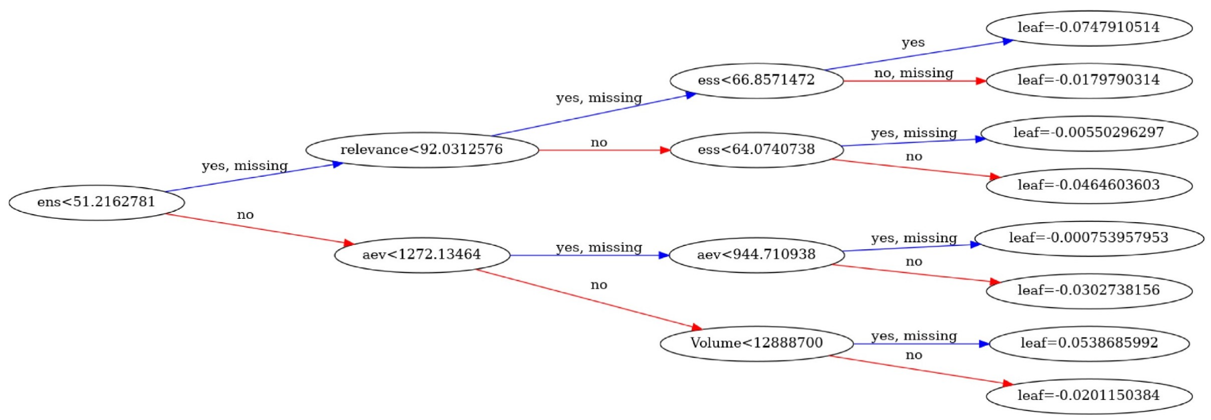


Figure 3.4: One of 200 random trees from the RavenPack tree ensemble

Even though we tried different hyper parameters, none seemed to make a large difference. Prediction accuracy was always around 50 Percent and the F1-measure (3.4.1) a standard evaluation metric of binary classification results (Haddi et al., 2013) was consistently below 0.5.

$$\text{F1-measure} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{3.3}$$

The results for the RavenPack data and our own sentiment scores can be found in Table 3.5 and 3.6. By looking at the cross tables we also discovered that for some stocks class imbalances might cause the algorithm to mostly predict one class. Variable importance plots can help to identify how important certain features are for the trees. As an example we illustrate the use based on the predictions for Disney. In Figure 3.5 the feature importance for the Disney RavenPack data is displayed on the left and the importance for the Disney sentiment scores is on the right. For the RavenPack data the estimated novelty score loads the highest followed by the number of events over a 90 day period. For the sentiment data from the analyst reports trading volume and the two sentiment scores contain the most information according to the model. Over the course of trying different

ticker	f1_score	accuracy	precision	recall
MMM	0.3604	0.5849	0.5740	0.2627
AXP	0.4100	0.5547	0.5061	0.3445
GE	0.5932	0.4981	0.5574	0.6339
INTC	0.4529	0.5169	0.4568	0.4491
JNJ	0.4131	0.5283	0.5432	0.3333
PG	0.5201	0.5056	0.4930	0.5503
UTX	0.3236	0.5584	0.5090	0.2372
VZ	0.4790	0.4830	0.5040	0.4565
V	0.4549	0.5134	0.4308	0.4818
DIS	0.3317	0.4830	0.4788	0.2537
mean	0.4339	0.5226	0.5053	0.4003

Table 3.5: Prediction results for RavenPack sentiment data using XGBoost binary classification

ticker	f1_score	accuracy	precision	recall
MMM	0.2745	0.4788	0.4666	0.1944
AXP	0.5263	0.4943	0.4385	0.6578
GE	0.4836	0.4513	0.5606	0.4252
INTC	0.3055	0.5575	0.3548	0.2682
JNJ	0.2380	0.5362	0.4347	0.1639
PG	0.4556	0.4487	0.4864	0.4285
UTX	0.4556	0.4342	0.4000	0.5294
VZ	0.4742	0.5142	0.5476	0.4181
V	0.3437	0.4683	0.3548	0.3333
DIS	0.4948	0.5663	0.6486	0.4000
mean	0.4051	0.4949	0.4692	0.3818

Table 3.6: Prediction results for analyst reports sentiments using XGBoost binary classification

model specifications the sentiment score from the library method reliably outperformed the Joint Sentiment Topic model. The standard deviation of the JST sentiment scores seems to confer no meaning at all for the model. These visual results need to be treated with care, because the model precision appears to be relatively poor. It would have been interesting to obtain a probability or certainty score together with the prediction. Unfortunately, this is not possible with XGBoost. Overall, our results are similar to the ones reported e.g. by Atkins et al. (2018) in that our prediction accuracy is usually lower than 50 Percent.

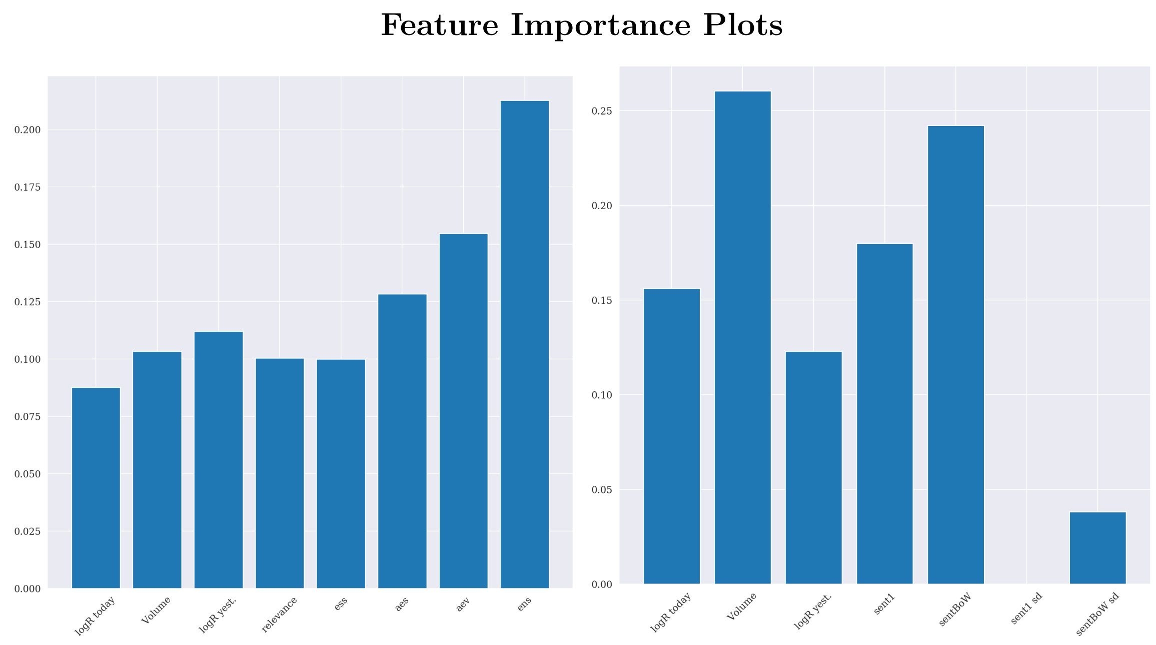


Figure 3.5: On the left the feature importance plot for Disney RavenPack model and on the right the corresponding one for the Disney analyst reports.

4. Predictions Using Time Series

The following chapter will first give a theoretical overview over different time series models and concepts. Then those models will be used to analyze a subset of the data (consisting of two stocks, Visa and Intel) in order to get a thorough understanding for the ways in which stock market data can or cannot be approached. Lastly time series predictions are generated and evaluated for all stocks.

4.1 Theoretical Overview

Time series predictions in this paper will be made using Random Walks, autoregressive (AR), moving average (MA) and generalized autoregressive conditional heteroscedasticity (GARCH) models. Those models are explained in the following.

4.1.1 Random Walks

Random walks serve as the baseline against which every prediction can be compared. Assuming a random walk as the underlying process implies that we know nothing about the future and can do no better than assuming tomorrow's stock price will on average be the same as today. Formally, a random walk follows

$$y_t = y_{t-1} + w_t \tag{4.1}$$

where y_t is the value of the time series at time t and w_t is a random realisation of a stationary white noise process with mean 0 and variance σ^2. Predictions for period t + 1 are therefore exactly the value at time t. We can expand equation 4.1 by allowing for a constant trend, a drift. A random walk with drift can be represented as

$$y_t = \delta + y_{t-1} + w_t \tag{4.2}$$

where δ is a drift parameter. Note that the random walk (with or without drift) is not a stationary process. Note also that a random walk implies that log-returns will be zero (and returns will be one) on expectation, as we expect tomorrows price to be equal to today's price + a random shock with mean zero.

4.1.2 Autoregressive Models

An autoregressive process of order p (AR(p)) implies that the current value of a time series can be described as a combination of the previous p values plus a random shock. As those previous values themselves depend on previous values, the current value is indirectly influenced by its entire past. Formally, an AR(p) process follows

$$y_t = \psi_1 y_{t-1} + \psi_2 y_{t-2} + ... + \psi_p y_{t-p} + w_t \tag{4.3}$$

where y_t is stationary, $\psi_1, ..., \psi_p$ are constants and w_t is white noise. The mean of y_t is assumed to be zero. If the mean is μ instead of zero, equation 4.3 can be rewritten as

$$y_t - \mu = \phi_1(y_{t-1} - \mu) + \phi_2(y_{t-2} - \mu) + ... + \phi_p(y_{t-p} - \mu) + w_t \tag{4.4}$$

which can also be expressed as

$$y_t = \alpha + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + w_t \tag{4.5}$$

with $\alpha = \mu(1 - \phi_1 - ... - \phi_p)$.

4.1.3 Moving Average Models

A moving average process of order q implies that the current value of a time series consists of the average of the previous q observations plus a random shock. As the mean of the time series μ is constant, this average can also be simply expressed as an average of the past random shocks $\{w_{t-1}, ...w_{t-q}\}$. In contrast to the AR(p) process, the shocks affect the future directly (and not only indirectly through past values) and only affect the next q values. Formally, the MA(q) process can be expressed as

$$y_t = \mu + w_t + \theta w_{t-1} + ... + \theta w_{t-q} \tag{4.6}$$

where w_t represents white noise and $\theta_1, ..., \theta_q$ are parameters and q is the number of lags in the moving average.

4.1.4 Autoregressive Moving Average Models

Autoregressive Moving Average Models of order p and q (ARMA(p,q)) form a combination of the above described AR(p) and MA(q) models. Formally, an ARMA(p,q) process follows

$$y_t = \phi_1 y_{t-1} + ... + \phi_p y_{t-p} + w_t + \theta_1 w_{t-1} + ... + \theta_q w_{t-q} \tag{4.7}$$

if the mean of y_t is μ, then the above results in

$$y_t = \alpha + \phi_1 y_{t-1} + ... + \phi_p y_{t-p} + w_t + \theta_1 w_{t-1} + ... + \theta_q w_{t-q} \tag{4.8}$$

with $\alpha = \mu(1 - \phi_1 - ... - \phi_p)$. ARMA(1,0) corresponds to AR(1) and ARMA(0,1) to a MA(1) model. Note that an ARMA(0,0) model for log-returns, a so-called constant-mean model, corresponds to a random walk with drift for the log time series (a constant expected log-return implies $\log y_{t+1} - \log y_t = c - w_t \Leftrightarrow \log y_{t+1} = c + \log y_t + w_t$).

4.1.5 GARCH Models

We have seen in Figure 2.6 that the variance of our time series is not constant over time. This conditional heteroscedasticity can be modelled with a class of models called General Autoregressive Conditional Heteroscedasticity models (GARCH) (see (Engle, 1982) and (Bollerslev, 1986)). While GARCH models cannot themselves forecast future returns of a time series, they can forecast the uncertainty with which a forecast can be made. The GARCH(p,q) model is specified as follows:

$$r_t = \sigma_t \epsilon_t \tag{4.9}$$

where ϵ_t is Gaussian white noise with $\epsilon_t \sim \mathcal{N}(0, 1)$ and

$$\sigma_t^2 = \alpha_0 + \underbrace{\alpha_1 r_{t-1}^2 + ... + \alpha_p r_{t-p}^2}_{\text{autoregressive part}} + \underbrace{\beta_1 \sigma_{t-1}^2 + ... + \beta_q \sigma_{t-q}^2}_{\text{moving average part}} \tag{4.10}$$

In equation 4.9 the returns r_t are therefore modelled as white noise with mean zero and variance variance σ_t^2. When compared to white Gaussian noise with constant variance this can produce a leptokurtic (fat-tailed) distribution similar to what we observed in the QQ-Plots in Figure 2.6. Equation 4.9 is called the mean model of the GARCH(p,q) process. This mean model can also be altered as needed. The GARCH model can then be specified in the following way:

$$r_t = x_t + y_t \tag{4.11}$$

where x_t can be any constant mean, regression or time series process and y_t is a GARCH process that satisfies equations 4.9 and 4.10. In a similar way, the distribution of ϵ_t can be altered. Researchers often assume a t-distribution instead of a standard normal distribution. Glosten et al. (1993) further expanded the model by including a separate term for past negative shocks. The model is sometimes called GJR-GARCH after the authors Glosten, Jagannathan, and Runkle. GJR-GARCH(1,1,1) is specified as follows:

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \gamma_1 \epsilon_{t-1}^2 I(\epsilon_{t-1}^2 < 0) + \beta_1 \sigma_{t-1}^2 \tag{4.12}$$

where $I(\epsilon_{t-1}^2 < 0)$ is an indicator function that is one if ϵ_{t-1}^2 is negative and zero else. This means that negative shocks may have a different impact on future volatility than positive shocks, e.g. a sudden drop in a stock will cause the stock to be disproportionally more volatile in the near future.

4.2 Full Analysis for two Stocks

In the following we will, representative of all, analyze the stocks of Visa (V) and Intel (INTC) in depth to get a thorough understanding for the possibilities and limits of time series analysis.

4.2.1 Exploring the Time Series and their Autocorrelation Structure

The entire time series could be seen back in Figure 2.1. The plots of the log-returns look very similar to the plot in Figure 2.6 and are therefore not shown. Looking at the ACF and PACF in Figure 4.1 we can see some significant autocorrelation for Visa and practically none for Intel. We may therefore start by assuming that for Visa, AR and MA models of order one or two might be a reasonable try. For Intel it looks like we cannot find any exploitable information in the time series data.
To complement our visual impression we formally check for autocorrelation by applying a portmanteau test for autocorrelation to the time series. The Box/Pierce and Ljung/Box tests both test the null hypothesis of zero autocorrelation in a pre-specified number of lags and only subtly differ in their test statistics (see Ljung and Box (1978)). For large sample sizes researchers usually go with the Ljung/Box test, but we decided to apply both to see whether they yield different results. Figure 4.2 shows the p-values for the first 40 lags for both tests. The test supports the conjecture that there may be some significant autocorrelations that could be used for modeling the log-returns of Visa. Compared to the plot of ACF and PACF however, the test seems overly optimistic. For Intel, the test confirms non-significance for the first few lags. While higher order lags may be significant, modeling a time series process with that many coefficients is almost certain to overfit the data.

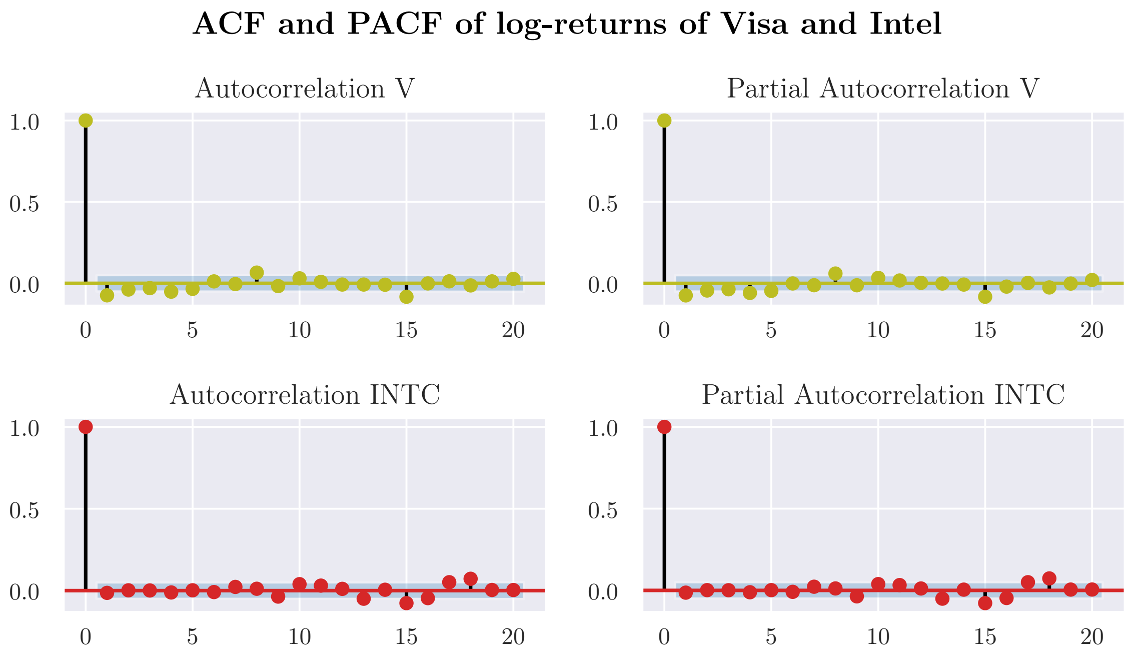


Figure 4.1: ACF and PACF of the log-returns from the stocks of Visa and Intel.

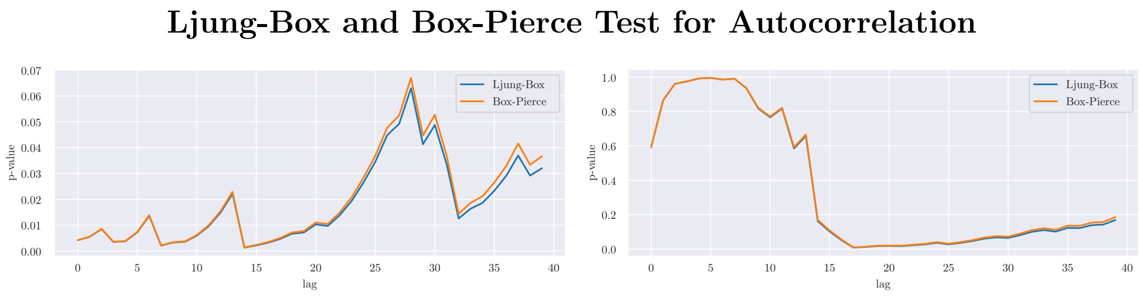


Figure 4.2: Ljung/Box and Box/Pierce test for autocorrelation applied to the the log-returns of Visa (left) and Intel (right).

4.2.2 Applying ARMA models

We start by assessing the fit of different ARMA models to the time series. We did this in python with the `statsmodels` module. The process turned out to be a bit problematic for two reasons: First, fitting turned out to be prone to numerical instability. For some of the models standard errors could not be computed as the algorithm was not able to invert the Hessian matrix. The underlying problem is exacerbated when dealing with GARCH models later on, as all residuals are squared. We eventually managed to alleviate this problem by multiplying log-returns by 100 (corresponding to an approximate percentage interpretation). Secondly, the function `ARMA_model.fit()` from statsmodels was not able to fit all combinations returning an error that "computed initial AR coefficients are not stationary". Maybe this problem could have been solved by supplying specific starting values for the optimization, but we were able to circumvent it by using the function `arma_order_select_ic` instead which produced AIC and BIC values for all models. We chose BIC over AIC as the former more heavily penalizes additional parameters and we are quite concerned about overfitting. Table 4.1 shows the BIC that were obtained from the different ARMA(p,q) models. To allow for comparison later on, models were fitted once with the original log-returns and once with the log-returns multiplied by 100.

BIC for different combinations of ARMA(p,q) for V

(p,q)	0	1	2
0	-8870.52	-8872.19	-8868.22
1	-8871.40	**-8872.93**	-8865.64
2	-8866.97	-8865.65	-8858.30
3	-8861.59	-8859.10	-8851.28
4	-8859.42	-8854.27	-8854.01

(p,q)	0	1	2
0	5027.88	5026.20	5030.17
1	5026.99	**5025.46**	5032.75
2	5031.42	5032.75	5039.26
3	5036.81	5039.29	5037.07
4	5038.98	5044.13	5044.39

BIC for different combinations of ARMA(p,q) for INTC

(p,q)	0	1	2
0	**-8692.98**	-8685.94	-8678.63
1	-8685.94	-8678.62	-8671.31
2	-8678.63	-8671.30	-8672.74
3	-8671.31	-8664.08	-8664.26
4	-8664.18	-8656.92	-8652.19

(p,q)	0	1	2
0	**5205.42**	5212.45	5219.77
1	5212.45	5219.77	5227.09
2	5219.77	5227.09	5226.21
3	5227.09	5234.32	5238.91
4	5234.21	5241.47	5246.20

Table 4.1: BIC presented for different combinations of ARMA(p,q) fit to the log-returns of Visa (top) and Intel (bottom). On the right side, those returns were multiplied by 100 in order to allow for comparison with the GARCH models later on.

For Visa the best model according to BIC is ARMA(1,1). However the difference to ARMA(0,0) seems as good as negligible. In order to avoid the peril of reading too much into random chance, it may therefore be more prudent to stick with a constant mean model (ARMA(0,0)). Recall that commons sense suggests that the past should not contain exploitable information about the future, which, however is an implication of assuming an ARMA(1,1) model. We will fit this model nevertheless, along with the constant mean model. For Intel the lowest BIC is reached by ARMA(0,0) which corresponds to our observation that there is no significant lower-level autocorrelation.
Table 4.2 shows the resulting model outputs. Let us look at the models for Visa first (top and middle). We can see that the change in AIC is larger than the change in BIC as the latter more heavily penalizes the number of parameters included in the estimation. Notably, even though the difference in BIC is quite small, the AR(1) and MA(1) are estimated very significantly and are high in relative magnitude. However, note that their effects go in opposing directions in similar magnitude and may as well cancel each other out. This interpretation may be somewhat plausible as the individual effects of the AR(1) and MA(1) terms are about an order of magnitude smaller (albeit still significant) and go in the same direction when estimating ARMA(1,0) and ARMA(0,1) models separately (-0.0736 for the AR(1) and -0.0809 for the MA(1) term in separate models (not shown)). The bottom table shows the result of the ARMA(0,0) model to the log-returns of Intel. Not even the mean is significant, which is not too surprising, as Intel hardly gained in value in the observed period from 2012 to 2017 (See again Figure 2.1. When exploratively fitting an ARMA(1,0), ARMA(0,1) or ARMA(1,1) model, as expected none of the coefficients reach significance (p-values all > 0.5).

Results for an ARMA(0,0) process fit to the log-returns of V

Dep. Variable:	log_returns	**No. Observations:**	1509
Model:	ARMA(0, 0)	**Log Likelihood**	4442.581
Method:	css	**S.D. of innovations**	0.013
Date:	Wed, 04 Sep 2019	**AIC**	-8881.162
Time:	10:03:08	**BIC**	-8870.524
Sample:	0	**HQIC**	-8877.200

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
const	0.0011	0.000	3.253	0.001	0.000	0.002

Results for an ARMA(1,1) process fit to the log-returns of V

Dep. Variable:	log_returns	**No. Observations:**	1509
Model:	ARMA(1, 1)	**Log Likelihood**	4451.108
Method:	css-mle	**S.D. of innovations**	0.013
Date:	Tue, 03 Sep 2019	**AIC**	-8894.215
Time:	13:15:26	**BIC**	-8872.938
Sample:	0	**HQIC**	-8886.291

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
const	0.0011	0.000	4.194	0.000	0.001	0.002
ar.L1.log_returns	0.6156	0.116	5.307	0.000	0.388	0.843
ma.L1.log_returns	-0.6997	0.105	-6.681	0.000	-0.905	-0.494

Roots:	**Real**	**Imaginary**	**Modulus**	**Frequency**
AR.1	1.6243	+0.0000j	1.6243	0.0000
MA.1	1.4293	+0.0000j	1.4293	0.0000

Results for an ARMA(0,0) process fit to the log-returns of INTC

Dep. Variable:	log_returns	**No. Observations:**	1509
Model:	ARMA(0, 0)	**Log Likelihood**	4353.809
Method:	css	**S.D. of innovations**	0.014
Date:	Wed, 04 Sep 2019	**AIC**	-8703.618
Time:	10:03:08	**BIC**	-8692.979
Sample:	0	**HQIC**	-8699.656

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
const	0.0005	0.000	1.574	0.116	-0.000	0.001

Table 4.2: Results for the ARMA(0,0) (top) and ARMA(1,1) model (middle) fit to the log-returns of Visa and for the ARMA(0,0) model (bottom) fit to Intel

ARMAX models

We proceed in our analysis by adding our own generated sentiments as well as the ravenpack sentiments as external regressors. However, the availability of external data data points does not perfectly match the stock market observations. Unfortunately, we only have about a third as many analyst reports (and therefore sentiment scores) as stock market observations. In order to obtain an estimable model we decided to discard all observations on days where no analyst reports existed. This also implies that we need to confine our analysis to a constant mean model (ARMA(0,0) as autoregressive and moving-average components do not make sense when many observations in the time series are missing. The time series model hence reduces to a regression with an intercept and the sentiment data as independent variables. While analyst reports were rather scarce, the Ravenpack data often had multiple entries per day with fewer days missing. We therefore needed to aggregate sentiments to obtain one single observation per day.

ARMAX(0,0) with analyst report sentiments fit to the log-returns of Visa

Dep. Variable:	log_returns	**No. Observations:**	535
Model:	ARMA(0, 0)	**Log Likelihood**	1508.522
Method:	css	**S.D. of innovations**	0.014
Date:	Sat, 14 Sep 2019	**AIC**	-3009.044
Time:	10:53:20	**BIC**	-2991.915
Sample:	0	**HQIC**	-3002.342

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
const	0.0045	0.002	2.217	0.027	0.001	0.009
sent1_mean	-0.0047	0.003	-1.602	0.110	-0.010	0.001
sentBoW_mean	-0.0007	0.003	-0.241	0.810	-0.006	0.005

ARMAX(0,0) with analyst report sentiments fit to the log-returns of Intel

Dep. Variable:	log_returns	**No. Observations:**	821
Model:	ARMA(0, 0)	**Log Likelihood**	2274.402
Method:	css	**S.D. of innovations**	0.015
Date:	Sat, 14 Sep 2019	**AIC**	-4540.805
Time:	10:53:20	**BIC**	-4521.963
Sample:	0	**HQIC**	-4533.575

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
const	0.0031	0.002	1.766	0.078	-0.000	0.006
sent1_mean	-0.0035	0.003	-1.215	0.225	-0.009	0.002
sentBoW_mean	0.0024	0.002	0.993	0.321	-0.002	0.007

Table 4.3: Results for ARMAX(0,0), i.e. a regression with a constant and our own sentiment data as external regressors. ARMAX(1,1) was not fitted as modelling dependence on specific lags does not make sense if those lags do not reliably exist in the data.

ARMAX Sentiments Analyst Reports
The fact that we need to omit two-thirds of all data points changes the BIC considerably. To be able to compare the models according to BIC we have refit the baseline ARMA(0,0) model to the reduced data and obtained a BIC of -3001.07 for Visa and -4522.15 for Intel (the difference owing to the different number of observations). Table 4.3 shows the results of the ARMAX(0,0) model for Visa and Intel. Both models fare worse in terms of BIC than the baseline.
ARMAX Sentiments Ravenpack

In the Ravenpack data for Visa, for 301 out of 1509 observations there were information on current relevance and no current sentiments scores (leaving only the information about 90 day averages). As the difference between ARMA(0,0) and ARMA(1,1) for Visa was not large anyways, we decided to omit the ARMA(1,1) model for Visa for the same reasons as above. We computed a new baseline BIC as we did in the previous analysis: The BIC for an ARMA(0,0) model with the 301 omitted values is now -7008.87. For INTC there were only 85 missing observations and the baseline BIC for the ARMA(0,0) model is -8238.47. We tried different combinations of regressors. Only a simple model that only included the estimated sentiment score 'ESS' (more precisely, the mean of all sentiments scores of that day) as a regressor, was able to reduce BIC (from -7008.87 to -7015.51 for Visa and from -8238.47 to -8259.24 for Intel). The results for the simple model are shown in Table 4.4. A full model is shown in Table A.2 in the appendix. This result is encouraging as it means that the Ravenpack sentiment scores actually contain relevant information about the market. However, there is one important subtlety hidden in the fact that we were regressing today's log returns against today's sentiment scores. This is fine if we are interested in whether the sentiments do contain any significant information about the target firms. As the log-returns are based on the closing values a significant effect means that information that comes in throughout the day is actually incorporated into the price. However, since we are trying to predict *tomorrow's* log-returns this may also mean that any information has already been priced in at the end of the day. Gidofalvi and Elkan (2001) mention that an effect on the stocks can only be measured up to 20min after the news appear. Indeed, when we included yesterday's sentiment scores instead of today's in the regression, the effect was insignificant (p-value = 0.519 for Visa and 0.318 for Intel, result not shown to avoid redundancy) and BIC worsened. We will therefore not include external regressors when trying to predict stock returns in the following.

GARCH models
As detailed earlier, financial data often exhibit conditional heteroskedasticity. We therefore want to see whether GARCH models are able to improve the fit. Figure 4.3 shows the squared log-returns of V and INTC as well as their ACF and PACF. For both the first lag visually seems to be significant.
We formally test for the presence of a GARCH effect, i.e. autocorrelation in the squared residuals of the time series using the Lagrange Multiplier Test proposed by Engle (1982) for ARCH and Lee (1991) for GARCH effects. For both tests, the null hypothesis of 'No ARCH effect' cannot be rejected with p-values of 0.1160 (V) and 0.2693 (INTC). We nevertheless proceed to fit a GARCH model to the data to see the results. Most often one will find a GARCH(1,1) model in financial literature. While there is extensive debate about the merits of this model (e.g. does anything beat GARCH(1,1) (Hansen and Lunde (2005)), does anything not beat GARCH(1,1)? (alexios, 2013), does anyone need GARCH(1,1)? (Reschenhofer, 2013)), we will for simplicity stick to the GARCH(1,1) with a constant mean model (the mean is modeled by an ARMA(0,0) process). Other models such as for example GARCH(3,1) did not perform significantly better than GARCH(1,1). To avoid issues with numerical instability all log-returns are multiplied by 100 and the new baseline BIC become 5025.46 for Visa (ARMA(1,1)) and 5205.42 for Intel (ARMA(0,0)) as shown on the right hand side of Table 4.1. Table 4.5 shows the result for a GARCH(1,1) fit to Visa, Table 4.6 shows the result for Intel. In both cases the fit improves and the BIC drops - not too much, but noticeably (from 5025.46 to 4948.68 (Visa) and from 5205.42 to 5169.75 (Intel)).

ARMAX(0,0) with Ravenpack sentiments fit to the log-returns of Visa

Dep. Variable:	log_returns	**No. Observations:**	1208
Model:	ARMA(0, 0)	**Log Likelihood**	3518.399
Method:	css	**S.D. of innovations**	0.013
Date:	Sat, 14 Sep 2019	**AIC**	-7030.798
Time:	16:11:38	**BIC**	-7015.507
Sample:	0	**HQIC**	-7025.040

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
const	-0.0063	0.002	-3.127	0.002	-0.010	-0.002
ess_mean	0.0001	3.65e-05	3.717	0.000	6.41e-05	0.000

Full ARMAX(0,0) with Ravenpack sentiments fit to the log-returns of Intel

Dep. Variable:	log_returns	**No. Observations:**	1434
Model:	ARMA(0, 0)	**Log Likelihood**	4140.521
Method:	css	**S.D. of innovations**	0.013
Date:	Sat, 14 Sep 2019	**AIC**	-8275.042
Time:	16:11:38	**BIC**	-8259.238
Sample:	0	**HQIC**	-8269.141

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
const	-0.0122	0.002	-5.000	0.000	-0.017	-0.007
ess_mean	0.0002	4.31e-05	5.323	0.000	0.000	0.000

Table 4.4: Results for ARMAX(0,0), i.e. a regression with a constant and the Ravenpack sentiment data as external regressors.

Results for GARCH(1,1) with constant mean fit to the log-returns of Visa

Dep. Variable:	log_returns	**R-squared:**	-0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	-0.000
Vol Model:	GARCH	**Log-Likelihood:**	-2459.70
Distribution:	Normal	**AIC:**	4927.40
Method:	Maximum Likelihood	**BIC:**	4948.68
		No. Observations:	1509
Date:	Wed, Sep 04 2019	**Df Residuals:**	1505

	coef	**std err**	**t**	**P> |t|**	**95.0% Conf. Int.**
mu	0.1199	2.971e-02	4.034	5.473e-05	[6.164e-02, 0.178]

	coef	**std err**	**t**	**P> |t|**	**95.0% Conf. Int.**
omega	0.0811	5.148e-02	1.576	0.115	[-1.979e-02, 0.182]
alpha[1]	0.0985	3.488e-02	2.823	4.758e-03	[3.010e-02, 0.167]
beta[1]	0.8585	5.388e-02	15.933	3.718e-57	[0.753, 0.964]

Covariance estimator: robust

Table 4.5: Result for the GARCH(1,1) model with a constant mean model fit to the log-returns of Visa

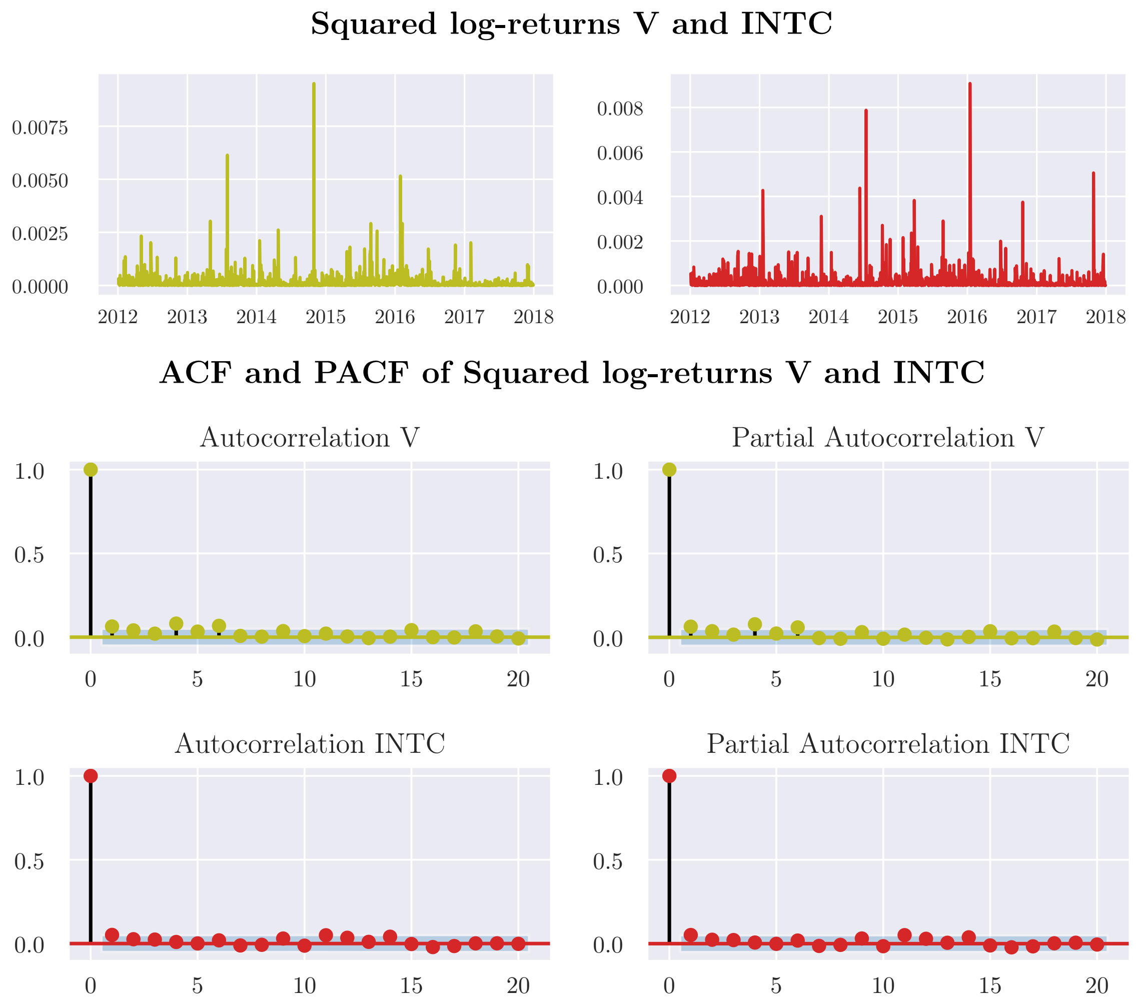


Figure 4.3: Squared log-returns and ACF and PACF for squared log-returns for V (top-left and middle) and INTC (top right and bottom).

We now relax the assumption of normally distributed error terms ϵ_t (see 4.9) and instead assume a Student t-distribution with fatter tails. This time the BIC drops considerably (from 4948.68 to 4794.91 (V) and from 5169.75 to 4986.41 (INTC)). The results are shown in Table A.3 in the appendix. Another expansion we try is the inclusion of asymmetric shocks (see **??**). This assumes that sudden drops in a stock's value lead to higher volatility than an increase of the same amount would. Results for the GJR-GARCH model with t-distributed residuals are shown in Table A.4 in the appendix. For Visa, the gamma-coefficient is positive and significant, indicating an asymmetric effect of shocks on volatility exists. The BIC drops accordingly by a small bit from 4794.91 to 4784.75. For Intel, the gamma coefficient is not significant and BIC even slightly increase from 4986.41 to 4993.23.

Forecasting and Forecast Precision for Visa and Intel

To assess the predictive performance we use the ARMA(0,0) model for Visa and Intel as well as the ARMA(1,1) model for Visa to generate predictions. To do so, we fit an ARMA model to the time series up to time y_t, predict the next value $\hat{y}_{t+1}$, then add the true value y_{t+1} to the time series and fit a model that lets us predict $\hat{y}_{t+2}$ and so forth. For the ARMA(0,0) model, this means essentially predicting the next value on the basis of a cumulative mean of the past log-returns. Estimating an ARMA(1,1) model iteratively for V turned out to be problematic because the result was not always a stationary process.

Results for GARCH(1,1) with constant mean fit to the log-returns of Intel

Dep. Variable:	log_returns	**R-squared:**	-0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	-0.000
Vol Model:	GARCH	**Log-Likelihood:**	-2570.24
Distribution:	Normal	**AIC:**	5148.47
Method:	Maximum Likelihood	**BIC:**	5169.75
		No. Observations:	1509
Date:	Wed, Sep 04 2019	**Df Residuals:**	1505

	coef	**std err**	**t**	**P> |t|**	**95.0% Conf. Int.**
mu	0.0565	3.422e-02	1.650	9.897e-02	[-1.061e-02, 0.124]

	coef	**std err**	**t**	**P> |t|**	**95.0% Conf. Int.**
omega	0.9185	0.289	3.181	1.469e-03	[0.353, 1.484]
alpha[1]	0.2295	9.374e-02	2.448	1.436e-02	[4.576e-02, 0.413]
beta[1]	0.2977	0.168	1.772	7.642e-02	[-3.161e-02, 0.627]

Covariance estimator: robust

Table 4.6: Result for the GARCH(1,1) model with a constant mean model fit to the log-returns of Intel

SSE and Binary Accuracy of Predictions

	SSE	Binary Accuracy	Naive Binary Accuracy
V - ARMA(0,0)	0.2456	54.1528 %	**54.2193** %
V - ARMA(1,1)	0.2445	**55.8139**	54.2193
INTC - ARMA(0,0)	0.2760	51.7608	**52.3588**

Table 4.7: Prediction accuracy for the models we tried for Visa and Intel. SSE is the Error Sum of Squares, Binary Accuracy is the Accuracy of the predicted sign of log-returns.

Therefore, at some points the estimation of an ARMA(1,1) model was impossible. We therefore had to replace the prediction for those specific points with the ones made by ARMA(0,0). To compare the results we looked at the Error Sum of Squares (SSE) as well as a binary prediction accuracy that merely judges whether the direction of the prediction was correct. The predictions of the ARMA(0,0) model with constant mean (which corresponds to a random walk for the log stock prices with drift) seems to at least perform better then mere coin tosses. However they do not perform better than the naive strategy of always predicting an upwards movement of the stock (i.e. a positive return). Only the ARMA(1,1) model for V reaches a higher prediction accuracy. This might, however, also be due to chance. Table 4.7 shows the results of those predictions. The forecasts are shown in Figure 4.4.

Forecasting Volatility

While GARCH models are not able to better predict future returns, they can help predict the volatility of future returns. Figure 4.5 gives an intuition to the usefulness of volatility forecasts. In the figure, the predicted volatility is plotted together with the absolute value of the log-returns of the stock. While obviously the log-returns themselves do not equal not the actual volatility, the figure serves the point of illustrating how future turbulences

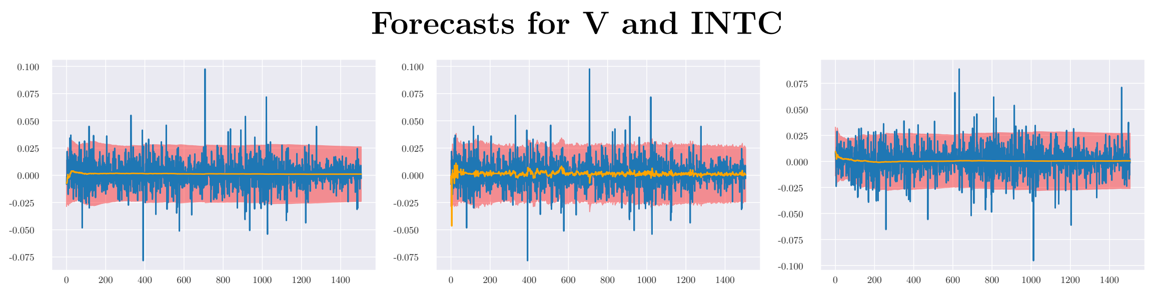


Figure 4.4: The figure shows predictions for log-returns plotted (orange) with confidence intervals (red) against the real returns. The left plot shows the the predictions generated by ARMA(0,0) for V, the middle predictions from ARMA(1,1) and the right plot shows predictions for INTC generated by the ARMA(0,0) model.

in the markets seem at least somewhat predictable. Note that the direction of movement remains still unpredictable, we can merely forecast uncertainty about future returns. This is nevertheless very useful, e.g. for balancing portfolios, pricing options, and estimating future risks. As the main focus of this paper is the prediction of returns, however, volatility forecasts will not be further discussed in detail.

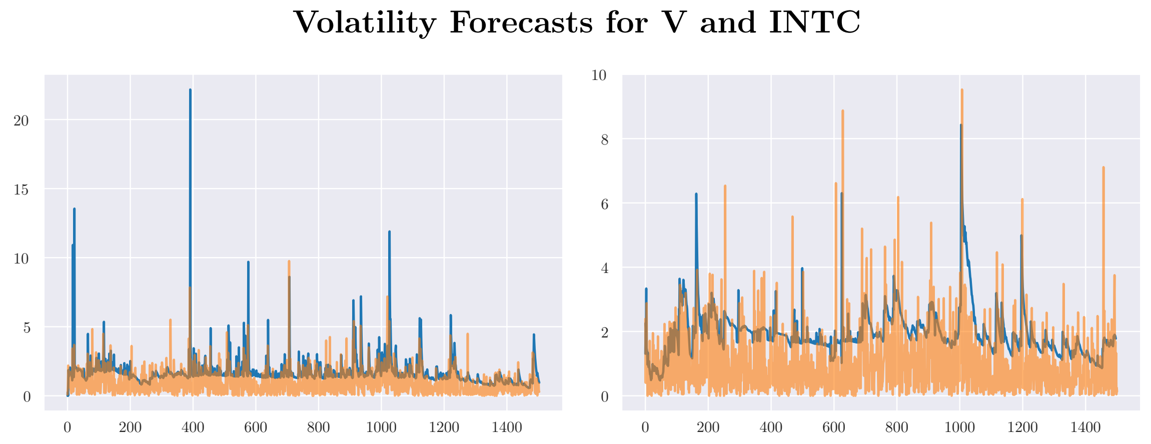
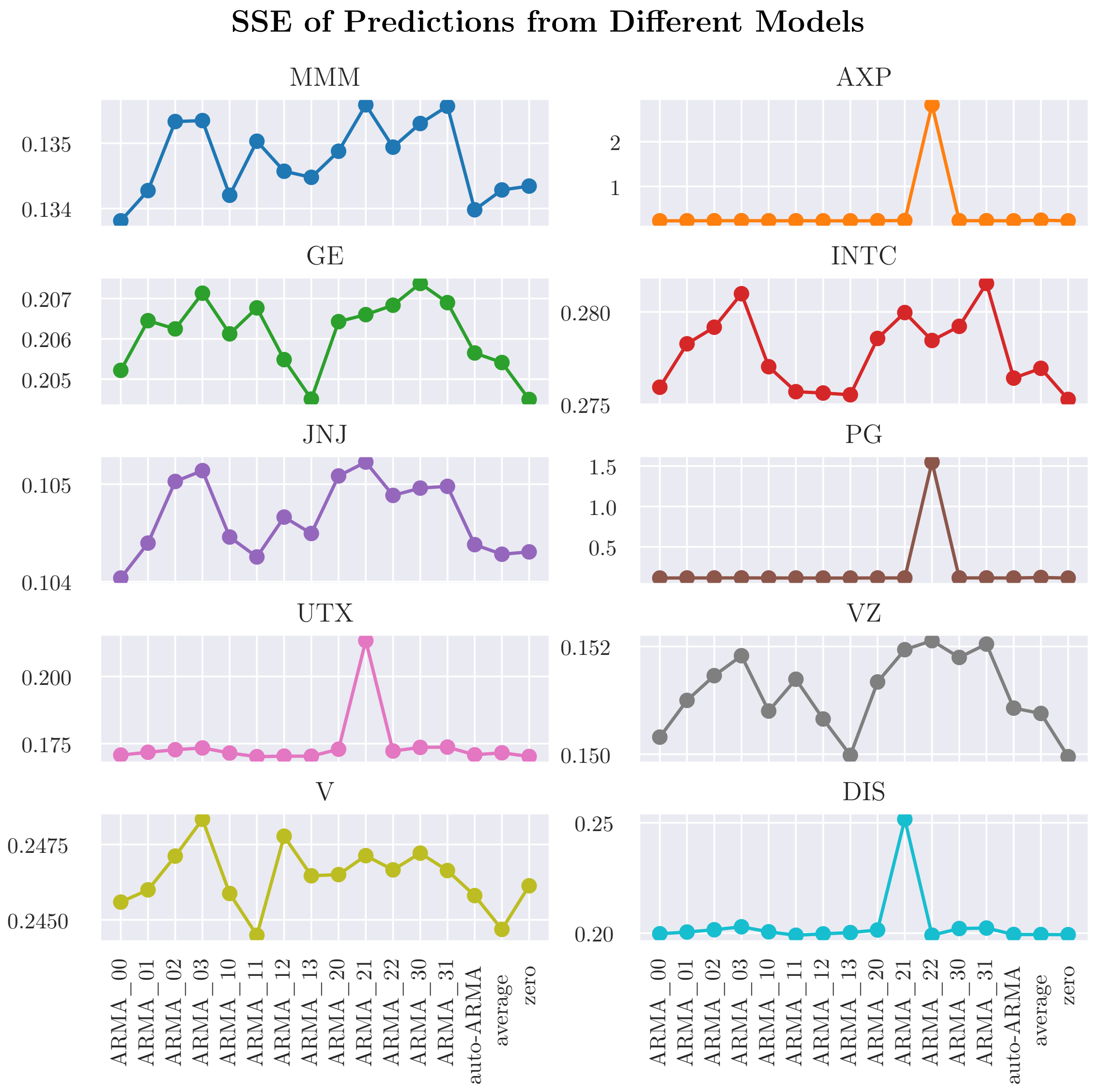


Figure 4.5: Volatility forecasts (shown in blue) are laid over the absolute value of log-returns (in orange) for V (left) and INTC (right). For V, the GJR-GARCH(1,1) model was used, for INTC, the GARCH(1,1) model was used (both using students-t-distributed resiudals).

4.3 Time Series Predictions with the Remaining Stocks

We now directly step into the shoes of a trader who is trying to predict future stocks. The trader has multiple options: He could apply different models to a pre-specified past of a time series, see which one fits best and then stick with it and hope that the future will behave similar to the past. He could iteratively refit the time series every day to produce the next forecast with the current best guess. Or he could pre-select different models and average over the predictions generated by these models. Seen from the angle of economic theory, all three methods somewhat resemble reading the leaves. It remains unclear why the future of the time series should behave like its past. And while the last strategy at least has the beauty of approaching the problem from a Bayesian perspective by averaging over possible future paths, it remains unclear which models should be included and what weight they should be given. But as traders all over the world are indeed trying to tackle the problem in similar ways, we have followed their lead to see what comes out.

We tried all combinations of ARMA(p,q) models that satisfy $p, q < 3$ and $p + q \leq 4$. This restriction on the one hand reduces excessive overfitting and on the other hand was a necessary requirement as computations already took about 12h to complete. Estimation for some models was again problematic (especially ARMA(1,2) and ARMA(1,3) as oftentimes the resulting parameters did not produce a stationary model. In this case, predictions were assumed to be zero. In addition to those models we have written an auto-ARMA function that generates predictions from the ARMA(p,q)-model that fit the past history best in terms of BIC. Yet another set of predictions was generated by averaging over all predictions from the other models. Lastly we also tried always predicting zero for future log-returns. This corresponds to assuming a random walk without drift for the original time series, as predicting a log-return of zero implies a return of $\exp 0 = 1$ which in turn implies that we always predict tomorrow's value of the stock to equal today's value.
Results are shown in Figure 4.6 and should strongly discourage from trading merely on time series predictions. We first looked at the SSE for a variety of different models for all 10 stocks. To assess the performance of the different models, they were then given a rank according to their performance on a single stock (judged by SSE). Those ranks were summed up to produce the second plot in Figure 4.6. We can see that none of the models consistently beat ARMA(0,0). Even more interestingly, none of the models consistently beat the zero-prediction model. So even though many of the stocks actually exhibit some kind of positive long-term trend, a random walk without drift yields better predictions than a random walk with drift.
In addition to the SSE, we also assessed performance in terms of binary predictions accuracy, as shown in Figure 4.7. In the top plot one can clearly see where the model estimations were problematic: Some values are outside of the bounds because model estimation often failed and therefore corresponding predictions were per default set to zero, leading to a very low binary prediction accuracy. We can also see from the bottom plot of the figure, that no model was able to consistently beat the naive strategy of always predicting positive returns.



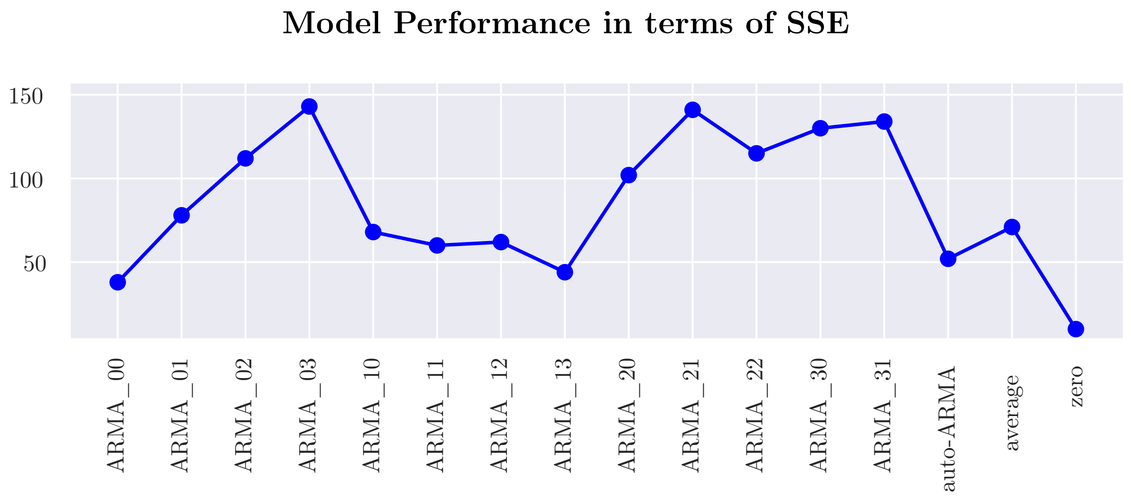
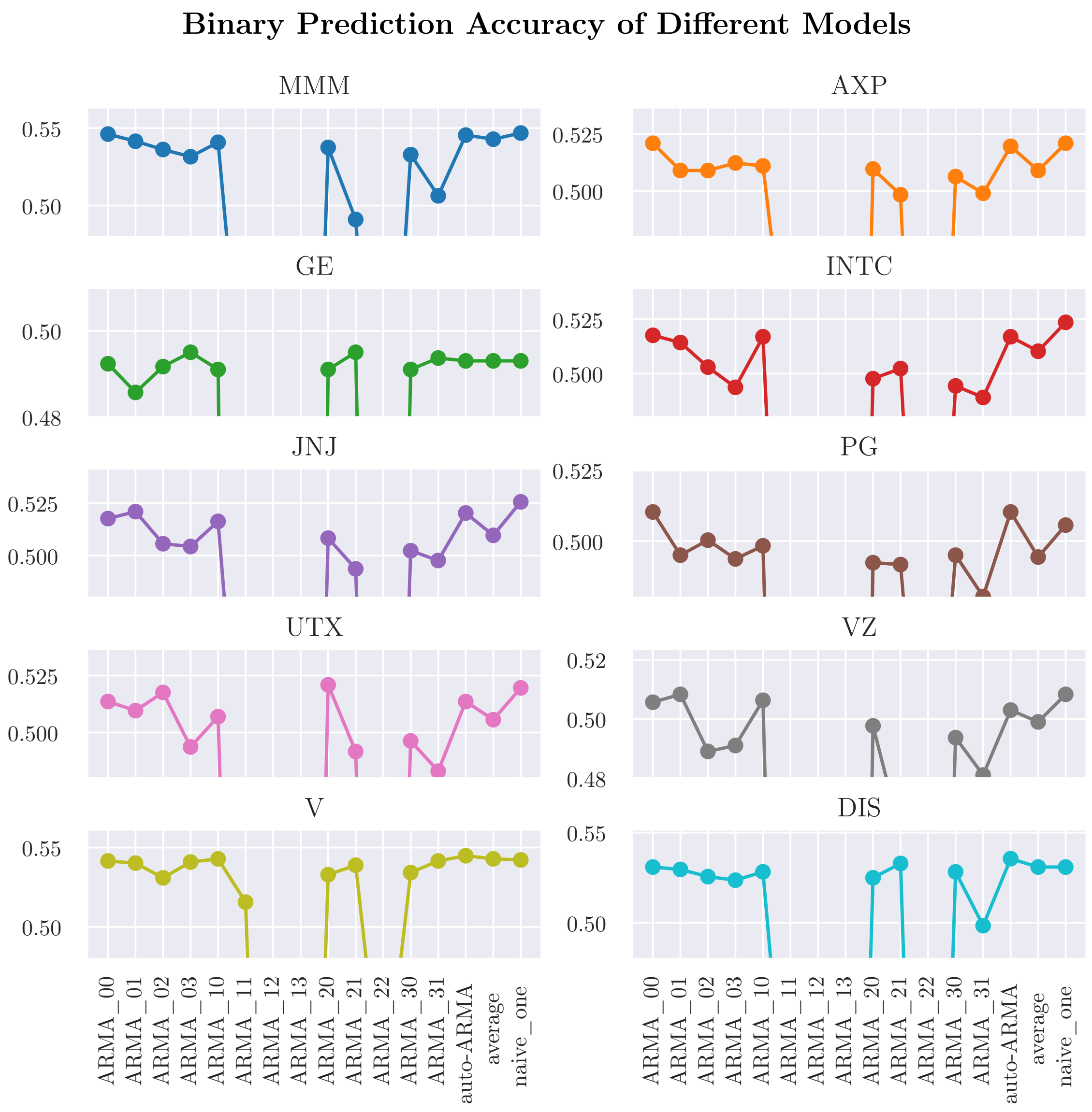


Figure 4.6: The top plots show the SSE reached by the various models for every single stock. Then the different models were given a rank corresponding to their SSE for the different stocks. This bottom plot shows the sum of the ranks achieved by the model across all stocks. As we look at SSE, a lower value is better.



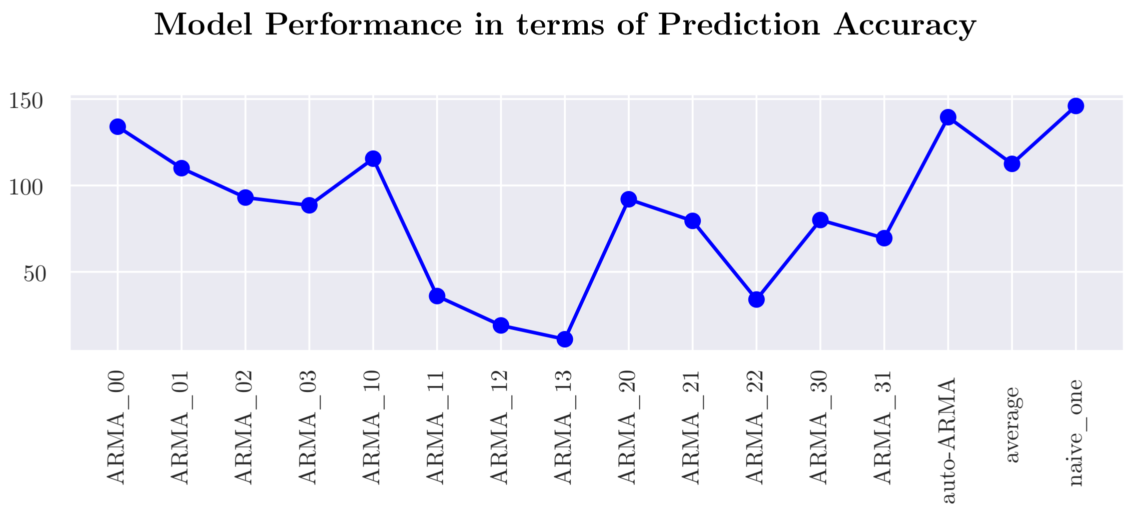


Figure 4.7: The top plots show the binary prediction accuracy reached by the various models for every single stock. Then the different models were given a rank corresponding to their binary prediction accuracy for the different stocks. This bottom plot shows the sum of the ranks achieved by the model across all stocks. As we look at prediction accuracy, a higher value is better.

5. Trading Strategies

The following chapter will give an overview over different trading strategies and compare their effectiveness with regards to trading with our ten stocks. While this an interesting exercise in and of itself, we also intended these trading strategies to serve as a baseline for the comparison with our prediction based trading strategy against. As we have seen in the last two chapters, however, generating predictions from machine learning and time series approaches has not shown to be effective. We will therefore simply explore some common strategies and see if they fare better. For simplicity, we have ignored trading costs.

5.1 Trading Strategies - Theoretical Background

5.1.1 Mean Reversion and Momentum Trading

A vast body of scientific literature has tried to develop models that allow to understand and explain movements in stock markets. An even vaster community of traders has tried to implement these theories to do actual forecasting. While many of the theories are much more complex, two basic ideas can be summarized as "Momentum Based Trading" and "Mean Reversion Based Trading". The former theory hypothesizes that stocks that do well now will likely continue to do so in the future. The latter states that especially good or past performances are an exception and that stocks will eventually return to their average performance. A third strategy, Pairs Trading, will be detailed later.

Mean-Reversion Trading
In 1985 Bondt and Thaler Bondt and Thaler (1985) were one of the early scholars to analyze mean reversion behaviour when they examined the hypothesis that markets tend to overreact. They looked at monthly returns of assets listed on the New York Stock Exchange in between 1926 and 1982 and constructed portfolios of winners and losers that were updated every three years. Winners (losers) were those stocks that had performed the best (worst) over the previous years. Their idea was that "if stock prices systematically overshoot, then their reversal should be predictable from past return data alone, with no use of any accounting data such as earnings. [...] Extreme movements in stock prices will be followed by subsequent price movements in the opposite direction" (Bondt and Thaler (1985)). Indeed they observed that the portfolio of losers outperformed the market significantly. While they focus on a very long time horizon of three years, (Jegadeesh and Titman, 1993) summarizes that this behaviour could also be observed in the shorter term: "[...] contrarian strategies that select stocks based on their returns in the previous week or month generate significant abnormal returns." Note that in their analysis, Bondt and Thaler assumed that the mean-reversion behaviour was due to the market overreacting, i.e. a market inefficiency. One can also look at mean-reversion from another angle: Economic theories like the famous model from Fama and French describe the return of the stocks of a company as the result of market properties like the baseline market return rate and inherent properties of that company, like for example its book-to-market-ratio (see Fama and French (1992) and Fama and French (1993)). If such a relationship exists than this

in turn implies that the actual observed stock movements should be random fluctuations around some much more slowly changing true return rate. However, it is unclear whether this mean reversion should happen in the form of a pendulum that swings forth and back (implying some short-term predictability) or more in the form of coin tosses reverting to their equilibrium by flooding past observations with new random ones (implying short-term non-predictability).

Momentum Trading
On the opposite side of the spectrum of trading strategies lies the idea of momentum. Jegadeesh and Titman (1993) were one of the first to describe "that strategies which buy stocks that have performed well in the past and sell stocks that have performed poorly in the past generate significant positive returns over 3- to 12-month holding periods." Interestingly, for the same portfolios they also saw mean-reverting behavior over the longer time frame of 12 to 48 months. Indeed, many scholars agree that mean reversion and momentum behaviour are not necessarily contradictions, but that the time frame determines in which way a stock will behave (see Balvers and Wu (2006)).

5.1.2 Pairs Trading

A third strategy we have examined is called pairs trading. The idea is to trade on the fact that some stocks usually move together. CocaCola and Pepsi for example sell a similar product and therefore should be affected similarly by events in the market. Assuming that the price difference between the two stocks should be somewhat constant, one can construct a trading strategy that profits if the spread randomly diverges too much and then converges again. While there are many ways to identify assets that move together (for an overview see e.g. Gatev et al. (1999), Ganapathy Vidyamurthy (2004), Do et al. (2006)) we rely on the idea of cointegration (introduced Engle and Granger (1987)) to find suitable stocks.
Cointegration means the following: If two time series x_t, y_t are both integrated of order d (i.e. they are stationary after applying d-th differences), then x_t, y_t are cointegrated if there exists a linear combination $ax_t + by_t$ that is integrated of order $d' < d$. Our stock prices (the log stock prices, to be more precise) are integrated of order one. One can therefore look for a time series $y_t - \beta x_t = u_t$ and check whether u_t is stationary. This leads to the two-step procedure developed by Engle and Granger (1987), where in the first step β is determined by least squares and in the second step u_t is tested for stationarity using the Augmented Dickey-Fuller test (ADF). Note that this procedure does not produce symmetric results. The result will differ slightly depending on the order (e.g. PG, JNJ or JNJ, PG).

5.2 Trading Strategies - Implementation

As detailed earlier, the success of a trading strategy may depend on the time frame over which it is applied. In this sense, the time frame is a hyperparameter to be optimized. However, in the following we will use a fixed time frame of one day for all strategies. This allows for direct comparisons and reduces the risk of overfitting. But above that, it was simply not feasible for us to do this hyperparameter optimization for every strategy.

[0]The Philipps-Ouiliaris integration test ((Phillips and Ouliaris, 1990)) may be slightly more appropriate, but we stick with the formerly described cointegration test, as it can be easily implemented in python

5.2.1 Mean Reversion Portfolio

We replicate the basic idea from Bondt and Thaler Bondt and Thaler (1985) and use the following trading algorithm:

Algorithm 1 - Mean-Reversion Portfolio

```
for day in all_days:
    rank portfolio along their returns from the last day
    Buy the two worst performers
    sell (short) the two best performers
```

While they found excessive returns of that strategy from 1932 to 1977 the results shown in Figure 5.1 cannot beat a simple buy-and-hold strategy. This may be due to the wrong time frame used, or to the fact that if the strategy had indeed once been profitable we would assume enough market participants to trade on it as to render it practically useless.

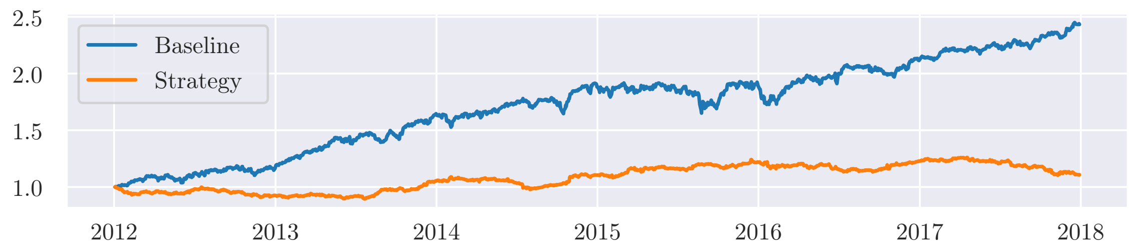


Figure 5.1: Performance of the mean-reversion portfolio (orange) against the returns of a portfolio that had once been bought in the beginning and kept unchanged throughout (buy-and-hold, blue).

5.2.2 Mean Reversion - single stocks

For single stocks, we have tried two different ways to implement a mean reversion strategy. The first strategy looks at whether the current stock price is above or below its past mean. The second one in spirit does the same thing, but with log-returns.

Mean Reversion - Stock Prices

The idea behind this strategy is that there is a true fundamental value underlying the stock that is represented by the mean of past stock prices. One naive implementation of the trading strategy, using the cumulative mean of the series, looks like this:

Algorithm 2: Mean-Reversion - Cumulative Mean

```
for day in all_days:
    cumulative_mean = cumulative mean of stock values until now
    if value of stock today > cumulative_mean:
        sell (short) the stock
    if value of stock today < cumulative_mean:
        buy the stock
```

One obvious problem with this strategy is that any trend in the time series will be very poorly captured. Figure 5.2 illustrates this by showing the time series plotted against their cumulative means (on the left) and the returns generated by trading on this strategy (on the right). One more sophisticaed approach to alleviate this problem is by comparing 30d and 60d averages. The strategy then looks like this:

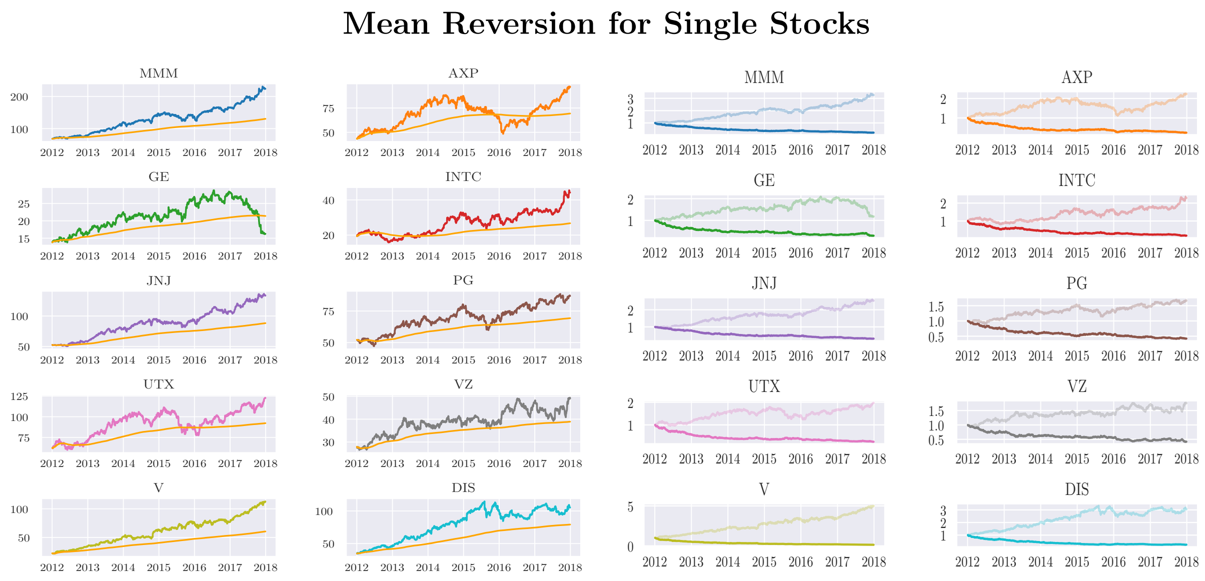


Figure 5.2: On the left the original time series are plotted against their cumulative average. On the right we see the returns from a simple buy-and-hold strategy (pale) against the returns from the trading strategy.

Algorithm 3: Mean-Reversion Moving Averages

```
for day in all_days:
    60d_mean = mean of stock values of the past 60 days
    30d_mean = mean of stock values of the past 30 days
    if 30d_mean > 60d_mean:
        sell (short) the stock
    if 30d_mean < 60d_mean:
        buy the stock
```

Figure 5.3 shows the 30d and 60d moving averages as well as the original time series on the left hand and the returns from the trading strategy on the right hand. While the plots on the left do look much better, the trading strategy is still not able to yield useful results.

Mean Reversion - Log-Returns

The second strategy looks at whether the current return is lower or higher than the cumulative mean of the time series. Analogous to algorithm 2 this algorithm just buys if the log-return of today are lower than the cumulative mean of log-returns. One can of course also easily replace the cumulative mean by e.g. the mean of the past 90, 60, or 30 days. The result for the cumulative mean is shown in Figure A.1 in the appendix. Again our strategy is not able to perform better than a simple buy-and-hold strategy.

5.2.3 Momentum Based Trading

Momentum based trading is essentially the reverse of mean-reversion trading (albeit time horizons are usually chosen differently). Our trading strategies stick with the time frame of 1 day. The momentum portfolio simply reverses algorithm 1. The strategy used for single stocks looks at whether the log-returns in time t are higher (lower) than the 30d mean of log-returns and use that as a buy (sell) signal. Results are shown in Figure A.2 in the appendix and are even less convincing than the results for mean-reversion trading.

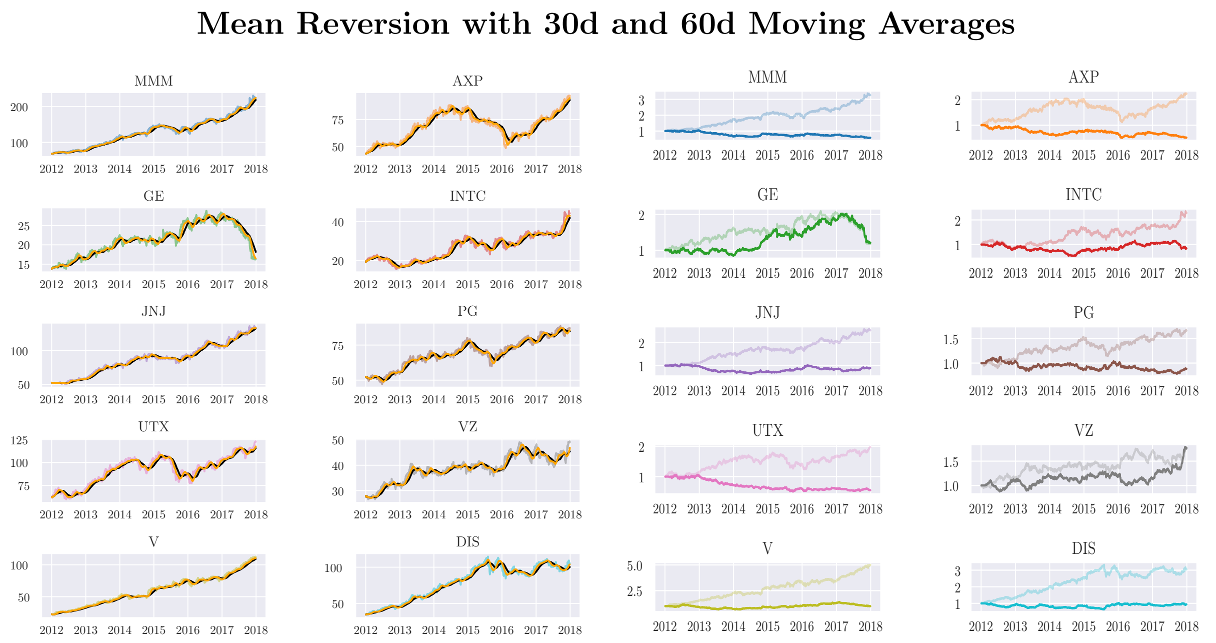


Figure 5.3: The left side shows the original timeline (pale), 60d moving averages (blue) and 30d moving averages (orange). The right side shows the returns generated by the strategy together with the returns from a simple buy-and-hold strategy (pale).

5.2.4 Pairs Trading

Identifiying Suitable Time Series

Many of our ten companies have overlapping areas of business (refer back to Table 2.1). Two obvious candidates are of course Visa (V) and American Express (AXP). United Technologies and General Electric are both producing electrical goods and Johnson&Johnson and Procter&Gamble both sell consumer goods. While this is somewhat problematic due to a multiple comparisons problem, we nevertheless checked all pairwise combinations of the ten stocks (log prices) for cointegration to see what comes out. From all tests, the pairs shown in Table 5.1 had a p-value smaller than 0.05.

companies	p-values	companies	p-values
JNJ, PG	0.0129	PG, JNJ	0.0083
JNJ, VZ	0.0466	VZ, JNJ	0.0118
VZ, MMM	0.0305		
VZ, PG	0.0393		
VZ, V	0.0266		
V, MMM	0.0415		

Table 5.1: Pairs of stocks tested for cointegration with a p-value < 0.05

We went for Procter&Gamble and Johnson&Johnson, as the p-values were smallest, the cointegration worked in both directions and the firms indeed operate in similar business areas. The left of Figure 5.4 shows the two time series. Note that even though the cointegration test was significant, we see that the two stocks slowly diverge in value. To see whether the strategy works nevertheless, we apply the following basic pairs trading algorithm [1]:

Algorithm 4 - Pairs Trading

```
for day in all_days:
    diff = stock price PG - stock price JNJ
    mu = 90d average stock price PG - 90d average stock price JNJ
    sd = standard devation of diff until day
    zscore = (diff - mu) / sd
    if zscore > 0:
        sell (short) PG
        buy JNJ
    else:
        buy PG
        sell (short) JNJ
```

The right side of Figure 5.4 shows the performance of the trading strategy on the sample. We can see that pairs trading, at least in the way we implemented it, did not work out.

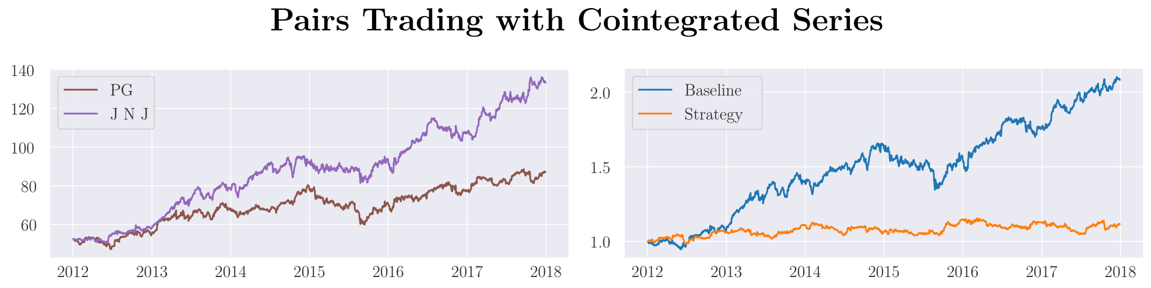


Figure 5.4: On the left the stock prices of Procter&Gamble and Johnson&Johnson are plotted over time. On the right the profits from applying a pairs trading strategy are shown.

[1]Note that dividing by the standard deviation only has an effect if one sets specific thresholds other than > 0, i.e. to trigger trades or stop-loss orders. The standard deviation is included to illustrate how the algorithm can be adapted to accommodate more complex needs.

6. Conclusion

In this report we have made a sincere attempt to predict stocks by using text analysis, machine learning and time series. We have firstly looked at two different ways to generate sentiments ourselves from any body of text data, the Bag of Words methods and the Joint Sentiment Topic method. Because of the limitations imposed by the analyst reports data set, we also worked with the much richer data set of precomputed sentiment scores from RavenPack. We secondly tried to use those two data sets in conjuncture with the machine learning method XGBoost to generate stock predictions. Thirdly, we examined a variety of different time series models and applied them to the stock data. We then used those model to generate predictions and compared them against different baselines. Lastly, we looked at a whole range of theory guided trading and compared their effectiveness on our data set.
While we were ultimately (and unsurprisingly) not successful in predicting future stocks returns, we still made some interesting findings. First, we learned that a custom sentiment analysis using the Bag of Words is possible, but somewhat difficult, because it depends on good sentiment libraries that are often not freely available. Secondly, JST is a very interesting and powerful model, yet the results are very hard to interpret. Thirdly we learned that the Analyst Reports don't contain information useful for forecasting that we were able to extract. In light of the efficient market hypothesis, this is not surprising, as the reports usually don't contain new information that has not been priced in at the time of writing of the report. We cannot, however, explicitly assess the usefulness of the recommendations made in the reports, because these have been meshed up with everything else in the report during preprocessing. We fourthly learned that the RavenPack sentiments do seem to contain information that is new and relevant to the market. This of course does not necessarily mean that the underlying news articles themselves influence the stock directly. It does, however, mean that the news collected by RavenPack are informed by events that really do influence the market. We were unfortunately not able to exploit this information, as we only have daily closing prices. By the time the market closes, all information has already been priced in and the RavenPack sentiments do not inform future returns. If one had access to the information and sentiment scores in real time, however, it might be possible to use that information for forecasting, even with the methods we have employed in this report. Fifthly, we learned that financial time series often look like they exhibit significant autocorrelation, but that these patterns may be due to random chance and do not allow reliably predicting future returns. Lastly, we saw that other trading strategies were also not successful in reliably generating excess returns - at least not in the way we implemented them. All of this does not definitively mean that predicting stocks is impossible. It is merely a lot harder than many popular internet sites and forums suggest. One can think of an intra day trader who is quicker than others in adapting to new information. One can also conceive that the trading strategies we discussed may work with some tweaking and over different time frames. This, however, is left for future work.

Bibliography

Robert H. Shumway and David S. Stoffer. *Time series analysis and its applications: with R examples.* Springer texts in statistics. Springer, New York, 3rd ed edition, 2011. ISBN 978-1-4419-7864-6.

Said E. Said and David A. Dickey. Testing for unit roots in autoregressive-moving average models of unknown order. *Biometrika*, 71(3):599–607, December 1984. ISSN 0006-3444. doi: 10.1093/biomet/71.3.599. URL `https://academic.oup.com/biomet/article/71/3/599/258758`.

Robert F. Engle. Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica*, 50(4):987, July 1982. ISSN 00129682. doi: 10.2307/1912773. URL `https://www.jstor.org/stable/1912773?origin=crossref`.

Ravenpack. `https://www.ravenpack.com/`. Accessed: 2019-08-30.

Emma Haddi, Liu Xiaohui, and Shi Yong. The role of text pre-processing in sentiment analysis. *Procedia Computer Science*, 17:26 – 32, 2013. ISSN 1877-0509. First International Conference on Information Technology and Quantitative Management.

Calum S Robertson, Shlomo Geva, and Rodney C Wolff. News aware volatility forecasting: Is the content of news important? In *Proceedings of the sixth Australasian conference on Data mining and analytics-Volume 70*, pages 161–170. Australian Computer Society, Inc., 2007.

Adam Westerski. Sentiment analysis: Introduction and the state of the art overview. *Universidad Politecnica de Madrid, Spain*, pages 211–218, 2007.

Tim Loughran and Bill McDonald. Textual analysis in accounting and finance: A survey. *Journal of Accounting Research*, 54(4):1187–1230, 2016.

Chenghua Lin and Yulan He. Joint sentiment/topic model for sentiment analysis. pages 375–384, 2009.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2017. URL `https://www.R-project.org/`.

Julia Silge and David Robinson. tidytext: Text mining and analysis using tidy data principles in r. *JOSS*, 1(3), 2016. doi: 10.21105/joss.00037. URL `http://dx.doi.org/10.21105/joss.00037`.

Jin-Cheon Na, Haiyang Sui, Christopher SG Khoo, Syin Chan, and Yunyun Zhou. Effectiveness of simple linguistic processing in automatic sentiment classification of product reviews. 2004.

Tyler W. Rinker. *textstem: Tools for stemming and lemmatizing text*. Buffalo, New York, 2018. URL `http://github.com/trinker/textstem`. version 0.1.4.

Kenneth Benoit, Kohei Watanabe, Haiyan Wang, Paul Nulty, Adam Obeng, Stefan Müller, and Akitaka Matsuo. quanteda: An r package for the quantitative analysis of textual data. *Journal of Open Source Software*, 3(30):774, 2018. doi: 10.21105/joss.00774. URL `https://quanteda.io`.

Max Boiten. *rJST: Joint Sentiment Topic Modelling*. Buffalo, New York, 2018. URL `https://github.com/maxboiten/rJST`. version 1.

Chuanbin Li, Xiaosen Zheng, Zikun Yang, and Li Kuang. Predicting short-term electricity demand by combining the advantages of arma and xgboost in fog computing environment. *Wireless Communications and Mobile Computing*, 2018, 2018.

Tianqi Chen and Carlos Guestrin. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 2016.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei James Chen, Weidong Ma, Qiwei Ye, and T. M. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *NIPS*, 2017.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Adam Atkins, Mahesan Niranjan, and Enrico Gerding. Financial news predicts stock market volatility better than close price. *The Journal of Finance and Data Science*, 4 (2):120–137, 2018.

Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, April 1986. ISSN 0304-4076. doi: 10.1016/0304-4076(86)90063-1. URL `http://www.sciencedirect.com/science/article/pii/0304407686900631`.

Lawrence R. Glosten, Ravi Jagannathan, and David E. Runkle. On the Relation between the Expected Value and the Volatility of the Nominal Excess Return on Stocks. *The Journal of Finance*, 48(5):1779–1801, December 1993. ISSN 00221082. doi: 10.1111/j.1540-6261.1993.tb05128.x. URL `http://doi.wiley.com/10.1111/j.1540-6261.1993.tb05128.x`.

G. M. Ljung and G. E. P. Box. On a measure of lack of fit in time series models. *Biometrika*, 65(2):297–303, August 1978. ISSN 0006-3444. doi: 10.1093/biomet/65.2.297. URL `https://academic.oup.com/biomet/article/65/2/297/236869`.

StatsModels: Statistics in Python — statsmodels v0.10.1 documentation. URL `https://www.statsmodels.org/stable/index.html`.

Gyozo Gidofalvi and Charles Elkan. Using news articles to predict stock price movements. *Department of Computer Science and Engineering, University of California, San Diego*, 2001.

John H. H. Lee. A Lagrange multiplier test for GARCH models. *Economics Letters*, 37(3):265–271, November 1991. ISSN 0165-1765. doi: 10.1016/0165-1765(91)90221-6. URL `http://www.sciencedirect.com/science/article/pii/0165176591902216`.

Peter R. Hansen and Asger Lunde. A forecast comparison of volatility models: does anything beat a GARCH(1,1)? *Journal of Applied Econometrics*, 20(7):873–889, 2005. ISSN 1099-1255. doi: 10.1002/jae.800. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/jae.800`.

alexios. Does anything NOT beat the GARCH(1,1)?, January 2013. URL `http://www.unstarched.net/2013/01/07/does-anything-not-beat-the-garch11/`.

Erhard Reschenhofer. Does Anyone Need a GARCH(1,1)? *Journal of Finance and Accounting*, 1(2):48–53, January 2013. doi: 10.12691/jfa-1-2-2. URL `http://pubs.sciepub.com/jfa/1/2/2/abstract.html`.

WERNER F. M. De Bondt and Richard Thaler. Does the Stock Market Overreact? *The Journal of Finance*, 40(3):793–805, 1985. ISSN 1540-6261. doi: 10.1111/j.1540-6261.1985.tb05004.x. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1985.tb05004.x`.

Narasimhan Jegadeesh and Sheridan Titman. Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency. *The Journal of Finance*, 48(1):65–91, March 1993. ISSN 00221082. doi: 10.1111/j.1540-6261.1993.tb04702.x. URL `http://doi.wiley.com/10.1111/j.1540-6261.1993.tb04702.x`.

Eugene F. Fama and Kenneth R. French. The Cross-Section of Expected Stock Returns. *The Journal of Finance*, 47(2):427–465, 1992. ISSN 1540-6261. doi: 10.1111/j.1540-6261.1992.tb04398.x. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1992.tb04398.x`.

Eugene F. Fama and Kenneth R. French. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33(1):3–56, February 1993. ISSN 0304-405X. doi: 10.1016/0304-405X(93)90023-5. URL `http://www.sciencedirect.com/science/article/pii/0304405X93900235`.

Ronald J. Balvers and Yangru Wu. Momentum and mean reversion across national equity markets. *Journal of Empirical Finance*, 13(1):24–48, January 2006. ISSN 0927-5398. doi: 10.1016/j.jempfin.2005.05.001. URL `http://www.sciencedirect.com/science/article/pii/S0927539805000708`.

Evan Gatev, William N. Goetzmann, and K. Geert Rouwenhorst. Pairs Trading: Performance of a Relative Value Arbitrage Rule. 1999. doi: 10.1093/rfs/hhj020.

Ganapathy Vidyamurthy. Pairs Trading: Quantitative Methods and Analysis | Finance & Investments Special Topics | General Finance & Investments | Subjects | Wiley, 2004. URL `https://www.wiley.com/en-us/Pairs+Trading%3A+Quantitative+Methods+and+Analysis-p-9780471460671`.

Binh Do, Robert Faff, and Kais Hamza. A New Approach to Modeling and Estimation for Pairs Trading. *2006*, page 31, 2006.

Robert F. Engle and C. W. J. Granger. Co-Integration and Error Correction: Representation, Estimation, and Testing. *Econometrica*, 55(2):251–276, 1987. ISSN 0012-9682. doi: 10.2307/1913236. URL `https://www.jstor.org/stable/1913236`.

P. C. B. Phillips and S. Ouliaris. Asymptotic Properties of Residual Based Tests for Cointegration. *Econometrica*, 58(1):165–193, 1990. ISSN 0012-9682. doi: 10.2307/2938339. URL `https://www.jstor.org/stable/2938339`.

A. Appendix

A.1 Appendix to Chapter 3

	topic1sent2	topic2sent2	topic3sent2	topic30sent2
1	risk	market	research	opinion
2	price	capital	investment	information
3	industry	research	analyst	analysis
4	target	bank	financial	otherwise
5	factor	report	company	herein
6	company	investment	limit	write
7	include	rate	affiliate	investment
8	earnings	client	issuer	datum
9	multiple	affiliate	investor	without
10	valuation	issuer	relevant	call
11	market	service	relate	provide
12	specific	subject	service	value
13	condition	analyst	information	advice
14	upon	financial	month	unless
15	subject	distribution	client	decision

Table A.1: Sentiment 2 for the topics 1, 2, 3, 30 generated by the JST.

A.2 Appendix to Chapter 4

Full ARMAX(0,0) with Ravenpack sentiments fit to the log-returns of Visa

Dep. Variable:	log_returns	**No. Observations:**	1208
Model:	ARMA(0, 0)	**Log Likelihood**	3519.897
Method:	css	**S.D. of innovations**	0.013
Date:	Sat, 14 Sep 2019	**AIC**	-7019.795
Time:	16:11:38	**BIC**	-6968.828
Sample:	0	**HQIC**	-7000.602

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
const	0.0019	0.016	0.115	0.908	-0.030	0.034
rel_mean	-9.181e-05	0.000	-0.565	0.572	-0.000	0.000
ess_mean	9.442e-05	0.000	0.867	0.386	-0.000	0.000
ens_mean	-8.462e-06	2.23e-05	-0.380	0.704	-5.22e-05	3.52e-05
aes_mean	5.237e-05	5.28e-05	0.992	0.321	-5.11e-05	0.000
aev_mean	-7.829e-07	2.78e-06	-0.282	0.778	-6.23e-06	4.66e-06
ess_min	4.855e-05	6.43e-05	0.755	0.450	-7.74e-05	0.000
ess_max	6.83e-06	6.42e-05	0.106	0.915	-0.000	0.000
aes	-4.069e-05	3.58e-05	-1.136	0.256	-0.000	2.95e-05

Full ARMAX(0,0) with Ravenpack sentiments fit to the log-returns of INTC

Dep. Variable:	log_returns	**No. Observations:**	1434
Model:	ARMA(0, 0)	**Log Likelihood**	4143.812
Method:	css	**S.D. of innovations**	0.013
Date:	Sat, 14 Sep 2019	**AIC**	-8267.625
Time:	16:11:38	**BIC**	-8214.942
Sample:	0	**HQIC**	-8247.954

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
const	-0.0424	0.020	-2.070	0.039	-0.082	-0.002
rel_mean	0.0003	0.000	1.630	0.103	-7e-05	0.001
ess_mean	0.0003	8.51e-05	3.110	0.002	9.78e-05	0.000
ens_mean	8.494e-06	1.67e-05	0.510	0.610	-2.42e-05	4.12e-05
aes_mean	-3.095e-05	6.04e-05	-0.513	0.608	-0.000	8.74e-05
aev_mean	-2.858e-06	1.75e-06	-1.633	0.103	-6.29e-06	5.72e-07
ess_min	-9.322e-06	4.59e-05	-0.203	0.839	-9.93e-05	8.07e-05
ess_max	-5.201e-05	5.42e-05	-0.960	0.337	-0.000	5.42e-05
aes	3.953e-05	5.22e-05	0.758	0.449	-6.27e-05	0.000

Table A.2: Results for ARMAX(0,0), i.e. a regression with a constant and the Ravenpack sentiment data as external regressors.

Results for GARCH(1,1) with t-Distribution (V)

Dep. Variable:	log_returns	**R-squared:**	-0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	-0.000
Vol Model:	GARCH	**Log-Likelihood:**	-2379.16
Distribution:	Standardized Student's t	**AIC:**	4768.31
Method:	Maximum Likelihood	**BIC:**	4794.91
		No. Observations:	1509
Date:	Wed, Sep 04 2019	**Df Residuals:**	1504

	coef	**std err**	**t**	**P> |t|**	**95.0% Conf. Int.**
mu	0.1150	2.551e-02	4.508	6.559e-06	[6.498e-02, 0.165]
	coef	**std err**	**t**	**P> |t|**	**95.0% Conf. Int.**
omega	0.1058	6.085e-02	1.739	8.196e-02	[-1.342e-02, 0.225]
alpha[1]	0.1297	4.143e-02	3.131	1.745e-03	[4.849e-02, 0.211]
beta[1]	0.8158	6.683e-02	12.208	2.823e-34	[0.685, 0.947]
	coef	**std err**	**t**	**P> |t|**	**95.0% Conf. Int.**
nu	4.9106	0.617	7.952	1.828e-15	[3.700, 6.121]

Covariance estimator: robust

Results for GARCH(1,1) with t-Distribution (INTC)

Dep. Variable:	log_returns	**R-squared:**	-0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	-0.000
Vol Model:	GARCH	**Log-Likelihood:**	-2474.91
Distribution:	Standardized Student's t	**AIC:**	4959.82
Method:	Maximum Likelihood	**BIC:**	4986.41
		No. Observations:	1509
Date:	Wed, Sep 04 2019	**Df Residuals:**	1504

	coef	**std err**	**t**	**P> |t|**	**95.0% Conf. Int.**
mu	0.0810	2.881e-02	2.810	4.950e-03	[2.449e-02, 0.137]
	coef	**std err**	**t**	**P> |t|**	**95.0% Conf. Int.**
omega	7.6664e-03	6.416e-03	1.195	0.232	[-4.909e-03,2.024e-02]
alpha[1]	0.0173	4.803e-03	3.604	3.137e-04	[7.896e-03,2.672e-02]
beta[1]	0.9792	5.563e-03	176.017	0.000	[0.968, 0.990]
	coef	**std err**	**t**	**P> |t|**	**95.0% Conf. Int.**
nu	4.1875	0.471	8.896	5.802e-19	[3.265, 5.110]

Covariance estimator: robust

Table A.3

Results for GJR-GARCH(1,1) with t-Distribution (Visa)

Dep. Variable:	log_returns	**R-squared:**	-0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	-0.000
Vol Model:	GJR-GARCH	**Log-Likelihood:**	-2370.42
Distribution:	Standardized Student's t	**AIC:**	4752.84
Method:	Maximum Likelihood	**BIC:**	4784.75
		No. Observations:	1509
Date:	Wed, Sep 04 2019	**Df Residuals:**	1503

	coef	**std err**	**t**	**P> |t|**	**95.0% Conf. Int.**
mu	0.0962	2.631e-02	3.657	2.549e-04	[4.465e-02, 0.148]

	coef	**std err**	**t**	**P> |t|**	**95.0% Conf. Int.**
omega	0.0857	5.488e-02	1.561	0.118	[-2.187e-02, 0.193]
alpha[1]	0.0347	2.612e-02	1.327	0.185	[-1.653e-02,8.585e-02]
gamma[1]	0.1630	5.265e-02	3.096	1.961e-03	[5.981e-02, 0.266]
beta[1]	0.8411	6.446e-02	13.050	6.383e-39	[0.715, 0.967]

	coef	**std err**	**t**	**P> |t|**	**95.0% Conf. Int.**
nu	5.1057	0.675	7.568	3.795e-14	[3.783, 6.428]

Covariance estimator: robust

Results for GJR-GARCH(1,1) with t-Distribution (INTC)

Dep. Variable:	log_returns	**R-squared:**	-0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	-0.000
Vol Model:	GJR-GARCH	**Log-Likelihood:**	-2474.66
Distribution:	Standardized Student's t	**AIC:**	4961.31
Method:	Maximum Likelihood	**BIC:**	4993.23
		No. Observations:	1509
Date:	Wed, Sep 04 2019	**Df Residuals:**	1503

	coef	**std err**	**t**	**P> |t|**	**95.0% Conf. Int.**
mu	0.0808	2.862e-02	2.822	4.777e-03	[2.466e-02, 0.137]

	coef	**std err**	**t**	**P> |t|**	**95.0% Conf. Int.**
omega	0.0152	4.217e-02	0.360	0.719	[-6.748e-02,9.783e-02]
alpha[1]	0.0146	9.131e-03	1.601	0.109	[-3.276e-03,3.252e-02]
gamma[1]	0.0133	5.218e-02	0.255	0.799	[-8.898e-02, 0.116]
beta[1]	0.9712	4.137e-02	23.476	7.225e-122	[0.890, 1.052]

	coef	**std err**	**t**	**P> |t|**	**95.0% Conf. Int.**
nu	4.1624	0.473	8.808	1.272e-18	[3.236, 5.089]

Covariance estimator: robust

Table A.4

A.3 Appendix to Chapter 5

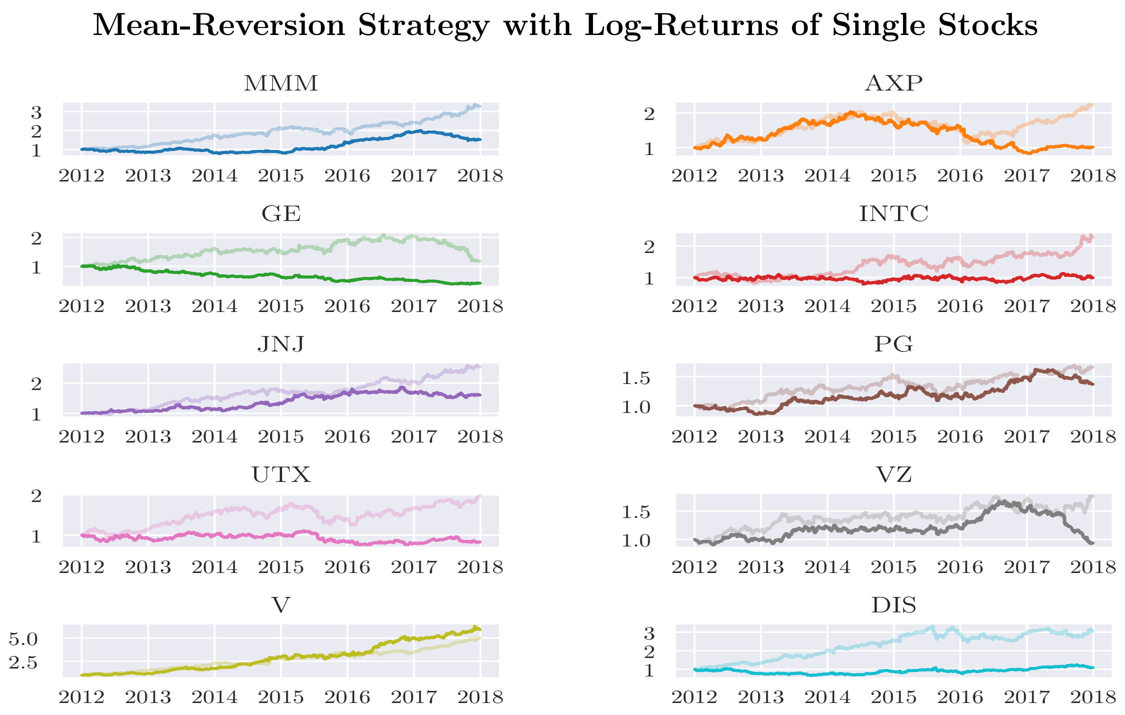


Figure A.1: Mean Reversion based on past returns of single stocks. The returns achieved by the strategy are in bold color, while the return of a simple buy-and-hold strategy, (i.e. the return from a single share bought on the first day) is shown in pale

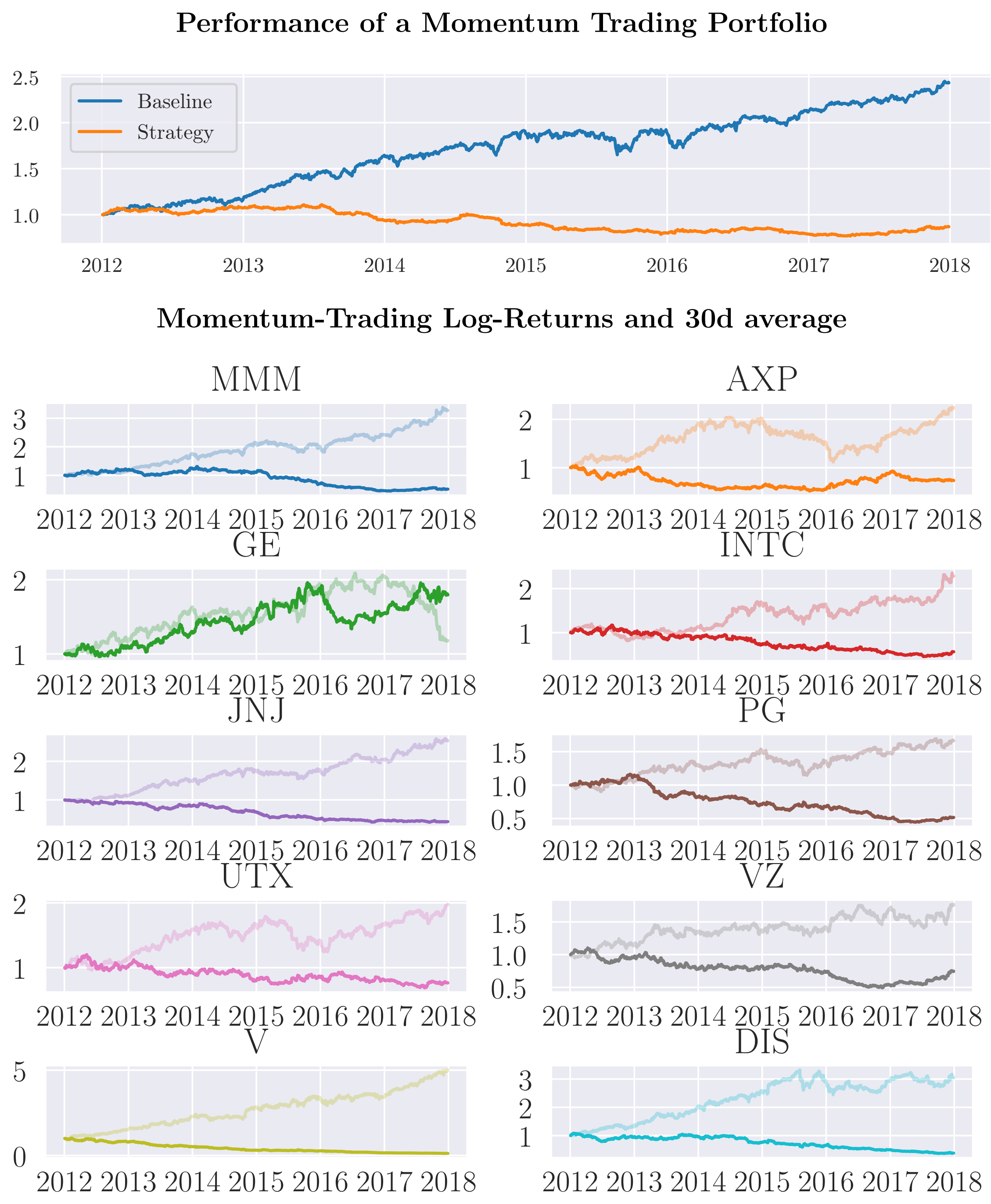


Figure A.2: Momentum-Trading for portfolio (top) and for log-returns of single stocks (bottom). The returns achieved by the strategy are in bold color, while the return of a simple buy-and-hold strategy, (i.e. the return from a single share bought on the first day) is shown in pale

Statutory Declaration

We declare that we have authored this thesis independently, that we have not used other than the declared sources / resources, and that we have explicitly marked all material which has been quoted either literally or by content from the used sources.

Georg-August-University Göttingen, September 15, 2019

Nikos Bosse
<nikos.bosse@stud.uni-goettingen.de>

Felix Süttmann
<felix.suettmann@stud.uni-goettingen.de>