



Forecasting German Government Bond Development by (Deep) Neural Networks on Technical and Economic Data

Master Thesis

Author: Hans Christian Schmitz
Student number: 6025668
h.c.schmitz@students.uu.nl
Universiteit Utrecht

Advisor: Prof. Dr. Arno Siebes
Second Advisor: Dr. Ad Feelders
Universiteit Utrecht
Algorithmic Data Analysis

Daily Advisors: {Prof. Dr. Damian Borth, Marco Schreyer}
German Research Center for Artificial Intelligence (DFKI)
Multimedia Analysis & Data Mining
University of St. Gallen
Artificial Intelligence & Machine Learning

October 2018

Abstract

Artificial Intelligence is on its way to change many aspects of every-day-life. One often underestimated industry, where this change happens, is the financial industry. Much work in the area of Artificial Intelligence and Finance is concerned with time series forecasting. One specific and economically important type of time series is the development of government bonds prices over time. This thesis presents an overview of state-of-the art forecasting techniques on government bond prices and compares the established techniques with a newly developed, long short term memory recurrent neural network based technique for bond price forecasting. Initial results show that neural network based approaches can outperform other established techniques. However, further research in this direction needs to be conducted.

Acknowledgements

For the realization of this thesis, I would like to gratefully acknowledge the support of the following people.

My first university advisor Arno Siebes, thank you for your extended feedback on the draft versions of all the parts of this thesis, enriching discussions over Skype and in your office, the extraordinary support also in organizational matters, and, maybe most importantly, for enabling me to conduct this research project at the DFKI in Kaiserslautern.

My daily advisor Damian Borth, thank you for facilitating me at the DFKI, being positive about this project from the very beginning, when I contacted you out of the blue, for your helpful and sharp comments on my work at DFKI and your insights regarding financial data science.

My second university advisor Ad Feelders, who agreed on being the second advisor for this research project of Utrecht University.

My daily advisor Marco Schreyer, thank you for helping to get up to speed in the beginning of the thesis project, very to-the-point feedback on many aspects of this work, for always being around when any question came up (and always having an answer or knowing where to search for a solution), and a vast variety of greatly helpful ideas, and for - not to be forgotten - the great company in the lunch breaks in Kaiserslautern.

Thank you to everyone else apart from the already mentioned from the German Research Centre for Artificial Intelligence, especially the Multimedia and Data Mining group.

And finally, special thanks go to my girlfriend Lea and my family, thank you for your infinite support, love and willingness to accommodate me on my many travels between Utrecht and the DFKI.

Contents

1	Introduction	1
1.1	Motivation	1
1.1.1	Applying Deep Learning Methods to Financial Time Series Data	1
1.1.2	Applying Deep Learning to Bond Price Forecasting	3
1.2	Thesis Objectives	4
1.3	Thesis Outline	5
2	Related Work	7
2.1	Introduction to Deep Learning	7
2.1.1	Recent Success of Deep Learning	8
2.1.2	Fundamentals of Deep Learning	8
2.1.3	Deep Feedforward Networks	10
2.2	Deep Learning for Time Series Analysis	18
2.2.1	Time Series Analysis	18
2.2.2	Recurrent Neural Networks	20
2.2.3	Long Short Term Memory	22
2.3	Financial Data Forecasting	25
2.3.1	Stock Price Forecasting	25
2.3.2	Bond Price Forecasting	29
2.4	Effects of Economic Indicators on Government Bond Prices	31
2.4.1	Federal or Government Bonds	31
2.4.2	Bond Price Influencing Indicators as Found in Related Work .	32
3	German Federal Bond (Bund) Data	36
3.1	Choice for Bunds as Government Bonds	36
3.2	Bund Data	36
3.3	Exploratory Analysis of Bund Data	37
4	Macro-Economic Data	42
4.1	Economic Indicators with Relevance for Germany	42
4.2	Feature Engineering: Economic Indicators as Additional Features . .	44
4.2.1	Economic Indicators and Bond Price Correlations	44
4.2.2	Correlations between Economic Indicators	48
4.2.3	Ranking the Relevance of Economic Indicators	49

5	Experimental Setup	53
5.1	Training and Test Data	54
5.1.1	Bund Data	54
5.1.2	Macro-Economic Data	58
5.1.3	Fusion of Bund Data and Economic Data	59
5.2	Different Forecasting Setups	60
5.2.1	Forecasting with Different Time Horizons	60
5.2.2	Forecasting with Different Target	61
5.2.3	Forecasting on Original and Fused Data	61
5.3	Models for Forecasts	61
5.3.1	Naive	61
5.3.2	Median and Mean	62
5.3.3	Linear Regression	62
5.3.4	ARIMA	63
5.3.5	Multi-Layer-Perceptron Regressor	66
5.3.6	LSTM	69
5.4	Features for different models	74
5.5	Generalizability of different models	74
5.6	Rolling Forecast vs Classical Forecast	75
5.7	Evaluating model performance	77
6	Experimental Results	79
6.1	Next Day Forecast: Comparison of Model Classes	79
6.1.1	Next Day Price Forecast: Best Performing Model Configurations of Each Class	80
6.1.2	Next Day Return Forecast: Best Performing Model Configurations of Each Class	81
6.2	Next Week Forecast: Comparison of Model Classes	83
6.2.1	Next Week Price Forecast: Best Performing Model Configurations of Each Class	83
6.2.2	Next Week Return Forecast: Best Performing Model Configurations of Each Class	85
7	Discussion	91
7.1	Evaluating Introductory Hypotheses	91
7.1.1	Hypothesis 1	91
7.1.2	Hypothesis 2	93

7.1.3	Hypothesis 3	94
7.1.4	Hypothesis 4	95
7.2	Open Questions	97
8	Conclusion	102
9	References	106
10	Appendix	113
10.1	Full Experimental Results: Next Day Forecasting	113
10.1.1	Comparison of Less Complex Models	114
10.1.2	Comparison of Performance of ARIMA Models	116
10.1.3	Comparison of Performance of MLP Models, Trained on All Bunds	117
10.1.4	Comparison of Performance of MLP Models, Trained on Each Bund	123
10.1.5	Comparison of Best Performance of LSTM Models, Trained on All Bunds	128
10.1.6	Comparison of Best Performance of LSTM Models, Trained on Each Bund	132
10.2	Full Experimental Results: Next Week Forecasting	134
10.2.1	Comparison of Less Complex Models	136
10.2.2	Comparison of Performance of ARIMA Models	138
10.2.3	Comparison of Performance of MLP Models, Trained on All Bunds	139
10.2.4	Comparison of Performance of MLP Models, Trained on Each Bund	144
10.2.5	Comparison of Performance of LSTM Models, Trained on All Bunds	148
10.2.6	Comparison of Performance of LSTM Models, Trained on Each Bunds	152
10.3	Development of R-Shiny-App	156
10.3.1	Motivation	156
10.3.2	Components	156
10.4	MAPE per Model for a Selection of 10 Bunds	162

List of Figures

1	Sample LSTM cell, figure from Olah [52]	2
2	US-Dollar denominated credit risk to non-banks outside the United States according to the International Monetary Fund ([65]). X-axis depicting years from 2000 until 2018. Y-axis referring credit outstanding in trillion USD.	4
3	Rule based, classical machine learning and representation learning as found in Goodfellow et al. ([25]).	9
4	Illustration of a deep learning model as found in Goodfellow et al. ([25]).	10
5	Comparison of common output units.	13
6	Comparison of activation functions (input $z \in [3, 3]$).	14
7	Example: computational graph ([46]).	17
8	Example: cyclic computational graph, black box indicates one step in the sequence ([25]).	20
9	Example: unfolding of a cyclic computational graph ([25])	21
10	Simple RNN, figure from Olah ([52]).	22
11	LSTM cell, figure from Olah ([52]).	22
12	Cell state, figure from Olah ([52]).	23
13	Forget layer, figure from Olah ([52]).	23
14	Input gate layer, figure from Olah ([52]).	24
15	Update layer, figure from Olah ([52])	24
16	Output layer, figure from Olah ([52]).	24
17	Shares by security of total German debt ([18]).	37
18	Bund 10 and 30 from 3rd Jan 2011 to 15th Feb 2018.	39
19	Active Bunds per year.	40
20	Issued/expired Bunds per year.	40
21	Share of 10 and 30 year Bunds over time span.	41
22	Correlations between top 10 overall, top 10 short term and top 10 long term economic indicators.	48
23	Development of four selected indicators over Bund data time span.	51
24	Development of seven selected indicators over Bund data time span.	52
25	Prices: Bund 10 and 30 from 3rd Jan 2011 to 15th Feb 2018.	56
26	Returns: Bund 10 and 30 from 3rd Jan 2011 to 15th Feb 2018, scaled to range (0,2).	57
27	Autocorrelation for ARIMA Parameter Selection by Box-Jenkins.	64

28	Partial autocorrelation of training data, Bund prices for ARIMA parameter selection.	65
29	Partial autocorrelation of training data, Bund returns.	66
30	Representative example for one selected MLP architecture: losses per epoch, training on each Bund, different colors indicate different models.	68
31	Representative example for one selected MLP architecture: losses per epoch, training on all Bunds.	68
32	Overview of MLP architectures, one to three hidden layers.	69
33	Overview of LSTM architectures, one to three hidden LSTM layers.	71
34	Representative example for one selected LSTM architecture: losses per epoch, training on each Bund, different colors indicate different models.	72
35	Representative example for one selected LSTM architecture: losses per epoch, training on all Bunds.	73
36	Visualization of time series transformation for LSTM input.	74
37	Visualization of rolling forecast method.	76
38	Next day price result visualization of the two best performing models on a randomly selected ISIN, limited to the first 50 days of the test data set.	82
39	Next day price result visualization, plot of x vs \hat{x}	83
40	Next day return result visualization of the two best performing models on a randomly selected ISIN, limited to the first 50 days of the test data set.	85
41	Next day return result visualization, plot of x vs \hat{x}	86
42	Next week price result visualization of the two best performing models on a randomly selected ISIN, limited to the first 50 days of the test data set.	88
43	Next week price result visualization, plot of x vs \hat{x}	88
44	Next week return result visualization of the two best performing models on a randomly selected ISIN, limited to the first 50 days of the test data set.	89
45	Next week return result visualization, plot of x vs \hat{x}	90
46	Mape per ISIN per Model for next day price forecasts.	92
47	Mape per ISIN per Model for next day return forecasts.	96
48	Plot of variance vs mape per model of next day price forecasts.	98
49	Plot of variance vs mape per model of next day return forecasts.	99

50	Longer Training does not improve LSTM model performance, trained on <i>all</i> Bunds, forecasting next day return.	101
51	Start Page Shiny App	157
52	Shiny App Navigation Bar	158
53	Shiny App Choosing Forecast Horizon	158
54	Shiny App Choosing Bund	159
55	Shiny App Choosing Model	159
56	Shiny App Choosing Days Range	159
57	Shiny App Export Function	160
58	Shiny App Price Graph	160
59	Shiny App Return Graph	161
60	Shiny App Current Performance	161
61	Mape per ISIN per Model for next week price forecasts	162
62	Mape per ISIN per Model for next week return forecasts	163

List of Tables

1	Reuters top economic indicators for Germany.	43
2	Average absolute overall correlations between Bund price and economic indicators (identified by RIC=Reuters Instrument Code). . . .	45
3	Average absolute short term correlations between Bund price and economic indicators (identified by RIC=Reuters Instrument Code, if not stock index such as DAX, NASDAQ or STOXX).	46
4	Average absolute long term correlations between Bund price and economic indicators (identified by RIC=Reuters Instrument Code). . . .	47
5	Selection of four promising features for Bund price forecasting.	50
6	Selection of seven promising features for Bund price forecasting. . . .	51
7	Additional information on training and test data.	55
8	ARIMA configurations for prices and return forecasts.	66
9	Next day price forecast, best performing model configurations per class. Notations as in chapter Experimental Results (6).	81
10	Next day return forecast, best performing model configurations per class. Notations as in chapter Experimental Results (6).	84
11	Next week price forecast, best performing model configurations per class. Notations as in chapter Experimental Results (6).	87
12	Next week return forecast, best performing model configurations per class. Notations as in chapter Experimental Results (6).	89
13	Next day price forecast, normalization instead of scaling, trained on <i>all</i> Bunds. Notations as in chapter Experimental Results (6).	100
14	Results of next week price forecast, extract of table 11. Best performing LSTM, MLP and Average with respect to the median of the mean absolute percentage error over all Bunds are reported. Standard deviation of mape scores over all Bunds per model is listed as well.	103
15	Next day price forecast results of less complex models	115
16	Next day return forecast results of less complex	115
17	Next day price forecast results of arima models	116
18	Next day return forecast results of arima models	117
19	Next day rice forecast of MLP models, all Bunds	118
20	Different random initializations for next day price MLP models, trained on all Bunds	118

21	Next day price forecast of MLP models, trained on all Bunds and fused data	119
22	Different random initializations for next day price forecast of MLP models, trained on all Bunds and fused data	120
23	Next day return forecast results of MLP models trained on all Bunds	120
24	Different random initializations for next day reutrnr forecast of MLP models, trained on all Bunds	121
25	Next day return forecast result of MLP models trained on all Bunds and fused data	122
26	Different random initializations of next day return forecast MLP models trained on all Bunds and fused data	122
27	Next day price forecast results of MLP models trained on each Bund	123
28	Different random initializations of next day price forecast MLP models trained on each Bund	124
29	Next day price forecast results of MLP models trained on each Bund and fused data	125
30	Different random initializations of next day price forecast MLP models, trained on each Bund and fused data	125
31	Next day return forecast results of MLP models trained on each Bund	126
32	Different random initializations of best performing next day return forecast MLP models trained on each Bund	127
33	Next day return forecast results of MLP models trained on each Bund and fused data	127
34	Different random initialization of next day return forecast MLP model trained on each Bund and fused data	128
35	Next day price forecast results of LSTM models trained on all Bunds	129
36	Next day price forecast results of LSTM models trained on all Bunds and fused data	130
37	Next day return forecast results of LSTM models trained on all Bunds	131
38	Next day return forecast results of LSTM models trained on all Bunds and fused data	132
39	Next day price forecast results of LSTM models trained on each Bund	133
40	Next day price forecast results of LSTM models trained on each Bund and fused data	133
41	Next day return forecast results of LSTM models trained on each Bund	134
42	Next day return forecast results of LSTM models trained on each Bund and fused data	134

43	Next week price forecast results of less complex models	137
44	Next week return forecast results of less complex models	138
45	Next week price forecast results of arima models	139
46	Next week return forecast results of arima models	139
47	Next week price forecast results of MLP models trained on all Bunds	140
48	Different random initializations of next week price forecast MLP mod- els, trained on all Bunds	140
49	Next week price forecast results of MLP models trained on all Bunds and fused data	141
50	Different random initializations of best performing next week price forecast MLP models	141
51	Next week return forecast results of MLP models trained on all Bunds	142
52	Different random initializations of next week return forecast MLP models trained in all Bunds	142
53	Next week return forecast results of MLP models trained on all Bunds and fused data	143
54	Different random initializations of next week return forecast MLP models trained on all Bunds and fused data	143
55	Next week price forecast results of MLP models trained on each Bund	144
56	Different random initializations of best performing next week price forecast MLP model, trained on each Bund	145
57	Next week price forecast results of MLP models trained on each Bund and fused data	145
58	Different random initializations of best performing next week price forecast MLP model trained on each Bund and fused data	146
59	Next week return forecast results of MLP models trained on each Bund	146
60	Different random initializations of best performing next week return forecast MLP model trained on each Bund	147
61	Next week return forecast results of MLP models trained on each Bund and fused data	147
62	Different random initializations of best performing next week return forecast MLP model trained on each Bund and fused data	148
63	Next week price forecast results of LSTM models trained on all Bunds	149
64	Next week price forecast results of LSTM models trained in all Bunds and fused data	150
65	Next week return forecast results of LSTM models trained on all Bunds	151

66	Next week return forecast results of LSTM model trained on all Bunds and fused data	152
67	Next week price forecast results of LSTM models trained on each Bund	153
68	Next week price forecast results of LSTM models trained in each Bund and fused data	154
69	Next week return forecast results of LSTM models trained on each Bund	155
70	Next week return forecast results of LSTM model trained on each Bund and fused data	156

1 Introduction

1.1 Motivation

Artificial Intelligence and increasingly *Deep Learning* are by this time almost omnipresent in news, media, politics and many other parts of everyday life. Recent and successful use cases include speech recognition, image recognition, personal assistants, machine translation, chat bots, text mining and autonomous systems (e.g. self-driving cars) ([61],[45],[29],[62]). However, there is (among others) one characteristic, which is shared by the many projects with a large share of publicity: they are not concerned with the financial sector. Public interest was attracted by projects from automotive, from gaming and media, health-care, logistics, retail and certainly other sectors. Apart from robo advisory in wealth management, the financial sector seems to not play a major role in where the public suspects changes due to advancements in artificial intelligence, often connected to deep learning. One possible reason for this might be that successful techniques for image and speech recognition or autonomous systems do not seem directly applicable to the financial industry. Still, 31% of 345 surveyed German companies expect artificial intelligence to have a major effect on the financial sector, only being surpassed by logistics (43%), production (40%), trade (33%) and service (31%) [62]. On this account, this work targets one use case of the financial sector: It is concerned with the essential financial instrument of government bonds and its aim is to investigate how their development can be foreseen by modern methods of artificial intelligence.

The understanding of the motivation for this work in greater detail is further discussed in the remaining lines of this chapter. This explanation is two-fold. First, the characteristics of financial time series data will be addressed together with their applicability of deep learning methods. Second, the choice of the specific matter of bond price forecasting will be illuminated.

1.1.1 Applying Deep Learning Methods to Financial Time Series Data

The majority of successful applications of deep learning use data in form of images or word sequences. However, in the financial sector, data hardly comes in these formats. Most of financial data has a temporal component where the development of objects of interest over time is relevant [44]. However, the analysis of real world time series data is hard as it is often *noisy, complex and high dimensional* [72]. Additionally, ergodicity, stationarity and absence of autocorrelation pose problems to time series modelling in the financial sector which (in addition to other statistical

features) have not been overcome yet or as Cont [14] puts it:

Unfortunately, most currently existing models fail to reproduce all these statistical features at once, showing that they are indeed very constraining.

Traditional methods therefore generally fail to model time series with sufficient precision [44]. This is when machine learning comes into play. Neural networks and in particular deep networks can model arbitrary complexity [25], which is why they seem to suit complex time series modelling. Hsu et al. already demonstrated that the state of the art econometric methods for financial index forecasting are outperformed by methods of machine learning [30] to the surprise of economic theory. Research in this area is naturally not limited to the already mentioned and various attempts to improve stock price prediction and other financial data forecast can be found. This work investigates bond price development which is to the best of the author's knowledge a field of financial data science where the applications of machine learning are yet limited. However, also in this field, neural networks seem to deliver promising results [23]. In addition to these few and promising results of deep learning techniques applied in financial data science, there is another reason, which motivates the application to financial time series data. This is the recent success of Long Short Term Memory Recurrent Neural Networks (LSTMs) ([28],[61]). A sample LSTM cell is shown in figure 1. LSTMs are accountable for a variety of the already named

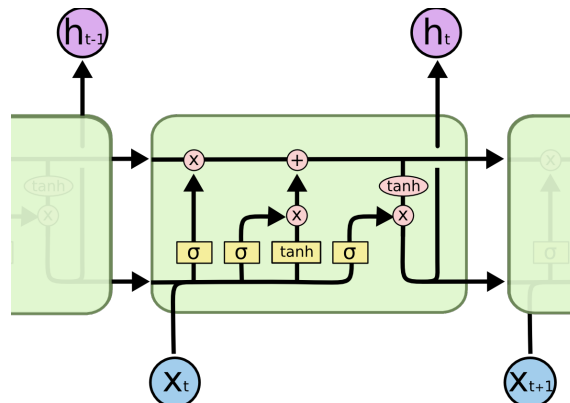


Figure 1: Sample LSTM cell, figure from Olah [52]

current advancements. They are successfully applied to machine translation ([71]), language modelling ([73]) and for example the generation of hand-writing ([26]). LSTMs success is justified by their ability to model sequences, which is key to solving the just mentioned problems. Furthermore, this ability to understand sequences and model temporal dependencies is essential to model financial time series data as well.

The combination of already successful application of neural networks to bond price forecasting without the use of for this purpose designed LSTMs, is encouraging for the idea of this work, the forecast of bond prices using LSTMs.

The motivation for bond prices as the domain for this investigation will be explained in more detail in the next paragraph.

1.1.2 Applying Deep Learning to Bond Price Forecasting

The motivation for investigating neural network based approaches and their general applicability to financial time series data has been addressed now. Next, government bond prices as an interesting field of applying neural network based methods to financial time series data will be discussed.

First, government bond data naturally has a relevant time component. Second, government bond prices share the characteristics mentioned earlier: they are complex, possibly noisy and high-dimensional. This might need additional explanation. Government bond development is certainly complex as economists have yet not unravelled all the factors and mechanism, which influence their pricing. This leads directly to the high-dimensionality of bond prices. The time series itself can be expressed by a function, mapping a point of time to a specific price which is not high-dimensional. However, trying to incorporate the vast variety of bond price influencing factors (which is probably necessary for satisfying modelling) quickly makes it a problem of high-dimensionality (2.4).

Third, government bonds play a major role in governmental finances (3), but seem underrepresented in related work regarding their importance for the international economy. The reason for this might be, that much of this work is conducted in secret due to the enormous financial interest in this area. Government bonds are the financial instrument responsible for most of German government debt (3) and a proxy for national economic well-being. Generally, their importance has increased for many countries, as government bonds played an important role to overcome the financial crisis of 2007 and 2008 ([35]). Bonds in US Dollars (USD), important as the currency of the world's largest economy and the standard unit of currency in international markets, account for around half of the USD denominated credit to non-banks (e.g. governments), outside the USA, (figure 2). Both, worldwide and in emerging markets (EMEs), the composition of credit is changing in favor of bonds. Understanding, being able to model and forecast the development of government bonds, would then have sustainable impact. Foreseeing the economic development of countries and acting ahead of potential financial or economical crises would just

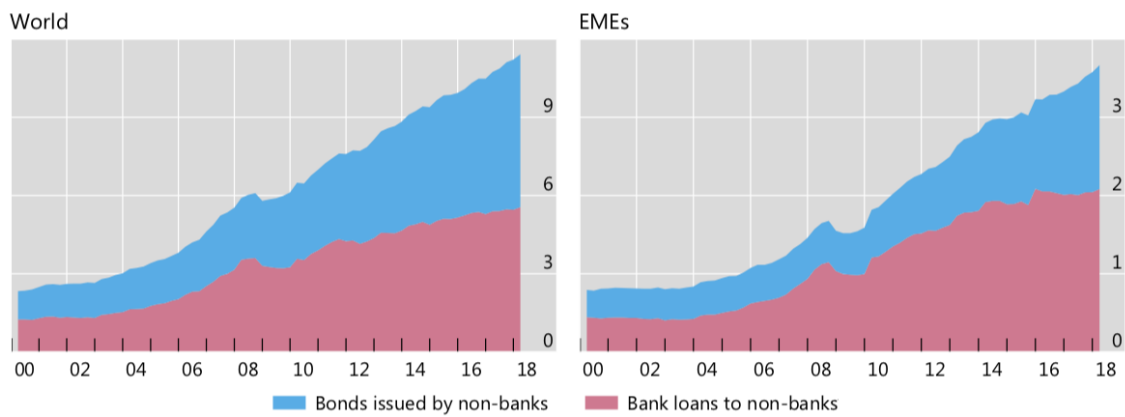


Figure 2: US-Dollar denominated credit risk to non-banks outside the United States according to the International Monetary Fund ([65]). X-axis depicting years from 2000 until 2018. Y-axis referring credit outstanding in trillion USD.

be one idea of enormous value to the public.

Fourth, due to their public nature, information on government bonds, their price development and their possibly influencing, external economic features, is often found to be publicly available in contrast to for example the related task of stock price forecasts where necessary information might not be available to the public.

Fifth, few existing work on government bond price forecast already indicates a superiority of neural network based approaches to other machine learning methods [23] as well as econometric models [44]. Additionally, the most successful government bond price forecasting models in related work, which are presented by Ganguli et al. [23], use neural network based models on the one hand. On the other hand, they do not report more complex networks (in terms of network depth or units per layer) and also do not utilize LSTMs, which are designed explicitly for this task. In addition, they omit any external data which might be highly relevant to the development of bond prices as economists claim ([27],[64],[48]) and focus their effort of model building solely on the technical data which comes with the government bond, e.g. coupon, maturity et cetera.

1.2 Thesis Objectives

In conclusion, the rapid advancement of Artificial Intelligence, the success of LSTMs for time series data, the tremendous importance of government bonds and the yet limited research in this combination, constitute substantiated motivation for this work. This thesis' overall objective is to investigate the feasibility of forecasting bond prices by deep learning methods. In order to achieve this overall objective,

several hypotheses have been formulated and will be evaluated in this work:

[Hypothesis 1]:

Government bond price development can be modelled by LSTMs on the basis of technical bond data and outperforms established computational models, including recent neural network based techniques.

[Hypothesis 2]:

Enriching the technical bond data by economic features improves the overall model performance of neural network based models in forecasting future bond development.

[Hypothesis 3]:

Neural network based models outperform other established methods for an increased forecasting horizon.

[Hypothesis 4]:

Neural networks for the task of return forecasting outperform neural networks aiming for price forecasting.

1.3 Thesis Outline

The investigation of the formulated objectives and hypotheses requires a variety of sub-tasks to be solved. This work is therefore structured in the following parts:

1. Following this general introduction an outline of the state of the art of related work is provided, from introductory words on deep learning ((2.1,2.2), over time series and financial data forecasting (2.3) until government bonds and their connection to economic development (2.4).
2. Afterwards, the data available to this work, is examined. First, the bond data (3) and later the economic data (4).
3. Subsequently, experiments are designed, based on related work and available data (5). The large amount of results will be summarized hereafter in the experimental results chapter (6). The full listing of results can be found in the appendix (10).
4. Finally, in chapter 7 the in this section formulated hypotheses are evaluated

based on the results of this work. The concluding evaluation results and potential future work is stated in Chapter 7 and 8.

2 Related Work

In this section, the related work to this thesis will be discussed. As this work investigates the application of sequence or time series modeling with deep learning in the domain of bond prices as a special type of financial data, the following will be structured into four subsections.

First, a general overview on deep learning will be provided answering questions about what deep learning actually refers to (at least in this work), its core ideas and a brief overview of what different problems with which deep learning methods have been solved in the past.

Second, from the broader introduction of deep learning the specific task of time series modeling and time series forecasting will be explained. Particularly, recurrent neural nets (RNN) ([25]) and long short term memory recurrent neural nets (LSTM) ([28][52]) as part of the deep learning umbrella term will be explained.

Third, since this is clearly not the first work, which applies machine learning techniques to financial data modeling, literature on this matter will be evaluated. This is also two-fold because (1) the different approaches to modeling this data are relevant for this work but also (2) the specific task. For this purpose, the promising approaches for modeling different types of financial will be discussed as well as the most successful ideas for the specific financial data type of bond prices.

Lastly, in addition to being the first to apply LSTMs to bond price forecasting, this is the first work which uses economic indicators in addition to the bond terms and past prices to forecast price development. Due to this novel approach it will be necessary to investigate which economic indicators were proven to influence the bond market. This will be conducted in the last subsection of the related work chapter.

2.1 Introduction to Deep Learning

The recent increase in usage of the term *deep learning* might lead to the believe that this is a very recent invention in the area of artificial intelligence research. However, deep learning is around since the 1940s ([25]) where researchers started to investigate how learning in biology works. Nowadays, the original motivation for this research is more and more obsolete as the differences between learning as humans do for example and deep learning are still large. These days the research in deep learning is mostly driven by prestigious successes in speech recognition, handwriting

recognition, text to speech synthesis, image classification, image segmentation and anomaly detection ([61],[63]). This recent development poses two questions: (1) Why it this success just recently discovered and (2) how do the techniques behind those results function? Both questions will be shortly discussed in this introductory section.

2.1.1 Recent Success of Deep Learning

The first question about the recent success of deep learning is answered for example by Andrew Ng, a professor at Stanford University and researcher at Baidu. During a presentation he gave at ExtractConf 2015 he argued that the availability of computational resources on the one hand and the huge amounts of data on the other hand is essential for the success lately ([51]) in contrast to the late 1980s and 1990s where the foundations of deep learning were already defined ([60],[28]), but the necessary resources and data were missing.

Today, the improvements on graphics processing units (GPUs) allow faster and more efficient training of larger (or deeper) neural networks as this process involves many simple matrix multiplications in the forward pass as well as in the backward pass through such a deep network (backward and forward pass will be discussed in the next paragraph). This task of matrix multiplication is highly parallelisable and the computational processing units (CPUs) although being faster than comparable GPUs do not allow for a high grade of parallelisation in contrast to GPUs.

Besides the hardware improvement for deep learning, there is the enormous increase in labelled data which is necessary for training deep networks. In the same talk ([51]), the value of deep learning is described as originating in the growing availability of large amount of supervised data as this is a task, at which deep learning models are especially good at.

2.1.2 Fundamentals of Deep Learning

Previously, an explanation of the recent success of deep learning has been provided as well as the information that this area of research is not as new as many people tend to think. What has not been discussed yet and is subject to this section are the basic concepts of deep learning. First, it will be reviewed how this is different from other supervised learning techniques. Then, the commonalities between the different deep learning approaches and their functionality will be discussed.

Classic Machine Learning vs. Representation Learning A major difference between deep learning and other supervised learning techniques is the fact that deep learning models learn the representation of the underlying problem by themselves. To illustrate this difference an example like handwritten digit classification might be helpful. Assuming the to be classified data is available as many, fixed grey-scale pictures. Then, in a classic machine learning approach the digital picture of the handwritten digit will be characterized by a variety of features. For example, the amount of greyness in a certain region or the overall amount of greyness in the picture to distinguish 1s from 8s as writing the digit 1 does not need as much "ink" as writing the digit 8 (at least in most cases). At first, this might seem to be a reasonable approach but as soon as the stroke width changes this could possibly lead to wrong classification. These types of hand designed features are obsolete in representation learning. Contrary to classic machine learning, the relevant features are learned by the technique itself. This difference is illustrated in figure 3, found in the Deep Learning Book by Ian Goodfellow et al. ([25]).

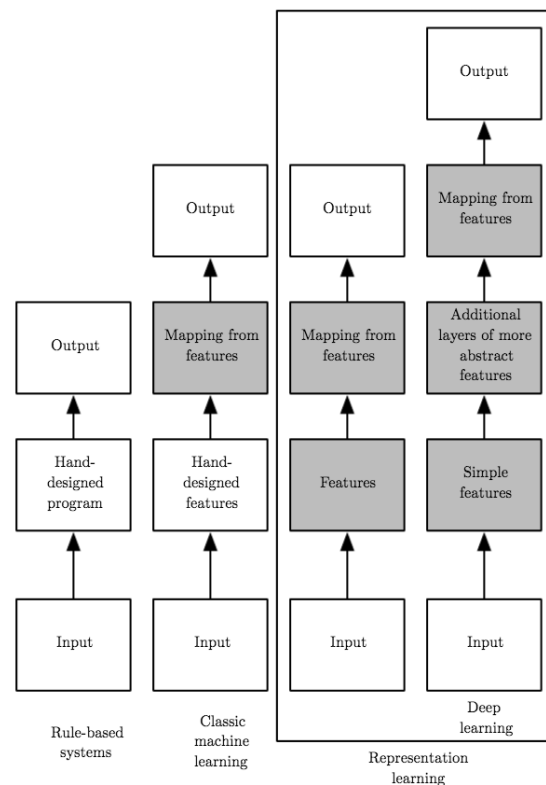


Figure 3: Rule based, classical machine learning and representation learning as found in Goodfellow et al. ([25]).

Deep learning then can be seen as a specific case of representation learning, in

which, from the features directly derived from the input, further more abstract features are obtained. This process of deriving more and more abstract features from the raw input features is the difference between deep learning and other representation learning approaches.

Deep Networks In the context of artificial neural networks, deep networks are artificial neural networks with multiple hidden layers. How many hidden layers are necessary for a network to qualify as "deep" is not defined and already two-hidden-layer-networks are referred to as deep from time to time ([25]) (more on this matter later in this section). Figure 4 illustrates a three-hidden-layer network (found in the Deep Learning book by Goodfellow).

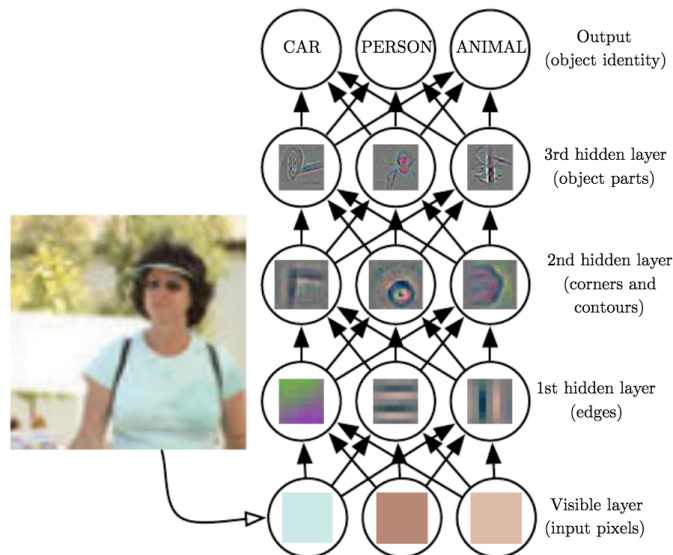


Figure 4: Illustration of a deep learning model as found in Goodfellow et al. ([25]).

2.1.3 Deep Feedforward Networks

Although this work will not use deep feedforward networks but a specific kind of RNNs, this paragraph will briefly introduce the basics of a deep feedforward network as they are widely used for many successful applications of deep learning methods and, more importantly, they share certain characteristics with RNNs, which are important to be understood for this work.

As in other machine learning disciplines, neural networks try to approximate a function $f^*(x) = y$, of which the exact f^* is unknown. Neural networks approximate f^* by a function $f(x; \theta) = y$ in a supervised manner ([25]). This approximate function $f(x)$ of the original, unknown function $f^*(x)$ is generally not a single func-

tion but on the contrary a composition of multiple functions. This is also how the term deep learning networks is different from other artificial neural networks: it relates to the number of functions that are composed in $f(x)$ ([25]). A simple feed-forward network (not deep though) of two functions could then look like the this: $f(x) = f^{(2)}(x)(f^{(1)}(x))$ where the result of $f^{(1)}(x)$ is used as input for $f^{(2)}$, which then calculates the output. This can also be used to illustrate another important term in ANNs, layers. Here, there are two layers. The input layer or the first layer, $f^{(1)}$, and the output layer or in this case the second layer, $f^{(2)}$. Any additional layer in between the two is then named hidden layer. The number of layers reflects the depth of the model ([25]). Although the depth of the model as a concept is rather clear (with some variations as well), there is no common definition of how many layers, equivalently which depth, is necessary to account for a deep network. So far, ANNs have been explained as a combination of functions. In this picture, it is obvious how an output for a given x as input is generated by applying the composed functions one by one to the input. Also, the to be defined parameters have been mentioned, which need to be adapted to approximate the unknown, to be estimated function as closely as possible. How can this be done?

Training is the term that is used in the context of machine learning to iteratively find the best parameters for the chosen architecture to approximate the original, unknown function. Training the chosen model is a process, which utilizes available observations of the input x and the desired output y and tries to find a general representation, which maps inputs x to an output y .

Training a machine learning model needs a defined optimization procedure, a cost function, and a model family. In addition to those, which are generally necessary for training a machine learning model, in deep learning, there are additional choices to be made, which are about the activation function and the architecture (How many layers? How many units? How are they connected?). Also, finding global minima of the cost function is not feasible due to the complexity of the composed function, which is why the efficient gradient based approach of back-propagation ([60]) is needed. This already answers some of the questions, which need to be decided to train a deep learning model. The optimization procedure is gradient based, most often with the back-propagation algorithm due to the chosen model family of neural networks. Before introducing the back-propagation algorithm later in this section, the other open questions are discussed, which are about the cost function, the architecture and the activation function in the hidden units.

The choice of a cost function is important for all kinds of machine learning

problems ([25]). In deep learning, the cross entropy is used for this purpose most often, defined as in the following ([25]):

$$J(\theta) = -\mathbb{E}_{\mathbf{x}, \mathbf{y} \sim \hat{p}_{data}} \log p_{model}(\mathbf{y}|\mathbf{x}) \quad (1)$$

Here, J dependent on θ (parameters of the approximation function) is the cost function. It is the cross entropy between training data and model prediction, which translates to the negative sum over all products between training data points and the logarithm of their corresponding model predictions. For discrete distributions, which is the case in this work as the investigated time series consist of a finite number of data points, the cross entropy can be represented with this formula, which might be easier to read ([77]):

$$J(\theta) = -\sum_x p(x) \log q(x) \quad (2)$$

Now as the cost function is defined which is not exclusively used in neural networks, next, output units are addressed. There are many different output units available and they all have their upsides and shortfalls, of which the most common output units are linear, sigmoid and softmax output units. As the output unit is part of the last layer, the output layer, the calculation of the output unit is expected to fit the problem, which is modelled. Then, there are natural choices for an output unit depending on a problem and its assumed underlying distribution. Firstly, the linear output layer consisting of possibly multiple output units, defined in the following equation, approximates the target y by a multiplication of the features h with the weights W plus a bias term b ([25]):

$$\hat{\mathbf{y}} = \mathbf{W}^T \mathbf{h} + \mathbf{b} \quad (3)$$

This can be used for the forecast of a numerical value, for example, and might be less suitable for a classification problem. Next, there is the sigmoid output unit. A sigmoid output unit is defined as the following ([25]):

$$\hat{\mathbf{y}} = \sigma(\mathbf{w}^T \mathbf{h} + b) \quad (4)$$

A sigmoid output unit consists of a linear part, exactly the same as the linear output

unit, but then applies a σ function defined like the following to the result ([55]):

$$\sigma = \frac{1}{1 + e^{-x}} \quad (5)$$

The common use case for a sigmoid output would be a binary classification. If a classification over more than two classes is desired, the softmax output unit is the one to choose generally. Here, the standard linear output (here denoted with z is calculated and then used as input to the softmax function, which is calculated for all i where i are the positions in the vector z .

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \quad (6)$$

Below, a comparison between the three common output units can be found for an exemplary input range (h) from -5 to 5 , $b = 0$ and $W^T = 1$.

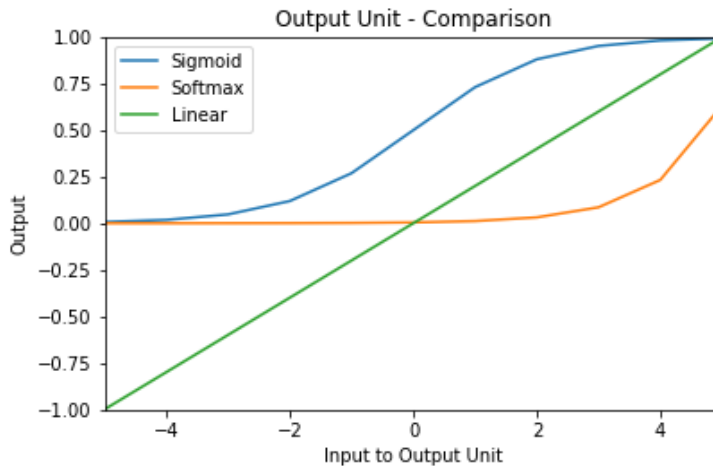


Figure 5: Comparison of common output units.

Next, hidden units and activation functions in neural network design are briefly introduced. As the design choices for hidden units and activation functions are still part of the active research area of deep learning, again, there is no globally superior choice for one or the other hidden unit or activation function. Generally, this is identified in the process of developing a deep learning model. However, rectified linear units (ReLUs) are a reasonable choice to start with as justified by Goodfellow et al. ([25]) due to their similarity to linear units, their consistent gradients and interpretable gradient direction.

Rectified linear units work similarly to linear output units. They also receive an input x and calculate a result in the same way. The difference then comes with

the activation function, which is used on top and applied for every element in the resulting vector z ([25]):

$$\mathbf{z} = \mathbf{W}^T \mathbf{x} + \mathbf{b} \quad (7)$$

These activation functions are non-linear and allow neural networks to fit non-linear data at the end ([25]). The activation function, here denoted with $g(z)$, for a rectified linear unit is defined as $g(z) = \max(0, z)$, meaning that any input z will be propagated unless it is smaller than 0. Then 0 is propagated through the unit into the next layer of the network. One rather obvious drawback of ReLUs is the fact that they cannot differentiate between negative inputs, which makes learning from negative input impossible for units using ReLUs impossible ([25]). To overcome this disadvantage, adaptations of ReLUs may be utilized, for example a leaky ReLU, which then does not set negative input to 0, but rather multiplies it by a small, positive value, or an Exponential Linear Unit (ELU), which calculates $\alpha(e^z - 1)$ for negative inputs. An overview of the different types of linear units, sigmoid and hyperbolic tangent, which are also used as activation functions is depicted below:

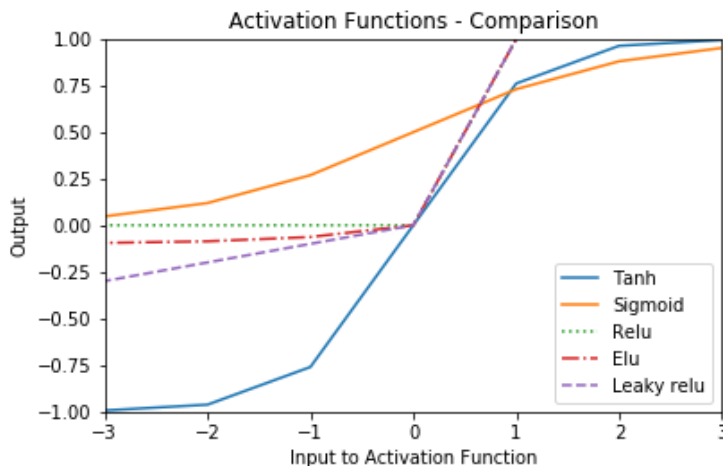


Figure 6: Comparison of activation functions (input $z \in [-3, 3]$).

The sigmoid and hyperbolic tangent (tanh) are widely used as activation functions. However, both share the same disbenefit: They both only have a high absolute slope in regions around 0. For smaller and larger values the slope gets close to 0, which impedes any gradient-based learning ([25]). This effect of the sigmoid function and the hyperbolic tangent can be observed in the figure above (blue and orange lines). This is one of the reasons why linear units have gained more attention recently in the domain of feedforward networks ([25]). Another one is the fact that gradient-based learning for linear units is easy. This is because the gradient is large

and the second derivative is 0 as long as the input is larger than 0, which means that the iterative gradient-based optimization approach in theory is very well applicable ([25]). One of the few downside of linear units is the problem of a non-existing derivation at input $x = 0$. This, however, is practically solved by using left or right derivatives in the established deep learning frameworks like TensorFlow ([49]).

Cost function, hidden and output units are now briefly introduced. Apart from the gradient-based learning algorithm, which will close this introduction to deep feedforward networks, the question of how to design the structure of a neural network remains to be discussed. The abstract structure of a feedforward network, however, is given. A group of hidden units is organized in a layer and attached to possibly more layers before the output layer containing the output units marks the final layer. Mathematically, the first can then be described as:

$$\mathbf{h}^{(1)} = g^{(1)} (\mathbf{W}^{(1)T} \mathbf{x} + \mathbf{b}^{(1)}) \quad (8)$$

Continuing with the second layer like this:

$$\mathbf{h}^{(2)} = g^{(2)} (\mathbf{W}^{(2)T} \mathbf{h}^{(1)} + \mathbf{b}^{(2)}) \quad (9)$$

This composition of layers continues until the output layer and yields this a chain-like structure where each element of the chain is a layer consisting of possibly multiple units. When defining the structure of the network questions regarding the number of layers, the number of units in each layer, the order of the layers and the connection between layers and units need to be addressed. Unfortunately, there is no universal guideline for choosing the most suitable architecture for a given task. The ability of neural networks to learn non-linearity with only one hidden layer ([25]) does not ease structural decisions. In general, a given task may be approximated with a single layer, which contains more units in the same way a deeper, multi-layer structure with far less units per layer might provide results of the same quality. However, there is a risk of non-deep networks to not generalize as well as deeper networks while the deeper networks are considered to be harder to train ([25]). At the end, the choice for a structure is dependent on the task to be tackled.

The missing part to this introductory section on deep feedforward nets is the training algorithm, which is essential for deep learning with the help of back-propagation and gradient based learning. By now, what has been introduced is the forward path through the network, meaning that an input x flows through the network and after passing through the output layer, an output is provided. What is

missing is the process of adapting the weights in the now decided network architecture to reduce the training error. Theoretically, this could be achieved by computing precise minima or maxima since the network can be seen as a function. However, these methods provided by analytical calculus are computationally costly and not feasible in most realistic scenarios ([25]). The back-propagation algorithm defined by Rumelhart et al. ([60]) together with any gradient based learning algorithm provides an inexpensive solution for computing the gradient of the cost function in a neural network, denoted as $\nabla_{\theta} J(\theta)$ ([25]). Gradient based learning techniques utilize the calculated gradient (the vector of partial derivatives, which can be understood as the descent in the directions of the dimensions) to find (most often) local minima to the cost function $J(\theta)$. The idea behind gradient based learning is to use locally available information to iteratively find a (local) minimum of the cost function. This is achieved by adapting the weights in the opposite of the gradient direction as this reflects the *steepest descent*. The only parameter, which needs to be adapted in this scenario, is the *learning rate*, which equals the step size in the direction of the steepest descent ([46]). The choice of this parameter can be critical as too large learning rates might overstep the sought minimum and too small learning rates might decelerate the approach. This idea of gradient descent as one gradient learning algorithm can be expanded and other faster gradient learning algorithm, often referred to as *optimizers* are available, e.g. stochastic gradient descent, gradient descent with momentum, Adagrad et cetera ([59]).

Although the main idea of gradient based learning is now introduced, the calculation of the gradient has not been tackled yet. The calculation of the gradient is only of interest because in deep learning the analytical tools, which are available to compute derivations are not easily applicable due to the size of the networks. Also, a numeric computation is not feasible due to its computational cost. These two reasons make the usage of the back-propagation algorithm essential for neural networks. To explain the concept of back-propagation it is helpful to introduce the concept of the computational graph as a representation of a neural network. In this graph, the vertices correspond to operations, directed edges are drawn from a vertex A to B if and only if A produces an output used as an input for B. The first nodes in this graph receive additionally variables as input and the last nodes in this directed graph output the result of the computation. Generally, many functions can be computed by these computational graphs (details about limitations can be found in Goodfellow et al. ([25])). An example of such a graph, representing the function $f(x, y, z) = (x + y) * z$ can be found below:

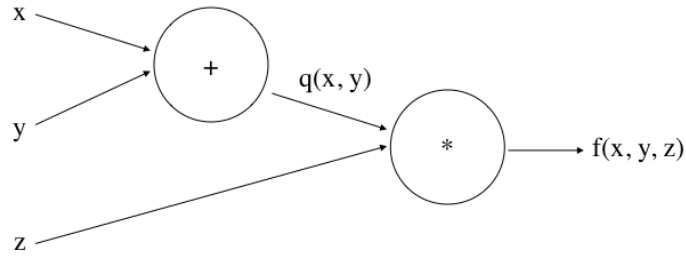


Figure 7: Example: computational graph ([46]).

Actually computing the gradient of the function f for an input tuple (x, y, z) then means to compute the partial derivations ([46]):

$$\frac{\delta f}{\delta x}, \frac{\delta f}{\delta y}, \frac{\delta f}{\delta z}. \quad (10)$$

The calculation of those is not directly clear, also when the function is translated in the computational graph. What can be derived from the graph already is the local information for each node, meaning the partial derivations at the nodes. For this example, these depict themselves as below ([46]):

$$q = x + y, \frac{\delta q}{\delta x} = 1, \frac{\delta q}{\delta y} = 1$$

$$f = qz, \frac{\delta f}{\delta q} = z, \frac{\delta f}{\delta z} = q \quad (11)$$

This information already provides the solution for the partial derivation of f in direction z . Still missing are the directions x and y . Those are calculated with the application of the chain rule (of calculus), which is applied below to calculate the direction of x ([46]):

$$\frac{\delta f}{\delta x} = \frac{\delta f}{\delta q} \frac{\delta q}{\delta x} \quad (12)$$

This iterative application of the chain rule is the main idea behind back-propagation and essential to a wide array of practical application of neural networks. Similarly to the provided, simple example, this approach can be used for other functions and other networks and for multi-dimensional input to model more complex problems.

As the relevant fundamental concepts of deep learning for this work are now introduced, in the next step, the possibilities of deep learning for the general problem of time series modeling will be discussed.

2.2 Deep Learning for Time Series Analysis

2.2.1 Time Series Analysis

Before investigating the use of neural networks and deep learning in time series modeling, it is certainly beneficial to first define what a time series and what the challenge of modeling a time series actually is. Naturally, time series occur in many application areas, of which economics and finance are two prominent ones ([8]). Time series in finance is usually described as in the quote below:

A time series is a sequence of numerical data points in successive order. In investing, a time series tracks the movement of the chosen data points, such as a security's price, over a specified period of time with data points recorded at regular intervals. ([38])

In this work, the chosen data points are Bund prices and also returns as described later in this work. Also, period of time and the regular interval selection is discussed in the same later chapter 3. Next, as the nature of a time series is introduced, the question remains how to actually analyze this data and more importantly how to make forecasts of the development of the time series as this is essential to this work. Generally, there are plenty of methods in use for this purpose. Providing detailed information on all the relevant methods for time series forecasting would certainly exceed the scope of this work. However, the idea of classical decomposition, stationary, autoregressive and moving average processes are beneficial for an understanding of the later tested ARIMA model and are therefore defined in the next few paragraphs.

Trend, Seasonality, Cycles and Residuals One of the well established methods to describe a time series is the method of classical decomposition ([8]). Following this approach, a time series ([8]) (denoted as $X_{t-n}, X_{t-n+1}, \dots, X_t = X$ for a fixed length n) into:

- Trend: long term differences in mean ([8])
- Seasonality: cyclical changes due to the calendar ([8])
- Cycles: other cyclical changes ([8])
- Residuals: random (or other) fluctuation ([8])

A time series, X , then equals a conjunction of trend, seasonality, cycles, and residuals.

(Weakly) Stationary Processes (Weakly) Stationary Processes are another description of time series. To satisfy the requirements of a stationary process, a time series must fulfill the following:

1. The time series must have a constant expected value: $\mathbb{E}(X_t) = \mu$, where μ is constant ([8])
2. The autocovariance function must be independent of γ_k : $cov(X_t, X_{t+k})$. This means that the function must not depend on t or k , but depend only on the difference $t - k$ ([8])

Autoregressive Processes The autoregressive process of order p is referred to by $AR(p)$, and can be defined as follows ([8]):

$$X_t = \sum_{r=1}^p \phi_r X_{t-r} + \epsilon_t$$

where ϕ is a sequence of fixed constants and ϵ is a sequence of independent random variables with mean 0.

Moving Average Processes The moving average process of order q is denoted by $MA(q)$ and defined as follows ([8]):

$$X_t = \sum_{s=0}^q \theta_s + \epsilon_{t-s}$$

where θ is a sequence of fixed constants and ϵ is a sequence of independent random variables with mean 0.

ARMA Processes The autoregressive moving average process, $ARMA(p, q)$ then is defined as follows ([8]), with ϕ , θ and ϵ as introduced:

$$X_t = \sum_{r=1}^p \phi_r X_{t-r} + \epsilon_t + \sum_{s=0}^q \theta_s + \epsilon_{t-s}$$

ARIMA Processes ARIMA processes are an extension to ARMA processes. They prove useful if the underlying data is not stationary, which is a requirement for a successful time series analysis as an ARMA process. The addition of ARIMA to ARMA comes with the idea of the first order difference. So, instead of analyzing the

original time series, the first order differences are then modeled denoted as follows ([8]):

$$X_t = \Delta Y_t = Y_t - Y_{t-1}$$

This slightly different time series might then be suitable to be analyzed as an ARMA process. If not, second and higher order differences can be tested ([8]).

2.2.2 Recurrent Neural Networks

Recurrent neural nets (RNNs) (first described by Rumelhart et al. ([60])) are a family of neural networks for the processing of sequential data. A popular view is that convolutional neural networks (CNNs) are specialized for processing grids of values (e.g. images) whereas RNNs are specialized for processing sequences of values ([25]). Although this view seems to change these days and there is work using CNNs for sequence modelling as well ([40]), the general idea behind the invention of RNNs was sequence modelling and it is still widely used for this purpose, which is why a brief introduction into RNNs is provided in this paragraph.

One of the key ideas behind RNNs is the idea of parameter sharing over different parts of the model. This means that, in contrast to feedforward networks as described earlier (2.1.3), not every input feature has its own parameters configured, but that they are the same over several input features in RNNs. This is referred to as sharing. Sharing can be achieved by including cycles in the computational graph (2.1.3,[25]). Cycles then define the influence of a value on its own successors in the sequence. These cyclic computational graphs then can be utilized to model dynamic systems, where the result of an equation is dependent on the result of that same equation at an earlier point of time. Formulated as a cyclic computational graph, this equation can be depicted as in the following ([25]):

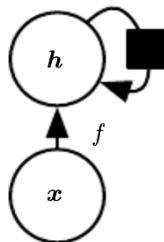


Figure 8: Example: cyclic computational graph, black box indicates one step in the sequence ([25]).

To avoid a recurrent formulation, the same equation can be applied a finite number of times, which yields an expression of the same equation without recurrence.

The advantage of avoiding the recurrent formulation then, is that this expression can then be depicted as a directed acyclic graph, as illustrated in the following ([25]):

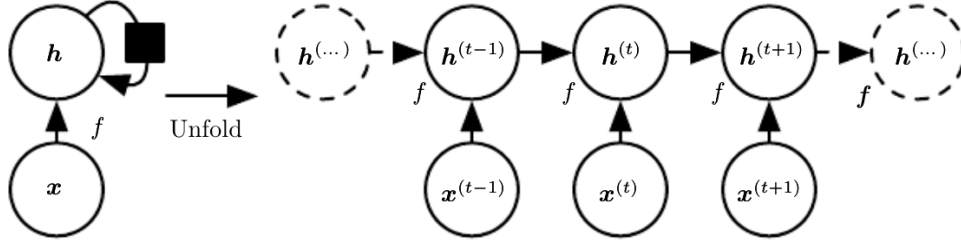


Figure 9: Example: unfolding of a cyclic computational graph ([25])

This figure corresponds to the following recurrent equation, where \mathbf{h} indicates that the state is hidden within the network, \mathbf{x} represents the feature input vector, t the element in the sequence and θ the configuration of the network ([25]):

$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \theta) \quad (13)$$

With the unfolded representation of recurrent networks, they can be treated as feedforward networks. Of particular interest then, is the fact, that backpropagation as introduced for feedforward networks, can then be used to model sequences in recurrent networks as well. This insight then results in a variety of possible recurrent network architectures, which are not specified in this work. More information on this matter can be found for example in Goodfellow et al. ([25]).

So, if sequence modelling by RNNs works, this should be used for the purpose of this work as well. However, there is one important drawback to RNNs, the challenge of long-term dependencies ([25]). The precise mathematical problem will not be discussed here, more information on that can be found in Goodfellow et al. in chapter 8.2.5 ([25]). There, the problem is summarized as:

The basic problem is that gradients propagated over many stages tend to either vanish (most of the time) or explode (rarely, but with much damage to the optimization).

This problem occurs in recurrent networks but not in other, non-recurrent networks. This can be easily illustrated by a multiplication of a weight w in the network by itself many times. The product w^t will either converge to almost 0 or to very large numbers depending on $w < 1$ or $w > 1$ ([25]). In case of a non-recurrent network with different weights $w(t)$ at each time step, it is not guaranteed that the problem does not occur but is certainly more likely that it does not.

2.2.3 Long Short Term Memory

One possible solution to prevent exploding or vanishing gradients, but still maintain the structural advantages of RNNs, are long short term memory recurrent neural nets (in short LSTMs, ([28])). LSTMs have the same chain-like structure as RNNs as earlier introduced. The difference lies in each unit of the chain. While RNN units mostly consist only of a simple input, activation function and output combination (figure 10), LSTM units, also denoted as LSTM cells, have a more complicated structure within their units (figure 11).

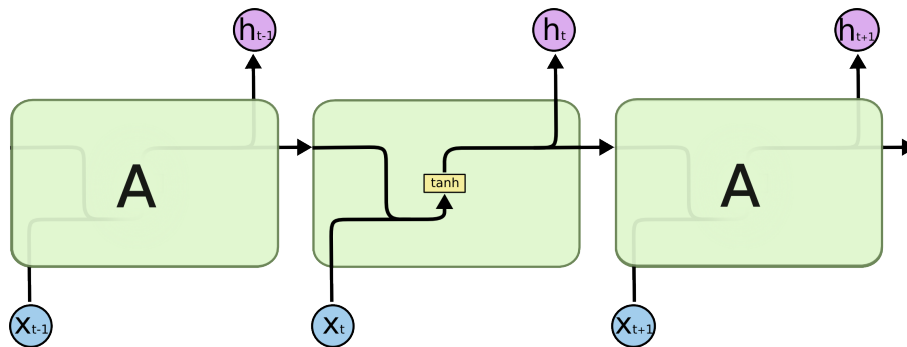


Figure 10: Simple RNN, figure from Olah ([52]).

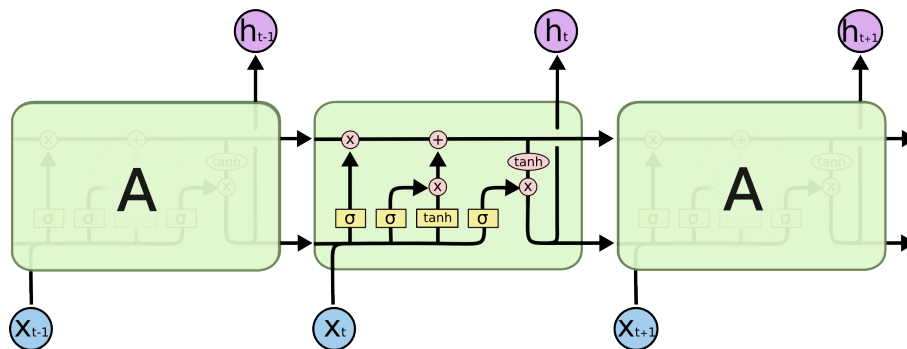


Figure 11: LSTM cell, figure from Olah ([52]).

This structure consists of the following components, which will be briefly addressed next:

- Cell state (12)
- Forget gate layer (13)
- Input gate layer (14)
- Update layer (15)
- Output layer (16)

Cell state The cell state, highlighted in the figure below (12), is the structure connecting the different LSTM cells and allowing information to easily flow from one cell to the next (and back when updating). The information contained in the cell state can be changed by other mechanism within the LSTM cells. These changes are conducted by so-called *gates*.

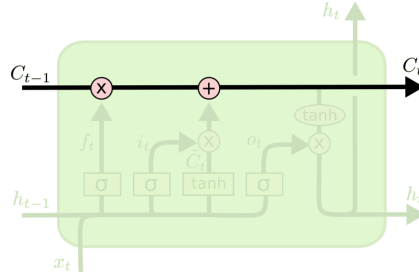
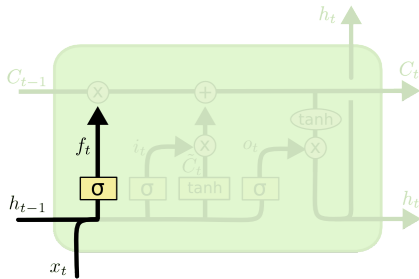


Figure 12: Cell state, figure from Olah ([52]).

Forget gate layer The forget gate (13), as one might guess, can be used to let or not let information through and incorporate this information in the cell state from LSTM cells located prior to the current cell in the chain. This is realized by a sigmoid function, which returns numbers between 0 and 1 as seen earlier. Dependent on this output, earlier information will be used ranging from not used at all (if sigmoid returned only 0) or used completely (if sigmoid returned 1). The amount of earlier information let through and the current cell state will be then point-wise multiplied to obtain the new cell state.



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Figure 13: Forget layer, figure from Olah ([52]).

Input gate layer This alteration of the cell state however only has effects on the question of how much of past information should be used. Current information available due to the new inputs and how this can be added to the cell state is next. This is achieved in the input gate layer (14), consisting of the actual input gate, which decided what needs to be updated, and a hyperbolic tangent layer, which decides by how much the update will effect the cell state.

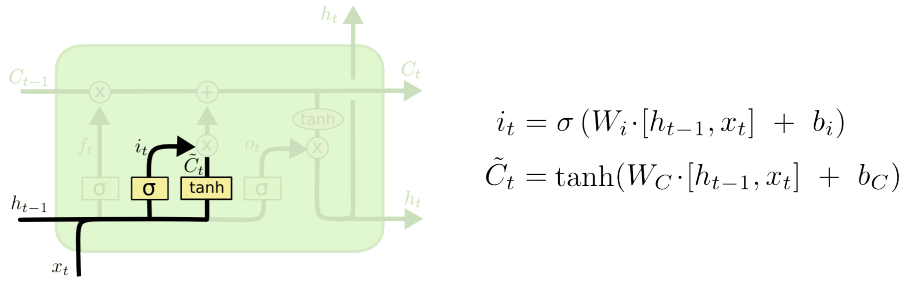


Figure 14: Input gate layer, figure from Olah ([52]).

Update layer Now, as it is calculated how much information from the past will be kept and how new information is added, this is actually conducted and the new cell state will be computed by multiplying the old state by f_t , resulting in the desired forget behavior, and adding the product of i_t and C_t . This results in a new cell state.

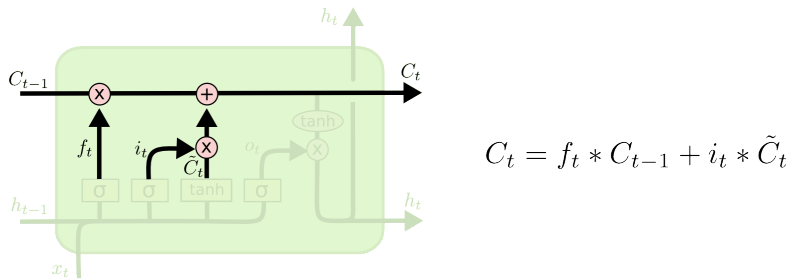


Figure 15: Update layer, figure from Olah ([52])

Output gate layer Before outputting this cell state then, there is a filter to adapt this state consisting of a sigmoid, a hyperbolic tangent layer and a point-wise multiplication (16). This filter can be useful if one does not actually want to output the cell state, but maintain implications, which this cell state might have for future cells.

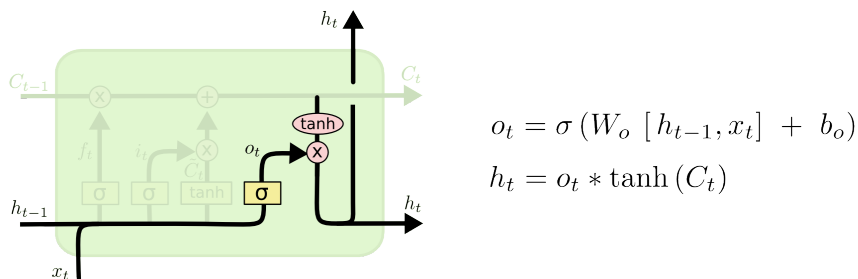


Figure 16: Output layer, figure from Olah ([52]).

LSTM networks have been shown to be better suited for learning long-term

dependencies than RNN networks ([25]), which is why they are certainly to be considered in this work.

2.3 Financial Data Forecasting

Until now, the focus of this chapter has been set solely on deep learning and its use for time series modeling. Next and independently from deep learning, financial data forecasting, in particular stock and bond prices, will be illuminated. Of special interest in this field are naturally results, which can be used as a baseline for the later proposed approach for bond price forecasting. However, also the work in the closely related field of stock prices will be investigated as ideas in this areas might be applicable to the task of bond price forecasting. First, successful results for stock price forecasting and their approaches will be discussed as the number of publications in this area is rather high. This comparison will allow for a first idea of which methods might be fruitful for the task of bond price forecasts. Then, ideas for bond price forecasting will be addressed.

2.3.1 Stock Price Forecasting

The amount of work, which is concerned about understanding stock prices or forecasting them is rather high. The efforts, which are put in this area by banks and other players in the financial sectors are most probably even higher than in public research as price predictions with high accuracy guarantee profits.

Generally, efforts for price forecasting in the stock market can be distinguished into two categories, the fundamental and the technical analysis ([10]). The difference can be briefly explained as technical analysts believe that all the necessary information for price forecasting can be found in the price development itself whereas fundamental analysts incorporate other information, for example information about the company in their analysis and their estimation of the price ([10]). The approach presented in this work belongs rather to the fundamental analysis although the training of the model later might yield the result that the price development is the major feature for price forecasting and then identify patterns in past prices to forecast future values. In order to obtain an overview, which methods are used in fundamental analysis lately and how successful they are, related work in this area will be introduced in three categories: traditional statistical learning, machine learning and deep learning approaches.

Traditional statistical linear learning approaches, which are widely used (according to Cavalcante et al. ([10])) are time series regression, exponential smoothing, autoregressive integrated moving average (ARIMA) and its variations, generalized autoregressive conditional heteroskedasticity (GARCH). An example of a pure ARIMA approach can be found in the work by Adebisi et al. ([2]).

Before diving deeper into the results of Adebisi et al., the general idea behind ARIMA models will be recapitulated as already discussed earlier (2.2.1). As the name already implies, ARIMA models combine three ideas in one modeling approach. ARIMA models are defined by the parameter tuple (p, d, q) . The letter p stands for the autoregressive part, the letter d for the integrated part and the letter q for the moving average part of the ARIMA acronym. Next, the meaning of these parameters will be elucidated.

First, there is the *autoregressive* or *AR* component defined by the parameter p . This autoregressive component is the weighted sum over a constant number of past data points of the time series (2.2.1). The choice of weights is chosen by training on the past. How many past values are chosen, which is at the end referred to by p , has to be defined differently and this will be discussed later in this paragraph.

Second, there is the term *integrated*, which is called the differencing term and denoted by I in the ARIMA acronym and by the parameter d in the parameter tuple (2.2.1). It is used to make the series stationary which has several reasons. One is that for example correlation and other statistical measures are only useful in this context. Also, non-stationary time series will always be over- or underestimated by an ARIMA model. Making a time series stationary by differencing is the idea to not build the model directly on the time series y_t itself but rather on the d^{th} difference, $\Delta^d y_t$. The first difference $\Delta^1 y_t$ is then defined by $\Delta^1 y_t = y_t - y_{t-1}$. Important to note here is that the second difference is not defined by $\Delta^2 y_t = y_t - y_{t-2}$ but rather by $\Delta^2 y_t = \Delta^1 y_t - \Delta^1 y_{t-1}$. This can be expanded then for any integer number d , which is then reported in the parameter tuple (p, d, q) . The idea of differencing however cannot guarantee to yield a stationary series. This has to be identified in testing different parameters and observing mean and variance over the series for the obtained results.

Third and last, there is the idea of a *moving average* or *MA* in ARIMA, defined by the parameter q (2.2.1). The parameter q refers to the number of data points, of which the weighted errors are considered for calculating the current data point in time, y_t . Here again, the best weights for the given problem are defined by training whereas the number of points is chosen with a different tool, which is defined next.

The identification of suitable parameters p, d, q for an ARIMA model can be

conducted in different ways. One rather straightforward but also slightly arbitrary method is the utilization of autocorrelation for this purpose. Autocorrelation means the correlation between a time series and its past values. Depending on how strongly correlated a time series with how many past values is Tools to identify the three parameters: Autocorrelation to identify parameters p, d, q . Autocorrelation is the correlation between a time series and its past values.

Together, these parameters then define a specific ARIMA model ([82]) of the form:

$$y_t = c + \epsilon_t + \sum_{i=1}^p a_i y_{t-i} + \sum_{j=1}^q b_j \epsilon_{t-j} \quad (14)$$

In their work, Adebiyi et al., develop two ARIMA models, one, which forecasts the development of the Nokia stock price, and another one forecasting the Zenith Bank stock. They claim that both of their models prove that ARIMA models can achieve satisfying results in stock price prediction. However, the applicability of their results to other stock prices and their development might be less successful as the Nokia and Zenith bank rates seem to have developed rather linearly. Non-linear, complex price developments over time might be not sufficiently modelled with the chosen ARIMA approach. This in general is a problem of the so-called traditional statistical learning approaches. They all share the property that they cannot facilitate highly non-linear modeling ([68],[31]). However, as ARIMA (and other) approaches show, for certain stocks and their price development modeling linearity seems to be sufficient. What is necessary at the end however, is a model with the ability to model both, linear and non-linear behaviour on the markets ([79]).

Non-Linear Machine Learning This paragraph provides a short overview of the approaches on stock price forecasting using what is here called machine learning techniques. In contrast to traditional statistical learning approaches, machine learning techniques can capture non linearity without assumptions about data ([10]), which is why in literature there are currently favored over the provided example of ARIMA models. Frequently used methods in machine learning for stock price forecasting are support vector machines adapted for regression (SVR) and artificial neural networks (ANN). In the review paper by Cavalcante et al. ([10]) it is stated that previous work using ANN was mostly focusing on feedforward ANNs with a limited depth (a number of layers of around 3 is observed frequently). However, this seems to be sufficient to show the listed advantages of being able to cope with non-linearity firstly and secondly being able to handle data with unknown underlying properties ([47]). One rather recent work by Zhong et al. uses ANNs: Stock market return

([85])

Despite all the advantages of ANNs in forecasting financial time series, Cavalcante states that ANNs are highly sensitive to various parameters, such as input data and target variables, the type of the neural network, the number of layers, learning rate and optimization algorithm and are therefore complex to train correctly ([10]). This is one of the reasons why much literature using SVR can be found ([31],[9],[32],[17]).

Hybrid Models have gained momentum due to the fact that researchers wanted to combine advantages of multiple forecasting techniques and eliminate disadvantages of for example finding local vs. global optima (it is claimed that ANNs tend to find local optima ([47])), the difficulty of hyper-parameter optimization and the avoidance of overfitting ([10]) in addition to the earlier mentioned disadvantages of SVMs and ANNs. Combinations include ARIMA and ANNs by Zhang et al. ([84]), ARIMA and SVR by Wang et al. ([79]), the combination of decision trees, ANNs and SVMs in an expert system, which includes price forecasting by Weng et al. ([81]) and a combination of SVM and KNN for stock market forecasting by Chen et al. ([11]).

Deep Learning approaches, lastly, have especially recently drawn more attention although the field of deep learning in finance is still rather unexplored ([10]). The recent rise in attention might be caused by the the general recent success of deep learning as described earlier. However, there is literature, which suggests that deep learning can be a viable approach for stock price forecasting. Successful work can be for example found in Chong et al. ([13]), Jiang et al. ([40]), Ding et al. ([15]) and more in the review paper by Langkvist ([44]). Interestingly, none of these methods use the for the exact purpose of time series modeling invented LSTMs. This might be due to the assumption that prices do not depend heavily on past prices and can be expressed by other factors (fundamental analysis). This seems to conflict with the big advantage of neural networks that they can be both, a fundamental and a technical analysis ([43]). Technical analyses by definition works with past stock data. Not using past data at all then might not make good use of the major advantage of neural nets as conducting possibly both, a technical and a fundamental analysis at the same time. This also causes the thought that a LSTM based approach for financial data forecasting is worth investigating.

This as the first reason for investigating LSTMs in a financial data forecasting context is supported by a second reason, which are the results that have been observed

in the referenced related work. Chong et al. ([13]) seem to outperform ARIMA models just slightly with the presented hybrid DNN approach. Others (e.g. Jiang et al. ([40])) show promising results for DNNs but with a slightly different task again (portfolio management). So, results for different tasks in the domain of stock markets mostly with DNNs are mixed. It is certainly not decided yet, which approach works best for the different tasks discussed in this area of research.

2.3.2 Bond Price Forecasting

The specific task of bond price forecasting is less well studied than the different facets of stock price forecasting, be it trend or concrete price forecasting. Most the work on this matter has a financial or economic background and does not investigate specific algorithms for forecasting the bond price or a trend for the bond price development. Two sources from the area of computer science, which will be used for creating a baseline for the later introduced work here are the Kaggle challenge on bond price forecasting ([69]), which provides a first idea of the necessary quality for satisfactory results. In addition, there is the work by Ganguli et al. ([23]), which compares different approaches for bond price forecasting including autoregressive-moving average models (ARMA), similar to the earlier mentioned ARIMA model, regression trees, random forests as well as ANNs. Ganguli et al. ([23]) conclude that neural networks perform best for this task. They use neural networks with just two layers for this task using Levenberg-Marquardt optimization. They also report that more complex networks (meaning a higher number of neurons) yield a better performance on the test set while compared to two-layer networks with fewer neurons. The fact that increasing complexity improves performance is another hint why the implementation and evaluation of the more complex LSTM-based approach might prove successful.

Contrary to the here proposed solution however, is the fact that the information, which they use is limited to the information directly associated with the bond. This involves data that is slightly more than in a pure technical analysis but still not a fundamental analysis in the sense that it uses all the relevant data associated with the bond pricing. Ganguli et al. ([23]) use the same data which is used in the Kaggle competition. Columns of this data set are labelled as the following as can be found on Kaggle ([69]):

- `id`: id of the row
- `bond_id`: id of the bond (only in training data)

- `trade_price`: the price of the corresponding trade
- `weight`: weights the row for the evaluation process in the challenge. Long periods of no trading result in a higher weight for the row (assumption that price is correct if longer constant)
- `current_coupon`: coupon at the time of the trade (definition of coupon later in section about government bonds)
- `time_to_maturity`: time until the bond has to be paid back (more details also later)
- `is_callable`: true/false. Right of the issuer to buy back the bond before maturity date is reached
- `reporting_delay`: time difference between trade and report of the trade
- `trade_size`: amount of the trade
- `trade_type`: between customers or dealers
- `curve_based_price`: A fair price estimate based on implied hazard and funding curves of the issuer of the bond
- `received_time_last_diff{1-10}`: The time difference between the trade and that of the previous {1-10}
- `trade_price_last{1-10}`: trade price of the last {1-10} trades
- `trade_size_last{1-10}`: trade size of the last {1-10} trades
- `trade_type_last{1-10}`: trade types of the last {1-10} trades
- `curve_based_price_last{1-10}`: curve based price of the last {1-10} trades

In total, the data set contains around 750000 rows, which equals 750000 bond trades. It is important to notice that these are not 750000 different bonds. Many of the rows (trades) in the data set refer to the same bond identified by the `bond_id`. Overall, there are about 3500 different bonds. Noteworthy at this point is also that most of the information found in a row is duplicate information that is also found in a different row. By formulating the problem in this manner it seems that it is tried to shift the task from time sequence modeling to a task where the development of the price over time is not represented in greater detail apart from the last 10 prices, types and sizes.

2.4 Effects of Economic Indicators on Government Bond Prices

Related work on bond price forecasting focuses mainly on the bond price development itself and does not include many external features for model building. This is often called a technical analysis. However, also the analyses seen in the related work do use additional information, which is not regularly used in technical analysis, e. g. the type of the buyer/seller. This already provides a hint that additional information for price forecasting might be helpful. This work will study the possibilities of forecasting bond prices and their trends with past bond price data enriched by economic data as bond price development is strongly connected to the development of (national) economy ([27]).

This section studies the effect of economic development on bond prices for the purpose of identifying relevant economic measures, which will be used as features in the later model design. First, before studying economic indicators with impact on bond prices, government bonds and especially the German bonds, as these are the ones, which will be used in the experiments for this work, will be explained. Once the essential information on bonds is provided, it will be investigated how bond pricing and (national) economic indicators intertwine. In this part of this work, this investigation is limited to related work, of which the majority has a finance or economics background. Later, when experiments will be conducted, a possible selection of features of the total number of economic indicators will be discussed.

2.4.1 Federal or Government Bonds

Federal or government bonds are debt securities issued by a government to support government spending ([36]). The exact terms and conditions for these bonds vary from government to government. What the majority has in common is the mechanism that an investor provides equity to the government, which agrees to pay back the exact amount agreed on at a later point in time. This alone would not be appealing for investors as they receive the exact amount of money back at the end of the debt when the maturity date is reached. In addition to the original amount, investors normally receive an annual interest rate payment, called coupon. This can be fixed or reliant on a different measure and is defined when the bond is handed out by the government ([33]). This issuing process is normally conducted by the federal bank of the corresponding government on a regular basis, several times a year. Most countries' federal banks publish a calendar, which includes fixed dates for the issues of bonds. Maturities and coupons of bonds can vary vastly, which is

another reason why this work focuses on German bonds (in addition to data availability). Investors with a long-term investment strategy (e.g. pension funds) use bonds frequently as a tool to secure their assets over long time as governments (at least a high number of them) do not have the reputation to go bankrupt, which makes them a rather non-risky investment. This can also be observed in the high number of countries that Moody's (one of the three large rating agencies) rates as **investment grade** ([50]). A similar situation could be found at one of the other well-known rating agencies but is not reported here ([83]). Another property, which is shared by bonds is that they are tradeable on the secondary market. This means that investors can not only fund the government directly by buying a bond from the government when issued but bonds can also be traded between investors, which is why they function similarly to stocks. Some bonds even allow a separate trading of coupon and investment.

2.4.2 Bond Price Influencing Indicators as Found in Related Work

The question,, which indicators influence the bond price development, has been around for a long time. Macaulay ([48]) started searching an answer to that question in the 1930s already and it is continued today by other researchers ([27]). Macauley focused mainly on meta information of the bond price like curvature or duration. Curvature and duration can be calculated directly from the properties of the bond ([48]). In addition, there are many other bond influencing indicators. A brief overview on the different indicator categories with examples, which are reported to influence bond prices are given below.

Technical information means any information either directly available with the bond like maturity or yield or information, which can be calculated from the properties of the bond like convexity, curvature or duration ([48]).

Base (or bank) rates are possibly the most influential indicators apart from technical information itself for bond prices ([48],[64]). Base rates are the interest rates, which are the rates used by a federal bank of a government or a monetary union to lend money to the banks within the monetary union. The bond price of a nation is however not only influenced by the base rate of the corresponding federal bank but also by base rates defined by federal banks of nations, which have a strong economic relation to each other. This effect is well investigated for the United States of America (USA) ([64]). For German bonds this is not as well studied. However, as

a nation, which is highly connected in global trade, it seems reasonable to assume a similar mechanism which would then assume following federal banks' base rates to be influential:

- European Central Bank (ECB)
- Federal Reserve (FED)
- Bank of England (BoE)
- Bank of Japan (BoJ)
- Schweizerische Nationalbank (SNB)
- Bank of Canada (BoC)
- Reserve Bank of Australia (RBA)

Credit ratings are also reported to have a major impact on bond pricing ([34]). However, the change of credit rating, which would then influence the bond price in a later model is a rather rare event. For the chosen domain of German bonds for example, the credit rating did not change for the last six years ([83]). This is why the credit rating might be less useful for this work but it is certainly highly impactful.

Financial market rates, similar to base rates, are rates, at which banks lend other banks money. Mostly, on a short term basis. Financial market rates are adapted on a daily basis and have the reputation of influence on bond prices as well ([48],[64]). The most important for the European and therefore for the German market are:

- EURIBOR and
- LIBOR

Stock indices as a indicator for the development of the economy in a nation or union of nations are also reported to impact bond prices ([66]). Applying this idea to the German bond would yield the following stock indices as relevant (possibly in this order) although it is not exactly reported, which of the following indices influences the price more or less strongly:

- DAX

- Dow Jones
- NASDAQ
- S&P 500
- EURO STOXX
- FTSE
- RTS
- Nikkei

Economic data publications , lastly, can have an impact on bond prices. There are various different types of economic data, of which the following are reported to influence either the US bond price or the German bond price ([1],[3],[24]) and therefore are candidates for a later model implementation:

- payroll
- industrial production
- new home sales
- durable goods orders
- the producer price index
- the consumer price index
- the consumer confidence index
- the ISM manufacturing index (formerly NAPM index)
- housing starts
- initial jobless claims
- Euro aggregated PMI (Purchasing Manager's Index)
- Euro aggregated PPI (Producer Price Index)
- Euro aggregated M3 (Measure of money supply)
- unemployment

- DE unemployment
- FR unemployment change
- FR industrial production
- FR business confidence
- FR consumer price index
- IT consumer price index
- durable goods orders
- housing starts
- initial jobless claims
- consumer confidence
- new home sales
- M2 medians
- Chicago PMI (purchasing managers index)
- consumer confidence
- employment cost index
- industrial production
- Michigan sentiment, final(University of Michigan Consumer Sentiment Index)
- real GDP, advance
- real GDP, final
- retail sales

3 German Federal Bond (Bund) Data

In this section, the reasons for choosing German bond prices as the domain for the model development are explained and the data, which will be used for training the model, is introduced including the sources of the data.

3.1 Choice for Bunds as Government Bonds

Limiting the scope of the models for bond price forecasting is a reasonable idea, as the general mechanism, which describes the bond price development, is yet unknown. Starting model development for this task right at the beginning with the most complex model trying to explain multiple or even all bond price developments in one model is not part of this work. However, at the moment, the goal of producing reasonable forecasts for bonds of a single country is sufficiently challenging.

This explains the choice for one country but the choice for German Federal Bonds (Bunds) is supported by another argument: practical considerations play an important role as well. For the model development there are two considerations, which need to be addressed. First, in addition to information on the general mechanisms of bond price development, it is desirable to find related work on the bond price development specifically for the chosen country. This will lead to countries, which have had a large economy over the last years, as the impression arises, that these are the ones economic research pays more attention to. Secondly and essential for any model development, will be the availability of a large amount of high quality data. There are possibly more candidates, which fulfill these criteria. Due to geographical vicinity the choice then fell on German Federal Bonds (Bunds).

3.2 Bund Data

As the data used for model development and feature selection and engineering is chosen, questions about the source of the data and the Bunds in general are to be answered in this section. The source of the data is the German Central Bank (Bundesbank), which publishes the latest information on Bunds on a daily basis.

As common for the majority of countries, Germany has a range of different securities, to which Bunds also belong. Countries normally offer different bonds, most often characterized by maturity, where a higher maturity yields a higher interest rate. The exact maturity then varies from country to country, starting at one year (e.g. USA: T-Bills ([75])) and ending at 50 years (e.g. France: Obligations assimilables du Trésor ([76])). Bunds have two different maturities, 10 and 30 years, of

which the 10 year Bunds account for the larger part of the Bund debt (17, [18]). Their prices are expected to be impacted by Germany’s economic development, since a major component of a bond price in general is the credit risk of the bond issuing government. On the Bundesbank’s website, the importance of Bunds is illustrated with the following figure about the share of total government debt between the different securities including the Bunds:

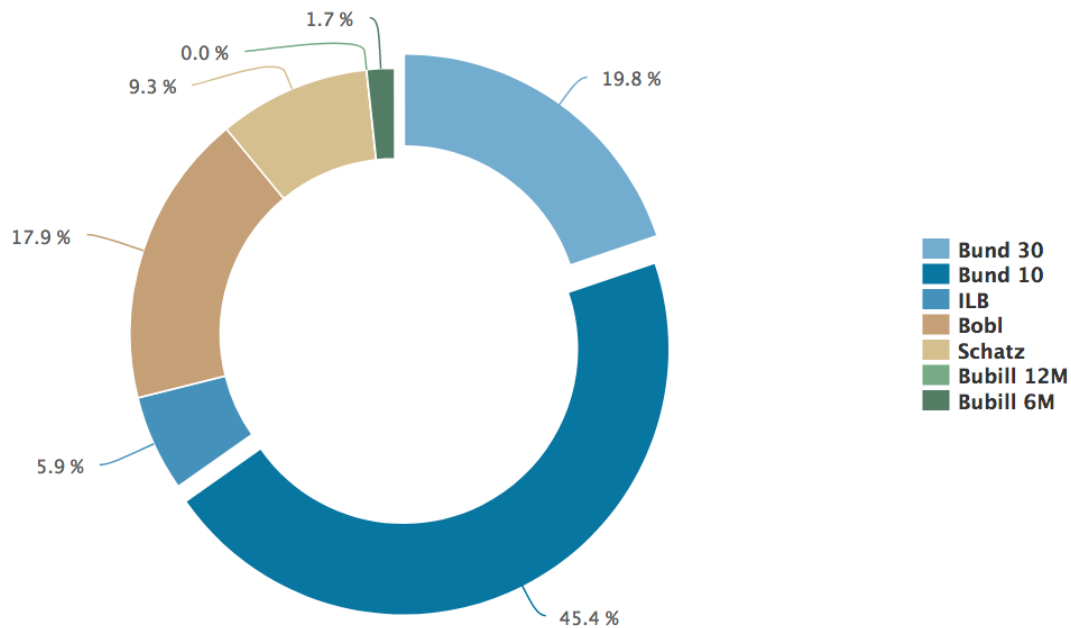


Figure 17: Shares by security of total German debt ([18]).

Another important point to make, before moving on to the exploratory analysis of the Bund data, is the distinction between primary and secondary market. The primary market is the market, in which a bond is issued from the government to the bond holder. As this situation, in which a Bund is issued, does not occur often (in the observed time span - which is discussed in the next paragraph - the maximum number of Bunds issued per year is five), this work focuses on the secondary market. The secondary market is the market, in which third parties can acquire bonds from other bond holders. The government is not involved in transactions of this market.

3.3 Exploratory Analysis of Bund Data

Now, that it is established, which data is used in particular and what different Bunds (Bund 10 and Bund 30) are included in this data, it seems reasonable to take a closer look at the selected data. The data available starts at the 3rd of January 2011 and ends at the 15th of February 2018. This results in around seven years of

Bund information. Bund information is provided on a trading-daily-basis, meaning that information is not available for seven days per week but rather for five days or even fewer days, in case there is a public holiday. Considering this, the total amount of trading days is 1797. For each of the 1797 trading days and for each active Bund, the following information can be retrieved on the website ([7]) on a daily basis:

- ISIN (International Securities Identification Number)
- Bezeichnung (name)
- Fälligkeit (maturity)
- Restlaufzeit (time to maturity)
- Emissionsvolumen (volume of issue)
- Kurs (price / rate)
- Rendite in Prozent (yield)
- Netto-Rendite in Prozent (net yield)
- Kurs plus Stückzinsen (price plus interest)

Overall, there are 61 Bunds active in the given time span. As some reach their maturity and others are issued in this time span, this equals 74963 data points. The whole time span ranges from the 3rd Jan 2011 to the 15th Feb 2018. The line graph on the next page illustrates the price development of every Bund in the available period per trading day.

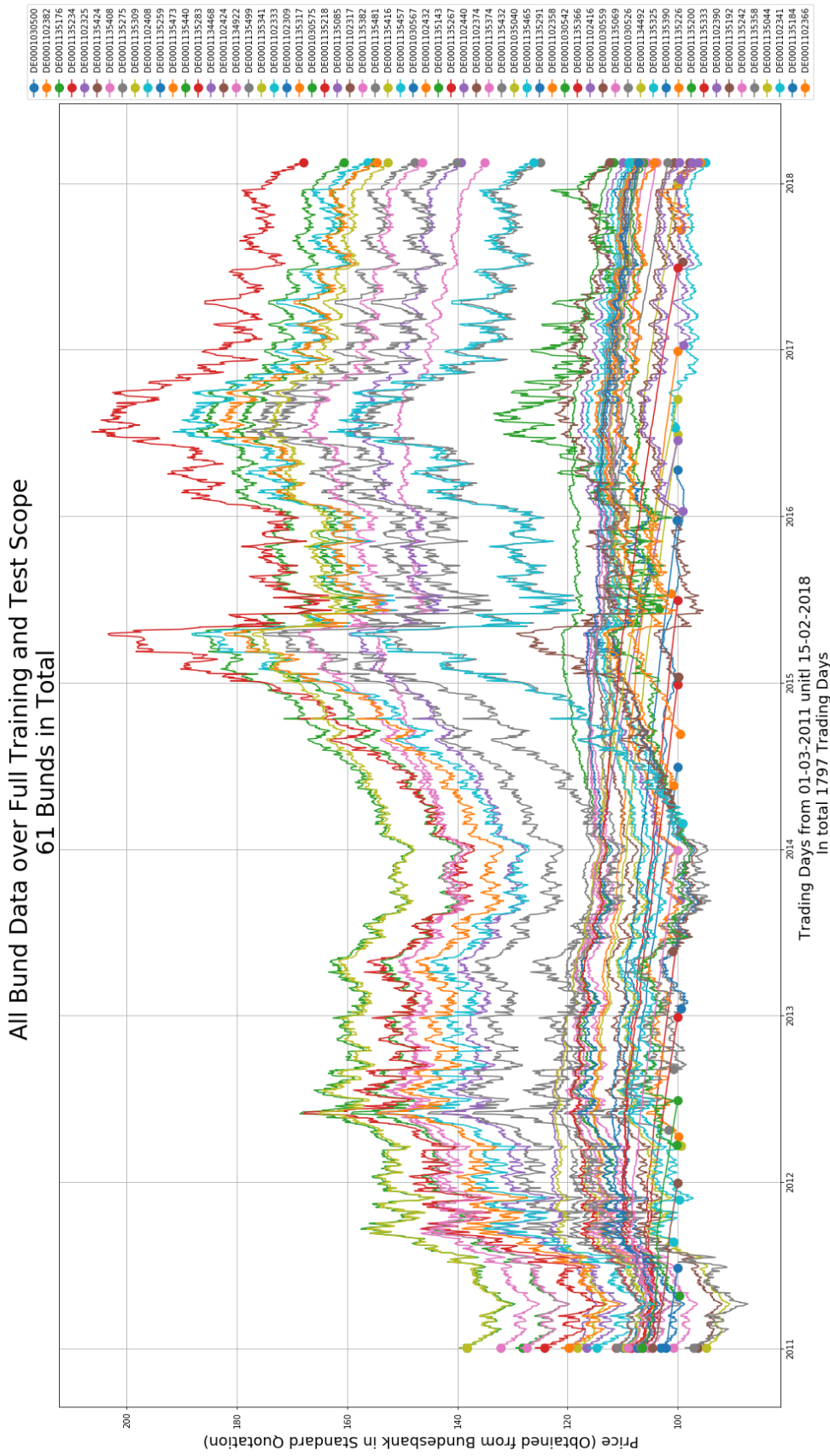


Figure 18: Bund 10 and 30 from 3rd Jan 2011 to 15th Feb 2018.

Over the whole time span, there is a minimum of 25 active Bunds and a maximum of 39 active Bunds per year. An overview of the number of active Bunds per year is given in the figure below:

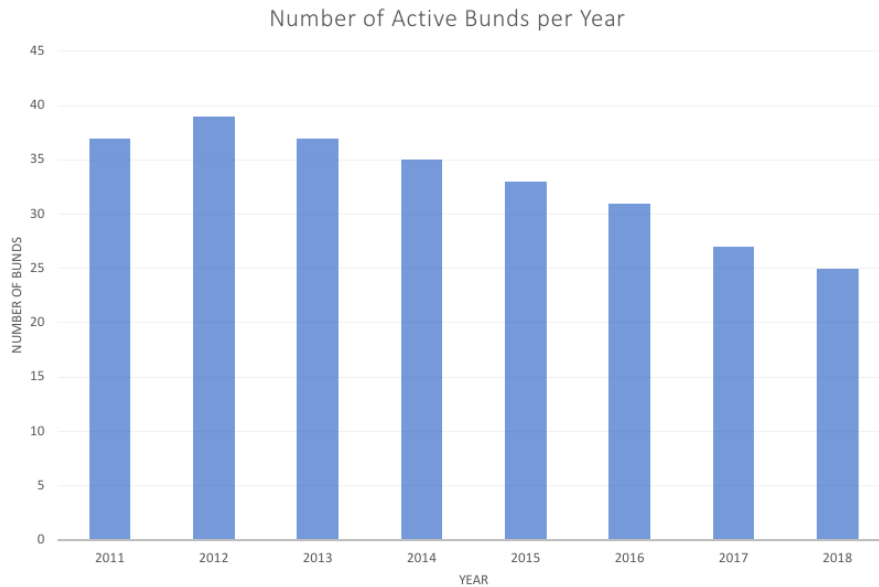


Figure 19: Active Bunds per year.

A closely linked question which is answered in the following graph is how many Bunds are issued and how many expire in each year. This changes from year to year but in the time span given, more Bunds are issued than expired per year on average.

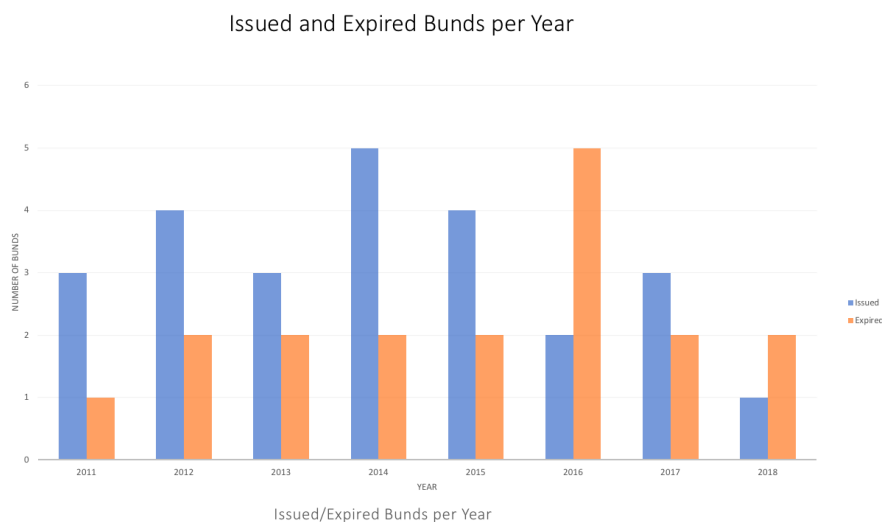


Figure 20: Issued/expired Bunds per year.

Not mentioned in the analysis of the data so far, has been the amount of 10 and 30 year Bunds. It is known, that both types of Bunds are in the sourced Bundesbank

data, but the exact shares have not been discussed. The following graph illustrates what might have been already guessed from the supreme importance of 10 year Bunds, that Bund 10 are more than twice as frequent as 30 year Bunds:

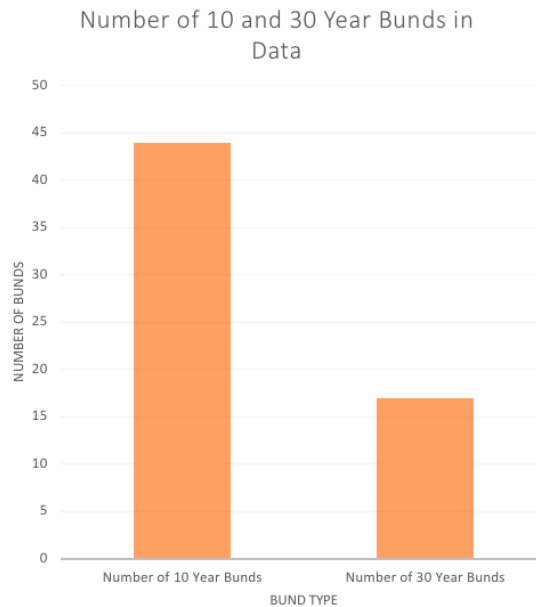


Figure 21: Share of 10 and 30 year Bunds over time span.

This fact, that the Bund 10 are represented in a higher number than the Bund 30 might be important, although the goal will be to develop one model for the two Bund types. This choice can be justified, because there should not be a difference between the two which cannot be explained by the fundamental data of the Bund. Otherwise, risk-less profits would be the consequences, since two identical products would be sold for two different prices. Further information on how Bund types are handled is presented in section 5.1.

In the following section, the economic indicator data available will be studied in more detail.

4 Macro-Economic Data

This section will describe the available data sources for macro-economic data. There are three natural resources available to this work to obtain macro-economic data for Germany, which are the governmental sources of the Statistical Office of the European Union (Eurostat) ([16]), the Federal Statistics Office of Germany (Statistisches Bundesamt) ([70]) and the commercial application Eikon by Thomson Reuters ([57]). The amount of economic data provided by the mentioned institutions is enormous. Not all data therefore is processed in this work. A reasonable pre-selection is essential to (a) not exhaust the means of this work and (b) also to sustain a certain level of interpretability. With the theoretical considerations on economic data, which effects bond price development, a promising starting point is set. In addition, Thomson Reuters provides a set of top economic indicators for economic development for most countries. The intersection between Thomson Reuters' top indicators for Germany and the theoretical considerations described earlier (2.4) will undergo the pre-selection process for promising features.

4.1 Economic Indicators with Relevance for Germany

Thomson Reuters' Eikon, a commercial financial analysis tool, provides extensive financial data for all kinds of institutions and also countries as well as monetary union. Both are of interest for this work with Germany as a member of the European monetary union. One feature, which is provided by Eikon, is the rating of all their available economic data. The most relevant - relevance defined by Thomson Reuters here - are then combined in so-called *top indicators* per country and monetary union. These indicators are not necessarily data from the country or monetary union itself. Also, data from interconnected economies can be found. For example, German top indicators include European data and vice versa. The full list of top indicators for Germany can be found in the table 1 on the next page.

Category	Name	RIC	Start Date	End Date	Source	Frequency
Consumer Sector	RETAIL SALES EXCL CARS (CAL ADJ) X-12-ARIMA	ADERLS/CA	01.01.94	1994 Jan 18	Federal Statistical Office, Germany	Monthly
Consumer Sector	RETAIL SALES EXCL CARS (CAL ADJ) X-12-ARIMA	ADERLXPVF/A	01.01.94	1994 Jan 18	Federal Statistical Office, Germany	Monthly
Consumer Sector	RETAIL SALES EXCLUDING CARS INDEX	ADERLSXMFV/C	01.01.94	1994 Jan 18	Federal Statistical Office, Germany	Monthly
Consumer Sector	POPULATION (1 DEC FROM 2003, PAN BD FROM 1991)	DEPOPOTOT	01.01.50	1990 2016	Federal Statistical Office, Germany	Annual
Consumer Sector	LEND TO DOM EXT+P&HH, OTH LOAN TO EMP&SOH+IND, TOTAL, ALL BNKS	DEDELEMPIND	10.01.68	1968 Q4 2017	Deutsche Bundesbank	Quarterly
External Sector	BOP CAPITAL & FINANCIAL ACCOUNT BALANCE (PAN BD M0790)	DEDCABRALA	01.01.71	1971 Jan 18	Deutsche Bundesbank	Monthly
External Sector	CURRENT ACCOUNT BALANCE	DEDCURAC	01.01.71	1971 Jan 18	Deutsche Bundesbank	Monthly
External Sector	BOP: EXPORTS FOB	ADEXP/A	01.01.91	1991 Jan 18	Deutsche Bundesbank	Monthly
External Sector	BOP: EXPORTS CIF	ADEMP/A	01.01.91	1991 Jan 18	Deutsche Bundesbank	Monthly
External Sector	BOP: VISIBLE TRADE BALANCE	ADEXTBAL/A	01.01.91	1991 Jan 18	Deutsche Bundesbank	Monthly
External Sector	EXPORTS OF GOODS (FOB)	ADEXPGBS/A	01.01.62	1962 Jan 18	Deutsche Bundesbank	Monthly
External Sector	IMPORTS OF GOODS (CIF)	ADEIMPGBS/A	01.01.62	1962 Jan 18	Deutsche Bundesbank	Monthly
External Sector	VISIBLE TRADE BALANCE	DETBAL=ECI	01.01.62	1962 Jan 18	Deutsche Bundesbank	Monthly
External Sector	INTERNATIONAL RESERVES	DEIFRGRES	12.01.98	1998 Feb 18	Deutsche Bundesbank	Monthly
External Sector	TOTAL EXPORTS OF GOODS	ADEEXPGBS/A	01.01.71	1971 Jan 18	Deutsche Bundesbank	Monthly
External Sector	TOTAL IMPORTS OF GOODS	ADEIMPGBS/A	01.01.71	1971 Jan 18	Deutsche Bundesbank	Monthly
External Sector	GERMAN MARKS TO US\$ (MTH AVG.)	DEXRUSD	01.01.57	1957 Feb 18	Bank of England	Monthly
Industry Sector	PRODUCTIVITY: OUTPUT PER MAN-HOUR WORKED, M-Q&MFG SCT (B+C)	DELPOTMHW/CA	01.01.91	1991 Dec 17	Deutsche Bundesbank	Monthly
Industry Sector	ULC PER UNIT OF TURNOVER ON HRLY BASIS (CHAIN-LINKED)	ADEPRLCPRUN/CA	01.01.91	1991 Q4 2017	Deutsche Bundesbank	Quarterly
Industry Sector	INDL PROD: INDUSTRY INCL CNSTR (CAL ADJ)	ADEIP/CA	01.01.91	1991 Jan 18	Federal Statistical Office, Germany	Monthly
Industry Sector	INDL PROD: MANUFACTURING (CAL ADJ)	ADEIPMAN/CA	01.01.91	1991 Jan 18	Federal Statistical Office, Germany	Monthly
Industry Sector	MANUFACTURING ORDERS	ADENOMFG/A	01.01.91	1991 Jan 18	Deutsche Bundesbank	Monthly
Industry Sector	MANUFACTURING ORDERS (CAL ADJ)	ADENOMFG/CA	01.01.52	1952 Jan 18	Deutsche Bundesbank	Monthly
Industry Sector	INSOLVENCIES - BUSINESS ENTERPRISES	ADENSOLPE	01.01.75	1975 Dec 17	Federal Statistical Office, Germany	Monthly
Labour Market	EMPLOYED PERSONS (RESIDENCE CONCEPT, ILO)	ADEEMPDE/A	01.01.92	1992 Jan 18	Bundesagentur für Arbeit, Germany	Monthly
Labour Market	UNEMPLOYMENT REGISTERED (PAN BD FROM JAN 1992) (CAL ADJ)	DEUEMP=ECI	12.01.91	1991 Feb 18	Deutsche Bundesbank	Monthly
Labour Market	UNEMPLOYMENT: % CIVILIAN LABOUR (% DEPENDENT LABOUR TO DEC 1968)	ADEUNR	01.01.50	1950 Feb 18	Bundesagentur für Arbeit, Germany	Monthly
Labour Market	WAGE & SALARY: OVERALL ECONOMY - ON A MONTHLY BASIS (PAN BD M0191)	ADEWSTOT	01.01.60	1960 *Jan 2018	Deutsche Bundesbank	Monthly
Labour Market	WAGE & SALARY: ON HRLY BASIS - PRDG. SECTOR (BDHRVAGEF)	ADEWCSLHRBS/C	01.01.60	1960 *Jan 2018	Deutsche Bundesbank/Thomson Reuters	Monthly
Labour Market	EMPLOYED PERSONS (RESIDENCE CONCEPT) (%YOY)	DEWEMPTOO/A	01.01.93	1993 Jan 18	Thomson Reuters	Monthly
Money & Finance	MONEY SUPPLY M0	ADEM0	01.01.80	1980 Jan 18	Thomson Reuters	Monthly
Money & Finance	MONEY SUPPLY - GERMAN CONTRIBUTION TO EURO M1 (PAN BD M0790)	ADEMLAB	01.01.73	1973 Jan 18	Deutsche Bundesbank/Thomson Reuters	Monthly
Money & Finance	MONEY SUPPLY - M2 (CONTRIBUTION TO EURO BASIS FROM M0195)	ADEM2BC/A	01.01.60	1960 Jan 18	Deutsche Bundesbank/Thomson Reuters	Monthly
Money & Finance	MNY. SUPL - M3 (CONTRIB TO EUR BASIS FM. M0196), FM M06 2010 EXC	ADEM3ABC/A	01.01.69	1969 Jan 18	Deutsche Bundesbank/Thomson Reuters	Monthly
Money & Finance	DISCOUNT RATE / SHORT TERM EURO REPO RATE	ADEPRATE	07.01.50	1950 Mar 18	ECB - European Central Bank	Monthly
Money & Finance	LONG TERM GOVERNMENT BOND YIELD - 9+10 YEARS	ADEGBOND	01.01.57	1957 Feb 18	Datstream	Monthly
Money & Finance	DAX SHARE PRICE INDEX, EP	ADESHRPRCF	09.01.59	1959 Mar 18	Reuters	Monthly
Money & Finance	LENDING TO ENTERPRISES & INDIVIDUALS	ADEBANKIPA	01.01.50	1950 *Jan 2018	Deutsche Bundesbank	Monthly
National Accounts	GDP	ADEGDP/CA	01.01.91	1991 Q1 2017	Federal Statistical Office, Germany	Quarterly
National Accounts	CONSUMER EXPENDITURE	ADEGPC/CA	01.01.91	1991 Q4 2017	Federal Statistical Office, Germany	Quarterly
National Accounts	GOVERNMENT CONSUMPTION	ADEGEX/CA	01.01.91	1991 Q4 2017	Federal Statistical Office, Germany	Quarterly
National Accounts	FIXED INVESTMENT	ADEGFC/CA	01.01.91	1991 Q4 2017	Federal Statistical Office, Germany	Quarterly
National Accounts	CHANGE IN STOCKS	ADEGFCHG	01.01.91	1991 Q4 2017	Federal Statistical Office, Germany	Quarterly
National Accounts	EXPORTS OF GOODS & SERVICES	ADEEXP/CA	01.01.91	1991 Q4 2017	Federal Statistical Office, Germany	Quarterly
National Accounts	IMPORTS OF GOODS & SERVICES	ADEIMP/CA	01.01.91	1991 Q4 2017	Federal Statistical Office, Germany	Quarterly
National Accounts	GNI	ADENNG/A	01.01.91	1991 Q4 2017	Deutsche Bundesbank	Quarterly
National Accounts	IPD OF GDP	ADEGDPDEF/A	01.01.91	1991 Q4 2017	Thomson Reuters	Quarterly
National Accounts	PERSONAL SAVINGS RATIO (PAN BD Q0191)	ADEGPS/A	01.01.60	1960 Q4 2017	Deutsche Bundesbank	Quarterly
National Accounts	DISPOSABLE INCOME (PAN BD Q0191)	ADEDIPNC/A	01.01.60	1960 Q4 2017	Deutsche Bundesbank/Thomson Reuters	Quarterly
National Accounts	NATIONAL INCOME: ENTREPRENEURIAL & PROPERTY INCOME	ADEGEPTPIN/A	01.01.91	1991 Q4 2017	Deutsche Bundesbank	Quarterly
Prices	CPI (CAL ADJ)	ADCECPI/A	01.01.50	1950 Feb 18	Deutsche Bundesbank	Monthly
Prices	CPI: TOTAL	ADCECPI	01.01.50	1950 Feb 18	Federal Statistical Office, Germany	Monthly
Prices	HICP: TOTAL	ADEHICP	01.01.96	1996 Feb 18	Federal Statistical Office, Germany	Monthly
Prices	EXPORT PRICE INDEX	ADEEXP	01.01.62	1962 Jan 18	Federal Statistical Office, Germany	Monthly
Prices	IMPORT PRICE INDEX	ADEIMP	01.01.62	1962 Jan 18	Federal Statistical Office, Germany	Monthly
Prices	PPI: INDUSTRIAL PRODUCTS, TOTAL, SOLD ON THE DOMESTIC MARKET	ADEPPI	01.01.50	1950 Jan 18	Federal Statistical Office, Germany	Monthly
Surveys	BUSINESS EXPECTATIONS (PAN GERMAN)	DEBUS=ECI	01.01.91	1991 Feb 18	Ifo - Institute for Economic Research, University of Munich	Monthly
Surveys	COMPOSITE LEADING INDICATOR - TREND RESTORED	ADECLCAD	01.01.61	1961 Dec 17	Main Economic Indicators, copyright OECD	Monthly
Surveys	CONSUMER CONFIDENCE INDICATOR - GERMANY	DEBCFM/A	01.01.85	1985 Feb 18	DG ECFIN - Directorate General for Economic and Financial Affairs	Monthly
Surveys	TRADE & IND: BUS CLIMATE, INDEX	DEBUS=ECI	01.01.91	1991 Feb 18	Ifo - Institute for Economic Research, University of Munich	Monthly

Table 1: Reuters top economic indicators for Germany.

4.2 Feature Engineering: Economic Indicators as Additional Features

One essential step to later improve traditional forecasts must be the selection of relevant economic features of the above mentioned. The major aim of this selection is to avoid over-fitting, improve training duration and interpretability. As literature does not specify, how development of a national economy, the world economy in total or other influential economies effect bond prices in greater detail, this needs to be investigated for later model development. Although research is vague about effects of the majority economic indicator, some must be seen as influential, e.g. the central banks' base rates. The investigation of economic indicator relevance in this work is limited by the nature of the bond data as time series data. Prominent feature selection methods like Principal Component Analysis (PCA) are not intended to be used for time series data. That is the reason why in this work correlations between economic indicators and bond price are the main criterion for feature selection. In addition, correlations between economic indicators themselves will be tested to avoid the inclusion of redundant information to the later model building. Detailed information on the feature selection is provided within the next paragraphs.

4.2.1 Economic Indicators and Bond Price Correlations

The first step to identify relevant economic indicators is to test correlations between the indicators and the bond price. Since there is a large number of Bunds, for which correlations between indicators are tested, the average of the correlations for each Bund and economic indicator is used as the result for the correlation between Bund prices in general and the specific economic features.

The identification of possibly relevant economic features then has been conducted in three parts. First, the overall correlations were calculated, meaning how Bund prices and economic features correlate over the whole time span in which the Bunds are active. Second, what is denoted here as short-term and long-term correlations has been calculated. This is motivated by the idea that certain economic features might effect the Bund price only when the remaining time to maturity is short or long. Short-term correlations were then identified on Bunds with a maturity less than two years. Long-term correlations on the other hand, were set to be those with a time to maturity longer than two years. This distinction might seem arbitrarily to the reader and although it is partially true that the distinction could have been chosen to be made otherwise. The results of the correlations of Bund price and economic features in the following tables of overall correlations (2), short-

term correlations (3) and long-term correlations (4) indicate that the decision is reasonable. This might need additional explanation which can be found in the remaining lines of this subsection.

Overall Correlations The overall correlations table (2) does not seem promising at first with no economic features correlating strongly to the Bund prices, and the ten highest correlations between 0.467 and 0.481. However, the economic features, which would be expected to occur in this table, when consulting economics research can be found, e.g. ECB policy rates, labour costs and consumer prices.

Econ. Indicator (RIC)	Avg. Correlation	Description
aDEPRLCPRUN_CA	0.481	Unit labour cost, hourly basis (Germany)
aXZBIDR	0.478	Euro zone, ECB main fixed rate
aDECPLA	0.475	Consumer price index, seasonally adjusted (Germany)
aDEHICP	0.475	Harmonised consumer price index (Germany)
aDECPI	0.472	Consumer price index (Germany)
sp500_low_ld	0.470	Lowest rate of S%P 500 of last day of the month
aDEINSOLPE	0.469	Insolvencies, businesses (Germany)
dax_tief_avg	0.468	Average (month) of lowest DAX rate for each day
dax_erster_avg	0.467	Average (month) of opening DAX rate for each day
dax_hoch_avg	0.467	Average (month) of maximum DAX rate for each day

Table 2: Average absolute overall correlations between Bund price and economic indicators (identified by RIC=Reuters Instrument Code).

As the low number of correlations contrasts many work in economics as introduced earlier (2.4,) which does find strong effects of certain economic features to government bond price development, this leads to the idea that this overall calculation of correlation might be too demanding. Possibly, different effects can be observed with different remaining times to maturity. The following two tables of short term correlations (3) and long-term correlations (4) support this claim.

Short Term Correlations In short-term correlations, identified correlations climb until 0.710 for ECB deposit rates which is closer to the expectations from an economic point of view. However, remaining economic indicators do not meet the expectations in short-term correlations as can be seen in table (3).

Econ. Indicator (RIC)	Avg. Correlation	Description
EUECBD=ECI	0.710	Euro Zone, ECB deposit rate
aDELPOUTMHW_CA	0.490	Labour productivity (Germany)
DEZEW=ECI	0.464	Current economic situation (Germany)
aDEEMPTOO_A	0.451	Employed persons (Germany)
DEBUSS=ECI	0.429	Business climate (Germany)
euro_stoxx_vol_last_day	0.415	EURO STOXX volume last day of month
aDEBUCFM_A	0.413	Consumer confidence (Germany)
aDEGPS_A	0.386	Personal savings (Germany)
NASDAQ_vol_ld	0.352	NASDAQ volume last day of month
aDEOLEMPIND	0.331	Loans to domestic employees (Germany)

Table 3: Average absolute short term correlations between Bund price and economic indicators (identified by RIC=Reuters Instrument Code, if not stock index such as DAX, NASDAQ or STOXX).

Long Term Correlations Testing for long term correlations, the correlations rise even higher to a maximum of 0.792 for the ECB main fixed rate. In contrast to short-term correlations, long-term correlations are also rather high for a variety of other indicators like labour cost, insolvencies and consumer prices for example. Details can be found in the following table (4).

At the end, these results do not come as surprising as one might think. Due to the nature of Bunds, their long-term investment character with very little risk involved, it seems reasonable, that short-term changes in the economy effect the Bund price differently than long-term changes. Slight economic changes do not seem to tangle the Bund price significantly on a short-term, as the risk can be considered low due to

Econ. Indicator (RIC)	Avg. Correlation	Description
aXZBIDR	0.792	Euro Zone, ECB main fixed rate
aDEPRLCPRUN_CA	0.779	Unit labour cost, hourly basis (Germany)
aDEINSOLPE	0.773	Insolvencies, businesses (Germany)
aDEPRATE	0.770	Policy rate, discount rate (Germany)
aXZECEB	0.770	Policy rate, ECB main refinancing rate
aDECPLA	0.767	Consumer price index (Germany), seasonally adjusted
aDECPI	0.764	Consumer price index (Germany),
aDEHICP	0.763	Harmonised consumer price index (Germany)
aDEGDPDEF_A	0.760	Implicit price deflator (Germany)
aDEM1AB	0.757	Domestic finance, money supply (Germany)

Table 4: Average absolute long term correlations between Bund price and economic indicators (identified by RIC=Reuters Instrument Code).

the fact, that the maturity soon will be reached. On the contrary, for long remaining times to maturity, these small changes might disturb the Bund price. This is caused, since risk might be seen as increasing due to the long remaining time to maturity and possibly evolving economic problems of the bond issuing government.

This limited investigation of economic effects on Bond prices showed that

1. the effects of economic indicators as claimed by economists (2.4) can be observed in the data for German Bund price development
2. the effects are different depending on the remaining time to maturity
3. there are promising economic features, which can be later used for model building

At this point, it remains open, how the economic indicators themselves are intertwined with each other. This question will be necessary to answer to achieve the actual goal of this chapter of identifying the indicators, which will be later used for model building.

4.2.2 Correlations between Economic Indicators

To study the relation between the economic indicators themselves, their correlations to each other are computed. This will help to exclude superfluous information by later including multiple economic features in a model, which carry the same informational value. In the following heat map (22), the highest short-, long-term and overall correlations calculated in the subsection prior to this are combined and their correlations to each other are depicted. Red indicates high correlations, whereas blue means hardly any correlation. Note that stock price information such as NASDAQ, EURO STOXX and DAX are not included in this visualization.

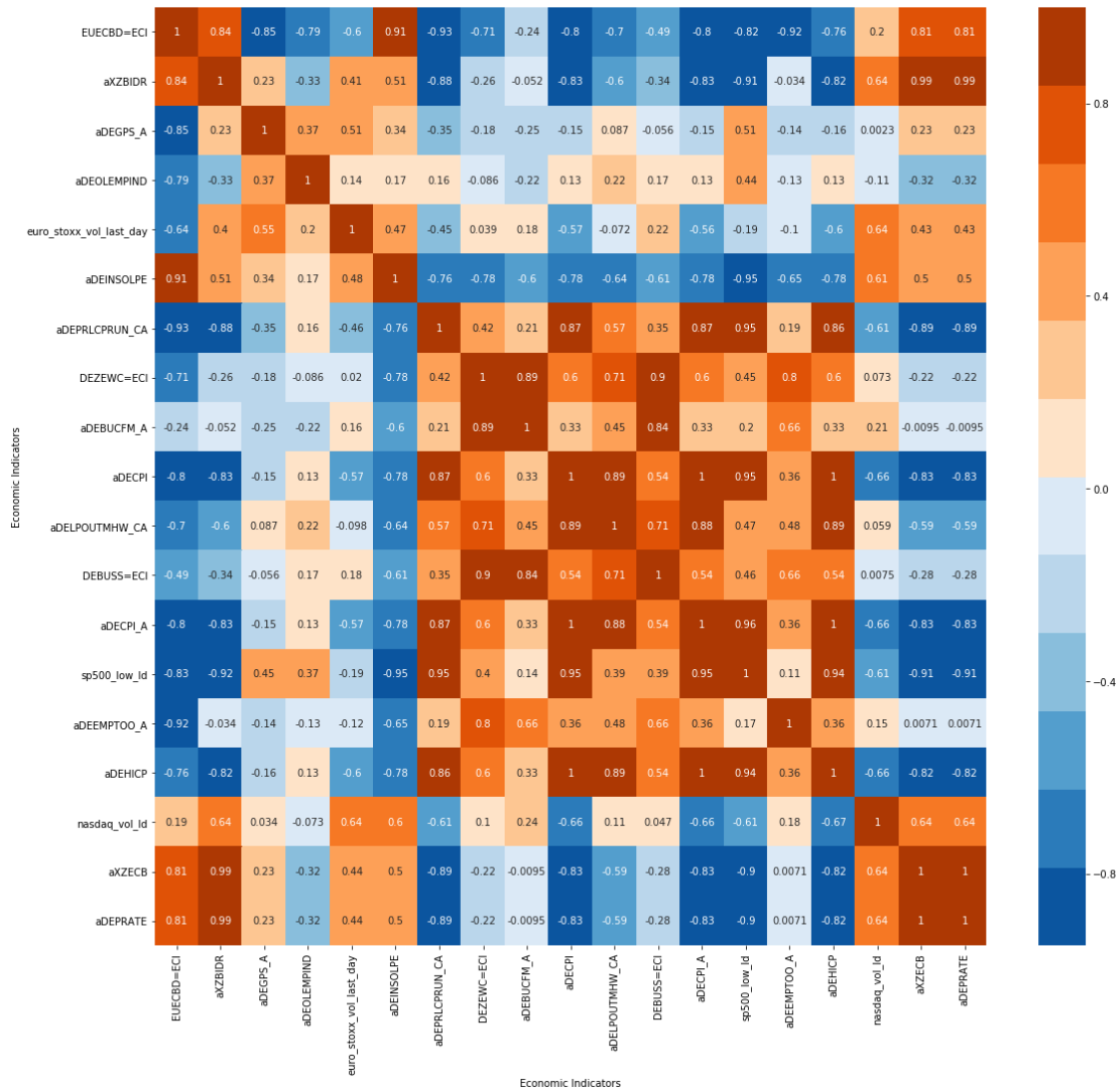


Figure 22: Correlations between top 10 overall, top 10 short term and top 10 long term economic indicators.

Now, as the relations between the economic indicators themselves become clearer, this helps to identify relevant indicators for later modelling. It is important to note, that not every blue mark in the map is useful or relevant. It needs to be kept in mind, that the whole column (or row) needs to be considered to evaluate an indicator's relevance. How indicators are selected at the end is discussed next.

4.2.3 Ranking the Relevance of Economic Indicators

Since the selection process of this work is certainly not the only reasonable approach and there might be others which are even more promising, two different sets of economic indicators will be tested in model building later. There are good reasons to further test different indicator sets, but this would expand this already extensive experimental work even more. In addition, the assumption is, that this should suffice to study the advantages and disadvantages of adding economic features to Bund forecasts. This is why only two different sets of economic indicators are introduced in the following.

Four economic indicators The first approach and a rather straightforward idea is to simply use the three economic indicators from short- and long-term and overall correlations with highest correlation to Bund prices. This would lead to including *Unit Labour Cost*, *ECB Deposit Rate* and *ECB Main Refinancing Rate* in the model. This also complies with economic research on bond price development. However, *ECB Deposit Rate* and *ECB Main Refinancing Rate* strongly with each other, as can be seen in the heat map of the prior subsection (22). Therefore, the indicator with the second highest correlation of the long-term correlations, *Insolvency Proceedings* is included as well. An overview of then four indicators can be found in the table (5) below.

Econ. Indicator	Description	Correlation
aDEPRLCPRUN_CA	Unit Labour Cost, per Unit of Turnover (Hourly)	0.481 (Overall)
EUECBD=ECI	Policy Rates, ECB Deposit Rate	0.708 (Short Term)
aXZBIDR	Policy Rates, ECB Main Refinancing Rate	0.792 (Long Term)
aDEINSOLPE	Insolvency Proceedings, Enterprises, Total	0.773 (Long Term)

Table 5: Selection of four promising features for Bund price forecasting.

Another interesting fact, which has been addressed only indirectly yet, is the question of how these economic indicators develop over time. It is already known, that they correlate to Bund prices and also how Bund prices develop over the time (3). However, the precise development of these indicators has not been looked into yet. This development is depicted in figure (23). Note that the y-axis is scaled to the range (0,1). This is the same min-max-scaling which will be used later in the model building. As the original units of these indicators are neither self-explanatory nor share the same units, this visualization has been chosen. It is believed, that it does not lose the relevant information about the development of the indicators.

Seven economic indicators The second set of economic indicators is selected differently. Here, the seven fields with minimal correlation ("most blue fields") of the heat map of indicator indicator correlation were selected. At the same time, fields need to have a correlation to the Bund price development of at least 0.5. This approach tries to combine the two ideas of having (a) a stronger correlation between Bund prices and indicators and (b) choosing those indicators, where correlation to others is not strong to not include already available information.

The choice for seven economic indicators then remains to be explained. Again, this could be criticized as arbitrarily, but within the experimental nature of this work this seemed reasonable due to the following reasons. (a) The selection of the seven indicators included a variety of economic data, which seems relevant because of the prior correlation analysis and in related, economic work, for example labour cost, consumer prices, ECB rates and insolvencies. (b) Selecting many more than the now selected seven indicators might result in lower interpretability. By limiting the number of indicators to seven here, it is believed to keep the later models simple

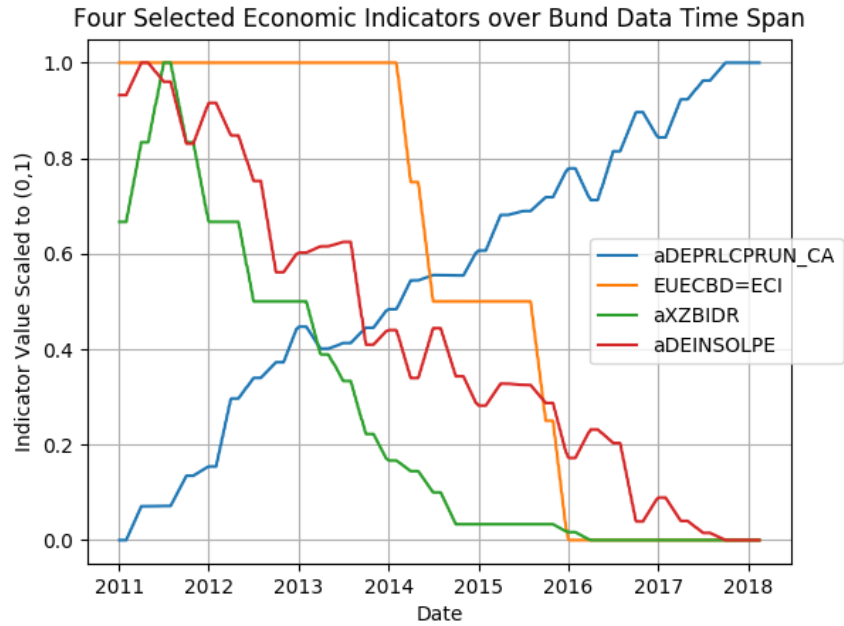


Figure 23: Development of four selected indicators over Bund data time span.

enough to understand the relevance each indicator plays at the end in forecasting Bund development. An overview of the second set of economic indicators to be tested is found in figure (6).

Econ. Indicator	Description	Correlation
aDEPRLCPRUN_CA	Unit Labour Cost, per Unit of Turnover (Hourly)	0.778
aDEHICP	Harmonised Consumer Prices	0.763
aDECPIA	Consumer Prices	0.764
aDEPRATE	Policy Rates, Short Term Discount Rate	0.770
aXZBIDR	Policy Rates, ECB Main Refinancing Rate	0.792
aXZECEB	Policy Rates, ECB Main Refinancing Rate	0.770
aDEINSOLPE	Insolvency Proceedings, Enterprises, Total	0.773

Table 6: Selection of seven promising features for Bund price forecasting.

Also for this set of indicators, the development is depicted (24). The visualization is created in the same way it has been for the set of four indicators (23).

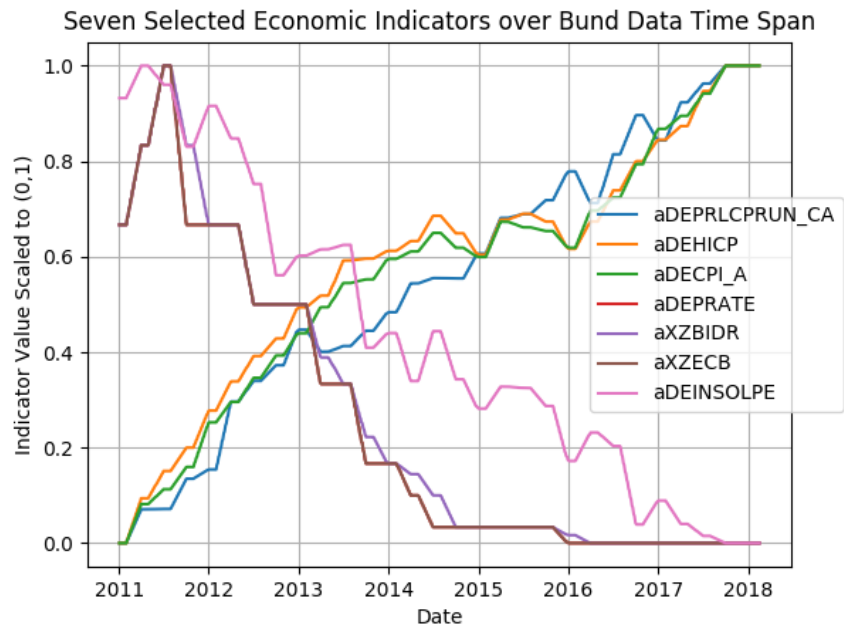


Figure 24: Development of seven selected indicators over Bund data time span.

5 Experimental Setup

In this section, the different experimental setups will be discussed. This is addressed by the following:

1. The available data will be recapitulated and the fusion of Bund data and selected economic data will be introduced (5.1).
2. The forecasts will be done with two different time scopes. One experimental setup will illuminate the next day forecast, whereas the other one investigates next week forecasts. Further explanation on the chosen time horizons can be found in the subsection *Forecasting with Different Time Horizons* (5.2.1).
3. Additional experiments are conducted with different targets. On the one hand, models for price forecasts will be implemented. On the other hand, the return, the difference between price at day t and price at day $t - 1$, will be modelled. Detailed explanation on this choice can be found in the subsection *Forecasting with Different Targets* (5.2.2).
4. In the next part of this section, the choice of implemented models will be explained, ranging from simple models to more complex neural network based approaches. Details in section *Model Choice for Forecasts* (5.3).
5. The four last subsections of this section are rather brief, but they are concerned with relevant choices, such as
 - which features can be incorporated in the various models (5.4)?
 - how are the various models expected to generalize (5.5)?
 - the introduction of the rolling forecast technique (5.6).
 - how are model performances at the end evaluated (5.7)?

Soft- and Hardware Tools Used in this Work Soft- and hardware have not been discussed so far, and they will not be addressed in greater detail. Nonetheless, a brief note, on which tools were used throughout the development of this work, seems appropriate at this point.

This work includes a rather extensive experimental section running different methods with different configurations on different data. Nevertheless, memory is not a problem as data sets are still small. However, especially neural network based approaches are computationally costly and therefore most of this computation has

been conducted on various different servers of the DFKI's multimedia analysis and data mining group. They were equipped with 64 to 512 GB of ram group running on Open Suse, equipped with one to eight NVIDIA GTX 1080 in different versions. Coding of the models has been done in Python 2 and 3. Parts of the evaluation were done in Python 2 and 3 as well as in R and MS Excel. Development of Python code was conducted in Visual Studio Code (Version 1.27.2), while R code was written in RStudio (Version 1.1.383).

5.1 Training and Test Data

In this subsection, the available data will be discussed as well as the decision of how to split the data into training and test. This is done in three parts:

1. The available information on the Bund data as seen earlier will be shortly revised (5.1.1).
2. The economic data discussed earlier will be reviewed in the context of how this can be used for experimentation (5.1.2).
3. The available and the chosen methods to fuse the data sources of Bund data and economic data will be addressed (5.1.3).

5.1.1 Bund Data

The Bund data which has already been discussed in detail in a separate section (3) is the basis for any other additional experiments with more data. This data set contains the price development over time, which this work aims to model and derive reasonable forecasts from. The reader is referred to the corresponding section (3) for more details on the data itself. Here, the focus is placed on the question of how to use this data set for a supervised training of the selected models (model selection is discussed later in this section (5.3)). At the end, this boils down to the answer of two major questions:

1. How is the data set split into training and test data?
2. Are the many different Bund time series all modelled by one model, by a number of different models or maybe one model for each time series?

In the following, the two questions will be answered in the order they have been posed. First, the choice of training and test data tries to mimic the real-world scenario, in which the future is unknown. This means, that at a certain point

of time, every information in the past will be known and therefore used for the supervised training. Every information in the future from that point of time will be unknown and used for testing the obtained model. For certain models, there are additional choices for a validation set available. These details are described in depth, when choices for the different models and their experimental setup is explained (5.3).

In this work, the point of time to divide data into training and test data has been set to the **1st of February 2017**. Every information before the 1st of February 2017 in time (starting at the 3rd of January 2011 as the 1st was not a working day) then is consequently training data, every information after (and including) the 1st of February 2017 in time (ending at the 15th of February 2018) then is test data. Why is this specific date then chosen, is a reasonable question to ask. The answer to that is, that the choice for the exact date is supported by arguments of training and test data size. Still, including one additional one month in the training or test set or vice versa would still be a reasonable choice. Setting the 1st of February 2017 as the point of time where to split the Bund data into training and test results in 63656 data points in the training data and 11307 data points in the test data. Additional information can be obtained in the following table 7.

	training data	test data
Number of Bunds	58	46
Average data points per Bund	1043	185
Median data points per Bund	1216	261
Maximum data points per Bund	1536	261

Table 7: Additional information on training and test data.

On the next two pages, all Bund data is depicted over the whole time span with the split into training and test data on the 1st of February 2017. The first page covers Bund price development, whereas the second one addresses Bund return development.

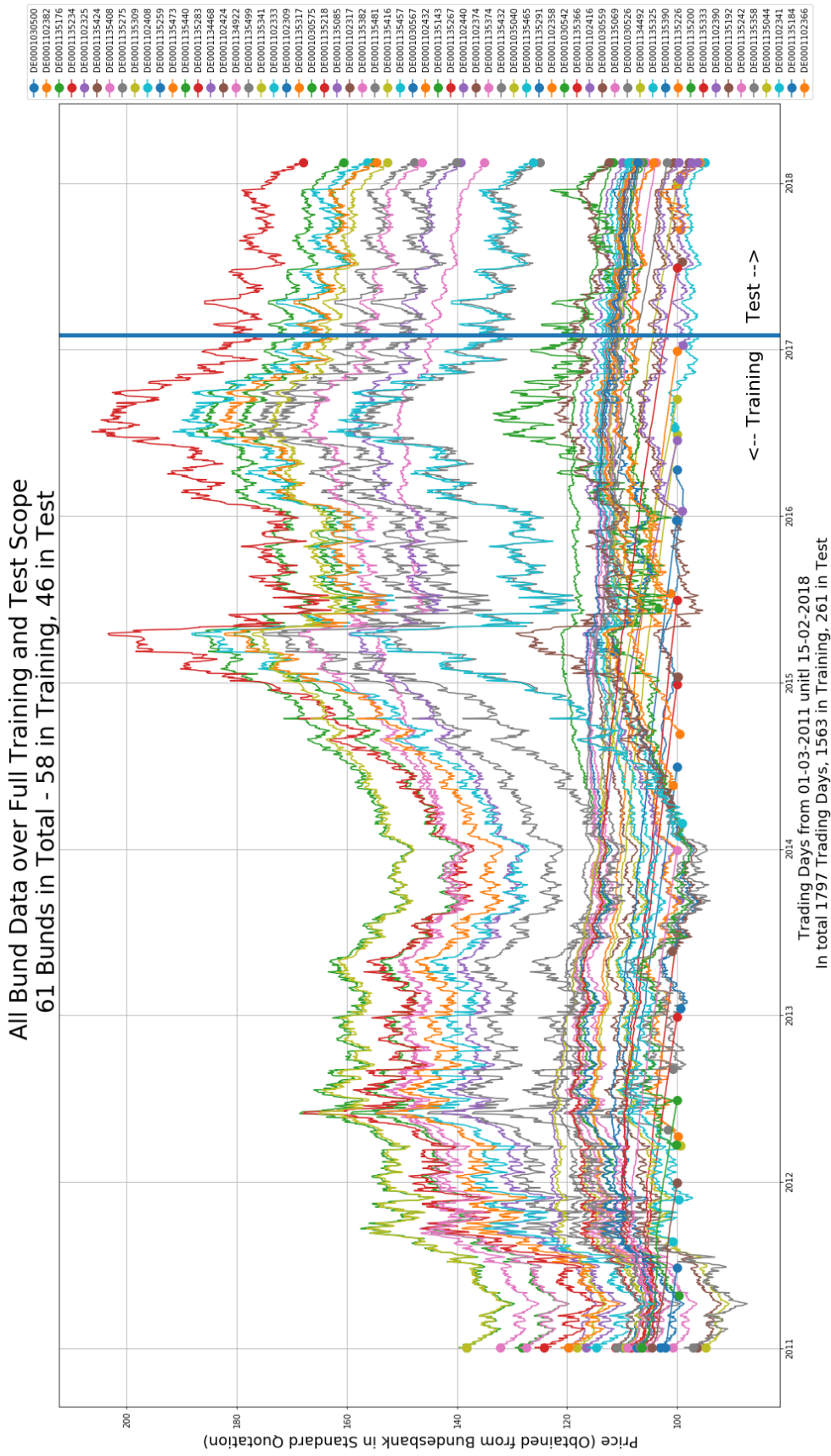


Figure 25: Prices: Bund 10 and 30 from 3rd Jan 2011 to 15th Feb 2018.

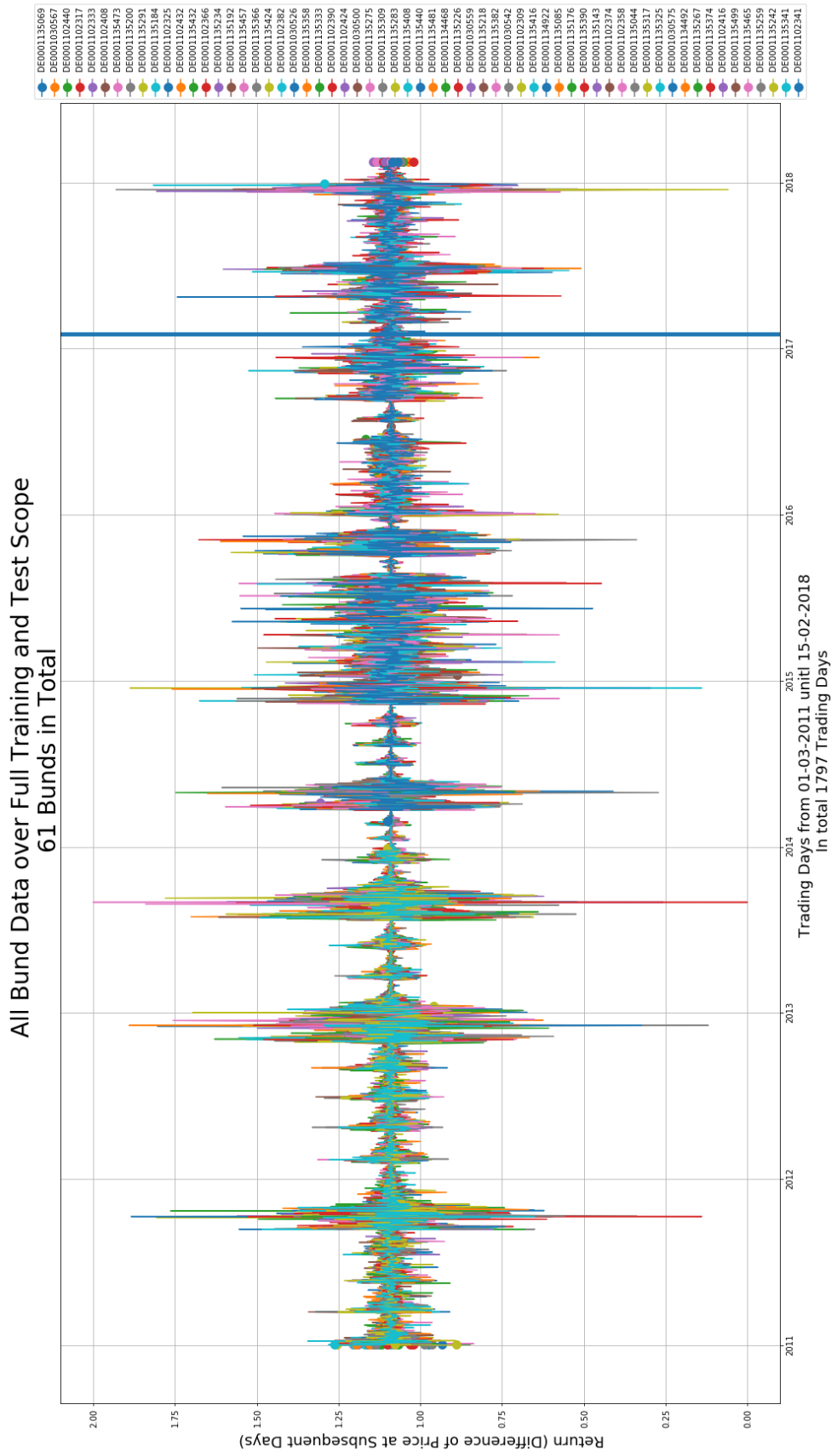


Figure 26: Returns: Bund 10 and 30 from 3rd Jan 2011 to 15th Feb 2018, scaled to range (0,2).

Second, the question of how many models should be built for training is addressed. At the end, this question is answered, when the model selection is explained due to the difference of the variety of models. However, some more general remarks can be made here already. Due to the nature of the simpler models, for example the naive approach of forecasting the next day price/return to be equivalent to the current day, they are only reasonable to be trained for each Bund separately. For models which aim to acquire a more general understanding of how Bund price development works, this is not so clear. Possibly, one model might work well due to the fact that Bunds can be distinguished by their technical information (e.g. coupon, maturity, et cetera). This means that a model would be able to find a well suited abstraction of the larger number of training Bunds. On the contrary, one model for each Bund might be easier to learn due to the large discrepancies in price range of different Bunds. Another, third option, might be that the Bunds can be clustered or categorized in certain groups which share characteristics which can be learned. For example, the earlier formulated hypothesis (3), that it should not play a role, if a Bund has 10 or 30 year maturity as it is defined by its technical details, might not be entirely true - although reasonable arguments for this claim exist. For models, which can be reasonably trained on sets of Bunds, this work will report results for both extrema, training on all Bunds and training on every Bund. This should suffice to point into a direction, which is worth further investigation.

5.1.2 Macro-Economic Data

The macro-economic data, obtained via Thomson Reuters Eikon, described in further detail earlier (4), will be used as additional features to the technical data of the Bund data set.

Naturally, there are again several options of how this economic data could be integrated in the model implementation, of which the chosen option will be discussed. In this work, it has been decided to focus on a rather limited number of additional features (4). Experiments will be conducted with two different additional feature sets, which share the characteristics, that they are small in size. The first additional feature set only contains four additional, economic features. The second additional feature set contains seven, economic features. The analysis in chapter 4 led to the two feature sets listed in tables 5 and 6.

5.1.3 Fusion of Bund Data and Economic Data

In this work, data from two sources is combined. On the one hand, there is the Bund data, which is obtained from the German Federal Bank (Bundesbank) (3). On the other hand, there is economic data made available by Thomson Reuters (4). In this work, the two data sources are combined with an early fusion approach, meaning the combination of the two data sources before model building. Details on this data fusion are described next.

Technically, these two data sources are stored as data frames within Python. Rows of the Bund data contain the price for a Bund on a specific date (on a daily resolution) with the column information already given earlier (3). Rows of the economic data contain the value of a specific economic indicator referred to in the columns for a specific date on a monthly resolution. For the combination of the two data frames into one, which makes modelling later feasible, it will be necessary to align the timely resolution of the two data frames. First, it needs to be decided (a) if Bund data should be transformed to match economic data resolution or (b) if economic data should be aligned with Bund data resolution (to be precise, there would be a third option of adapting both resolutions to a different one, but this is not discussed here, as it seems without cause at this moment). This first question is comparably easy to answer, as the main focus is still placed on Bund prices. This means, that losing information about the Bund data is not desired (and transforming higher, daily resolution to a monthly resolution will certainly lose information).

Consequently, this results in the question of how to adapt economic indicator resolution (monthly) to Bund price information (daily). This could be achieved in various ways, of which three are discussed here. First, one option (a) is to set the value of the economic indicator to a constant for each day in the month. This constant would be naturally set to the value, which is available for that month in the economic data set. That approach has the advantage that it is rather easy to implement. Also, it does not make strong assumptions about the economic data, although the scenario is not entirely realistic. This is because it is unknown for each economic indicator, at which exact day of the month the data is published. In case it is not published on the 1st of every month, this way of adapting the resolution of the data then might include knowledge about future development of the economic data. The second option (b) is the idea of linear interpolation. The values per day of each month then are modelled with a linear function through the original economic data for each month. One drawback is, that the days, which are starting and ending points

for the interpolation, are in this scenario rather arbitrarily chosen, as it is unknown, when this data is published exactly. It will be assumed, that it is the first day of each month, which again might at some points include future knowledge. However, for the purpose of modelling the Bund price correctly (and showing as a first step its feasibility) this seemed more reasonable than omitting already available information, which would be the other option. An important benefit of this method is, that those values, when interpolated, change on a daily basis. As forecasting will be done with a daily time horizon as well, as it will be explained in the next subsection (5.2.1), this characteristic of the linear interpolation seems almost essential. This is also, why it has been chosen for the fusion of the two data sources in this work. For the sake of completeness, a third option (c) is to model the days in between the obtained economic data points not linearly but with more complex approximations. This has similar consequences as (b) with the additional disadvantage that this modelling might add more complexity to the data fusion than required. However, this will be kept in mind and addressed later in this work again.

5.2 Different Forecasting Setups

5.2.1 Forecasting with Different Time Horizons

Until now, it has not been discussed, what the actual scope of the forecasts will be. This will be covered in this subsection.

Next Day Considering the daily Bund data, the most intuitive forecast would probably be a next day forecast. This is also, what can be seen in the available related work (2.3.2). Also, less complex models are expected to perform already quite well on this task, which makes a comparison between more sophisticated approaches easier. More details on model selection and their configuration is found in 5.3.

Next Week In contrast to next day forecasts, forecasts about the more distant future are not found frequently in related work. However, forecasting the more distant future is not only an interesting task in theory, but also valuable in practice. Additionally, looking further into the future than the next day might be a task, in which less complex models are outperformed by neural network based approaches. This however, needs to be verified with the upcoming experimental results.

5.2.2 Forecasting with Different Target

Another distinction, which will be included in the experiments as well, is the distinction between price and return (return = difference between price at day t and price at day $t - 1$) as the target for the forecasting.

Price This distinction is primarily made due to the non-stationarity of the Bund price development. Stationarity has been discussed earlier already (2.3.2) and finds application in modelling time series.

Return In contrast to the Bund price development, the returns of the Bunds are stationary and therefore it is assumed, that this stationarity allows for better forecasting models.

Both, price and return forecasts, will be tested and reported in the next sections.

5.2.3 Forecasting on Original and Fused Data

5.3 Models for Forecasts

A major component of this work is to compare neural network based and deep learning inspired solutions to bond price forecasting to other state of the art methods for this task. In order to enable a fair comparison between the LSTM based approach developed in this work, the common technique of ARIMA time series modeling has been implemented. In addition, the neural network approach proposed by Ganguli et al. ([23]) particularly designed for the challenge of bond price forecasting, as stated in the work by Ganguli et al. ([23]), has been implemented based on the bond price data used in this work. In addition, a LSTM based approach with its focus on sequence modelling will be developed. All in all, this leads to an extensive comparison of less complex models, such as naive, median, mean and linear regression and more sophisticated ones, like ARIMA, MLP and LSTM models. For all of them, the selected configurations will be discussed next. Note that all the definitions in the remaining lines of this chapter cover the next day forecast. With small adaptations, this can be easily extended to next week forecasts.

5.3.1 Naive

The first forecasting technique is also the most simple one, but surprisingly not uninteresting as later experiments will show. The method which is often referred to as *Naive* uses the value of the current moment (y_t) to forecast the next moment

(y_{t+1}) by assuming that the next moment will be equivalent to the current one. In the context of this work, t refers to days in the available data.

$$\hat{y}_{t+1} = y_t \quad (15)$$

Especially for stationary time series with low volatility, naive forecasts are expected to approximate the actual values well.

5.3.2 Median and Mean

In addition to the naive approach, the most common approaches are probably median and mean forecasts. Both methods use a number of points of the past to calculate mean, or respectively median, from this number of points and assume that this calculation is close to the value of the next moment. As in naive forecasting, t here denotes a day and y_t represents the value at point t . \hat{y} denotes a forecast, not a (necessarily) true value.

$$\hat{y}_{t+1} = \begin{cases} y_{\frac{t+1}{2}} & t \text{ odd} \\ \frac{1}{2}(y_{\frac{t}{2}} + y_{\frac{t}{2}+1}) & t \text{ even} \end{cases} \quad (16)$$

Formula for mean:

$$\hat{y}_{t+1} = \frac{y_t + y_{t-1} + \dots + y_{t-n}}{n+1} \quad (17)$$

Both, mean and median, can have similar results. However, there are also differences. For example, the median is more robust to outliers due to its property, that it ignores any extreme values in a set. The mean can be influenced by those, which is not necessarily a disadvantage and can possibly include useful information in the forecast.

What needs to be decided for median and mean forecast however, is the number of days, which will be considered for calculating median, respectively mean. Looking back one day into the past would be equal to the naive approach which is why this is omitted. In this work, three, five and ten days will be tested and results reported in section 6.

5.3.3 Linear Regression

Linear models are also commonly used due to their rather good interpretability and also their limited complexity. They also use a number of points of the past to forecast the upcoming moment in time (\hat{y}_{t+1}). In contrast to median and mean,

every point in the past (y_{t-i}) is weighted (a_i) and their sum is added to an intercept (b).

$$\hat{y}_{t+1} = b + \sum_{i=0}^n a_i y_{t-i} \quad (18)$$

The adjustment of those weights is calculated on the already available data. The specific implementation in this work is based on the scikit-learn Python library ([53]). Also, for linear regression, a number of past days, which will be considered, needs to be chosen. As for median and mean, three, five and ten days will be tested and results reported in the next section.

5.3.4 ARIMA

ARIMA processes have been already introduced in section 2.2.1 about time series analysis. They have been proven useful in time series forecasts and will be further applied in this work. As explained earlier, an ARIMA model consists of an autoregressive, an integrated and a moving average part, for each of which a parameter must be set. How to best choose these three parameters has been investigated by George Box and Gwilym Jenkins, which is why the gold standard method for selecting those parameters is named Box-Jenkins method ([4]). The Box-Jenkins method uses autocorrelation and partial autocorrelation to identify the parameters for the ARIMA model. Parameter choices for this work will be discussed next.

Identifying the autoregressive parameter First, the parameter for the autoregressive part, denoted as p , is identified. This is achieved by calculating the autocorrelation of the to be modelled time series, in this case all the available Bund data. Autocorrelation generally is the correlation between a sequence of values and a delayed or shifted version of the same sequence of values. The number of shifts is referred to as lags and in the following autocorrelation plot, autocorrelation (on the y-axis) is plotted against the number of lags (on the x-axis). Following the Box-Jenkins approach, to identify a suitable parameter p , the number of lags, where a sufficiently strong autocorrelation exists, should be chosen. Here, the precise selection criterion differs in literature ([6]). In this scenario, the autocorrelation seems rather high for up to 200 lags which indicates that a p of up to around 200 would be reasonable, in theory. However, in practice calculating ARIMA models for a p this large is not feasible due to enormous computation times. Running experiments with p of 100 would have taken months, and even with p of 50 multiple weeks. That is why it has been chosen to limit p to 10 in the conducted experiments, which already

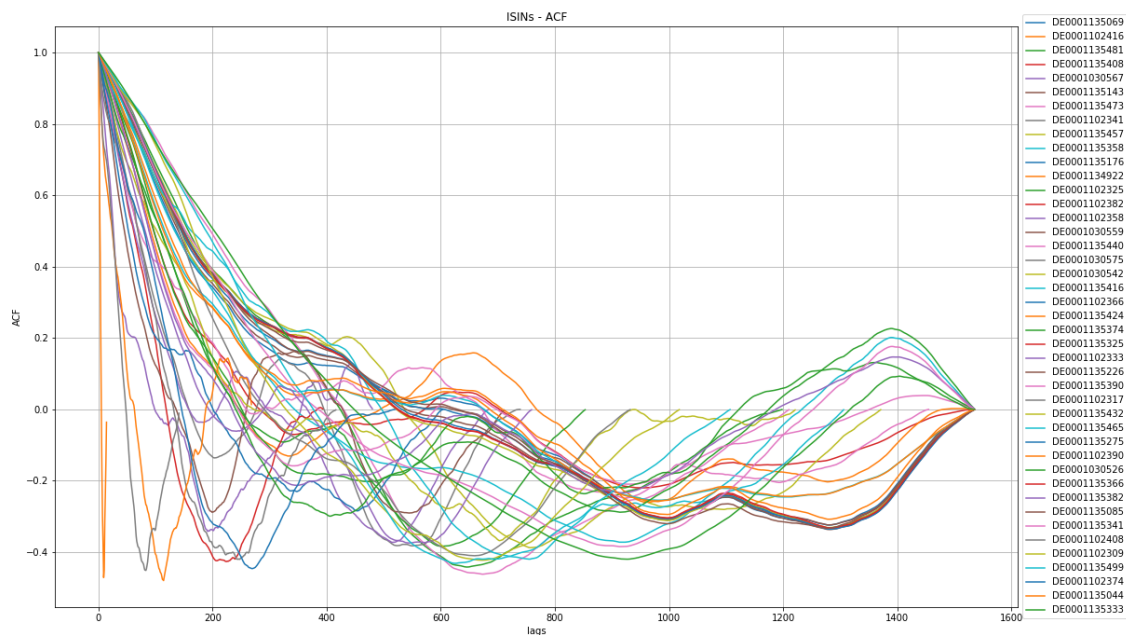


Figure 27: Autocorrelation for ARIMA Parameter Selection by Box-Jenkins.

takes more than a week to compute for the total number of Bunds. With 10 as the upper bound for p , other p tested were 1,3 and 5. The reader might have noticed that this selection of p was conducted based on the autocorrelation of the prices. If returns are also considered in model building, this would require to inspect the autocorrelation of returns as well. However, as the computational limitations have been made clear, limits for p in case of forecasting returns are set. This is why p for return forecasts is chosen as for price forecasts.

Identifying the integration parameter For the parameter of the *integrated* part of ARIMA, i , which helps to obtain stationarity, it will be necessary to distinguish between prices and returns. As seen earlier, the returns are stationary. This means the i in return forecasting can be set to 0. For the price forecasting on the contrary, it has been argued that prices are non-stationary. Here, setting $i = 1$ induces stationarity, which is necessary for the ARIMA model to be computed.

Identifying the moving-average parameter The last part of the ARIMA model, the moving-average parameter is selected based on the partial autocorrelation plot ([4]). Partial autocorrelation in contrast to autocorrelation itself does not only calculate correlation between value and lag, but also removes the influence from all lags smaller than the one calculated. This helps to identify correlation between value and lag without the effects of any smaller lags. Identifying the

moving-average parameter, or q , then is done similarly by inspecting the partial autocorrelation graph. Here, one can observe that the partial autocorrelation does

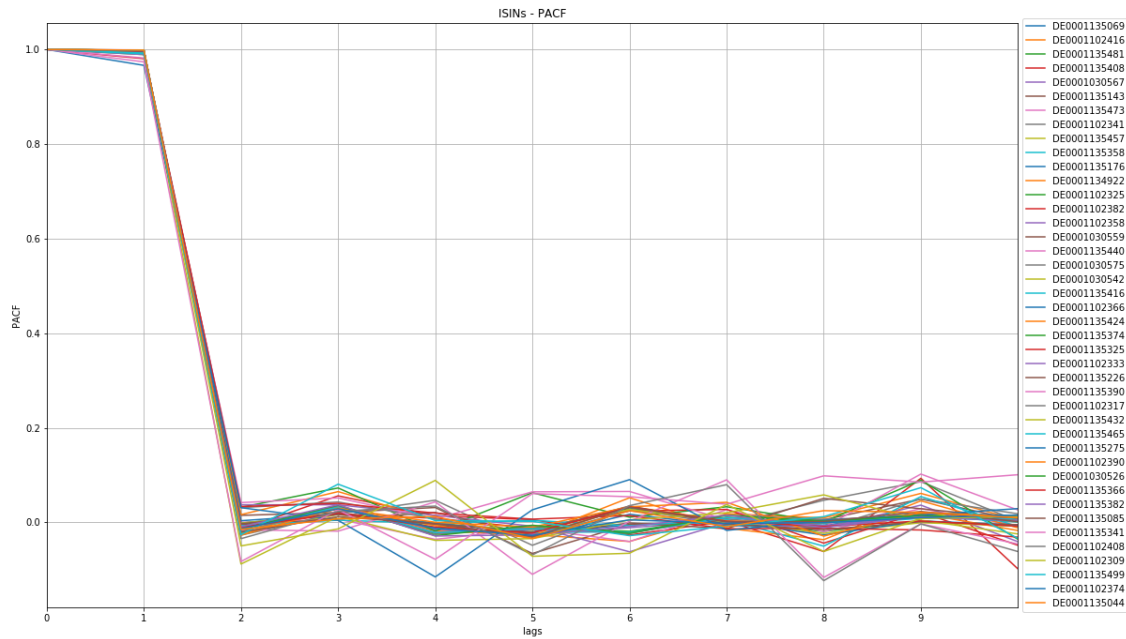


Figure 28: Partial autocorrelation of training data, Bund prices for ARIMA parameter selection.

decay rapidly to and circles around 0 after lag = 1. This is why q of 1 or 0 will be tested for Bund price forecasts.

This parameter, q , however, does not necessarily need to be the same for return and price. With a q of around 1, computational boundaries are not met yet. Hence for this parameter, an inspection of the partial autocorrelation plot of returns seems necessary as well. In this plot, it can be seen, that the partial autocorrelation already decays to 0 at lag = 1. This means that testing any higher number than 1 as q for return forecasting is not promising and does not comply with the here conducted Box-Jenkins approach.

All in all, this results in the following ARIMA (p, d, q) configurations for price and return forecasts:

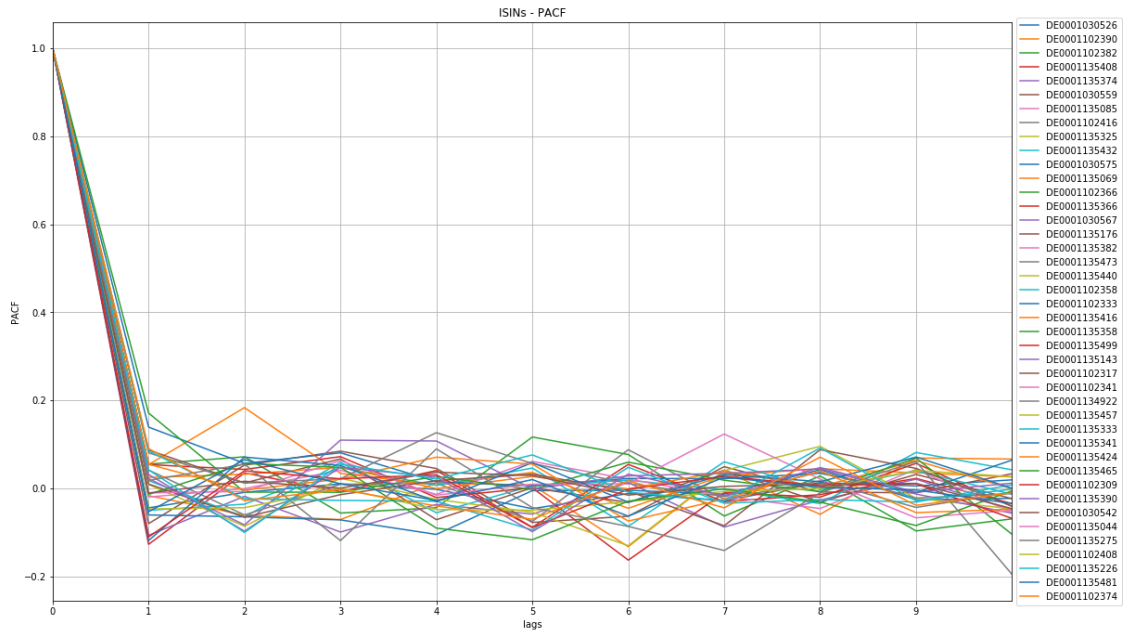


Figure 29: Partial autocorrelation of training data, Bund returns.

price ARIMA configuration (p, d, q)	return ARIMA configuration (p, d, q)
$(1,1,0)$	$(1,0,0)$
$(1,1,1)$	$(1,0,1)$
$(3,1,0)$	$(3,0,0)$
$(3,1,1)$	$(3,0,1)$
$(5,1,0)$	$(5,0,0)$
$(5,1,1)$	$(5,0,1)$
$(10,1,0)$	$(10,0,0)$
$(10,1,1)$	$(10,0,1)$

Table 8: ARIMA configurations for prices and return forecasts.

ARIMA model implementation in this work is based on the StatsModels package, statistics for Python, and their implementation of ARIMA modelling ([56]).

5.3.5 Multi-Layer-Perceptron Regressor

The last two model classes, of which the configurations will be discussed, are neural network based approaches, starting with the Multi-Layer-Perceptron (MLP) Regressor. An introduction into feedforward networks to which MLPs belong has been

provided earlier (2.1.3). Now, different MLP configurations and architectures are addressed. Fortunately, there are starting points provided in related work 2.3.2, especially in Ganguli et al. ([23]). Although the work presented there, is built on a different data set and also on different government bonds, it is assumed that the configurational choices can be transferred to the tasks discussed here. In addition of how to chose the number of units and layers in the network, there are several other questions which need to be answered of which just a few are named in the following:

- How to structure the input?

This questions aims at the way, the data is transformed, before fed into the network. One transformation, which is often used, and connected with the choice of activation function, is scaling. If for example hyperbolic tangent is chosen as the activation function (2.1.3), any values passed as input to an activation function larger than one only have a marginal effect. In the case of Bund price prediction this could be a problem as Bund prices start with a minimum at around 80 Euro (Bundesbank's standard price, 3). Still, it has been decided that the input is not scaled for MLPs which is connected to answers to the following questions.

- Which activation function to choose?

Of the various activation functions available, ReLU is the one used for this task, mainly due to its ability to handle values as large as the Bund prices. Also, for Bund returns this should be suitable as they are scaled to the range (0,2).

- Which optimizer to choose?

There are also various optimizers to choose from, which adapt the weights while training. For this task, the Adam is used, a stochastic gradient based optimizer. Other optimizers might work as well.

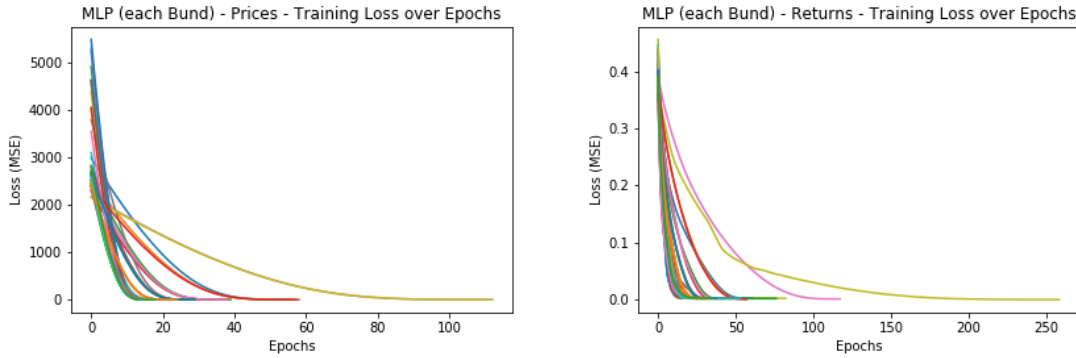
- Which learning rate might work best?

The choice of learning rate in combination with the optimizers (not all optimizers require a defined learning rate) is also important as unfortunate choices of learning rates might oversee a minimum or converging to a local minimum, while another minimum exists. For this task, the learning rate is set to 0.001.

- How many epochs of training?

Figure 31, depicting the loss per epoch when forecasting on price or return and trained on either each Bund or all Bunds, is used to derive an idea of how

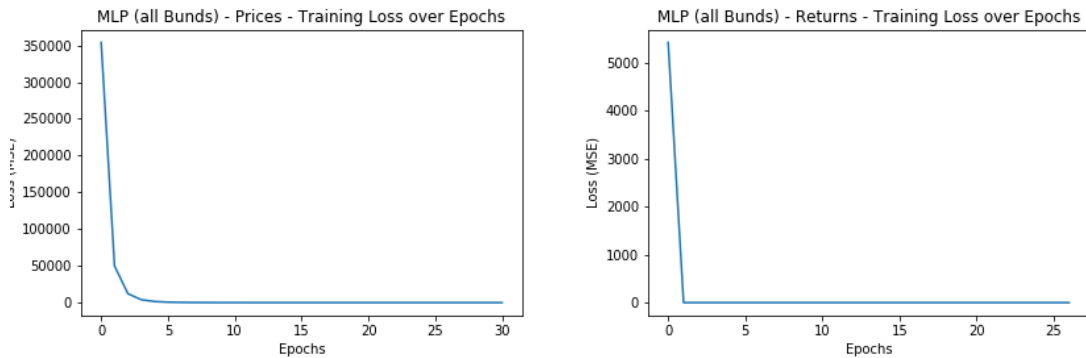
long training should be conducted (in epochs). All of the losses depicted in the figures below decay within the provided scope on the x-axis. Most of the loss curves are declining towards zero already after a few epochs. Few models need slightly longer and only outliers are not close to zero after the here reported scope of 50 epochs. This results in the selection of 50 epochs of training for the LSTM models. The number of training epochs will be discussed later again.



(a) Losses per epoch, price, each Bund, architecture: (20,10).

(b) Losses per epoch, return, each Bund, architecture: (25,10,10).

Figure 30: Representative example for one selected MLP architecture: losses per epoch, training on each Bund, different colors indicate different models.



(a) Losses per epoch, price, all Bunds, architecture: (10,10,10).

(b) Losses per epoch, return, all Bunds, architecture: (30,20,10).

Figure 31: Representative example for one selected MLP architecture: losses per epoch, training on all Bunds.

In addition to these question, there are more parameters to be set. For these configurations, however, default settings of the scikit-learn package ([53]), on which implementation of MLP models in this work is based, and its MLP regressor are utilized.

As briefly mentioned earlier, there is another important choice about the architecture of the network: How many units and how many layers should be used? Ganguli et al. report only results for one-hidden-layer-networks with 5, 10, 20 and 30 units in the hidden layer. Building on their experimental insight, MLPs in this work will also be tested with 10, 20 and 30 units per layer. However, 5 units will not be tested as MLPs with 5 units in one hidden layer were outperformed by MLPs with the other numbers of units per layer. In addition, 25 units per layer will be tested to illuminate performance between the two best performing MLPs with 20 and 30 units per hidden layer.

Furthermore, in contrast to research of Ganguli et al., this work will explore the effect of adding one and two additional hidden layers to the MLPs resulting in the experimental setup, that MLPs with one, two and three hidden layers will be tested. These extra hidden layers will contain 10, 20, 25 and 30 units just as the first hidden layer. This results in 84 different MLP architectures, which will be evaluated in this work. An abstraction of the 84 different architectures can be found below in figure 32.

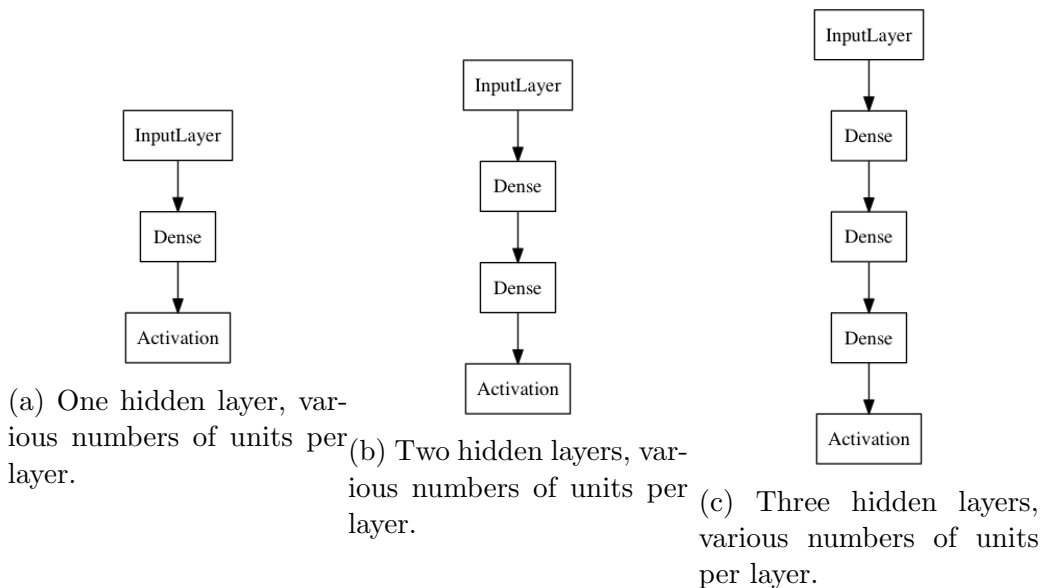


Figure 32: Overview of MLP architectures, one to three hidden layers.

5.3.6 LSTM

LSTMs (2.2.3), to the best of the author's knowledge, have not been applied to bond price forecasting. Consequently, there is no starting point, when addressing architectural and configurational questions. Although a comparison between MLP

and LSTM is not entirely fair, architectural choices for LSTMs are taken similarly to the MLP setup. This means, that between one and three LSTM layers of 10, 20, 25 and 30 LSTM cells are tested in this work, resulting in 84 different LSTM architectures equivalent to the number of MLP architectures. In addition to the LSTM layers, dropout is added in between them to avoid overfitting and allow for a more generalized model. The concept of dropout has not been introduced yet. Dropout is a common regularization technique in neural networks, meaning it is designed to avoid overfitting on the training data, which then could lead to better (because more general) models. These models possibly perform worse on the training data. However, they often perform better on test data. The technical idea behind dropout is simple: With a fixed chance (in this work 0.1 will be used) connections between layers are randomly *dropped* in each training epoch and information contained in the connection from the output of one layer is not transported to the next layer. This results in every LSTM layer being followed by a dropout layer. Similarly to the MLP architecture, the last two layers then are a Dense layer (fully connected layer) and an activation layer for outputting the forecast. An architectural abstraction can be found in figure 33.

In addition to the architectural design choices, the same questions, which were posed in combination with the MLP design, need to be answered:

- How to structure the input?

As mentioned in the MLP section, the question of how to structure input is connected to other questions, especially the question about the activation function. Before, scaling was not used due to the characteristics of the data and the capabilities of the ReLU function used in the MLP examples. In LSTMs however, hyperbolic tangent is used as the activation function (2.1.3), not ReLU. In contrast to ReLU, the hyperbolic tangent is almost constant for inputs larger than one. For the purpose of return forecasting this might not be a problem, but for Bund prices it is assumed to be problematic. This is why, for the LSTM experiments, all input will be scaled to the range (0,1) with min-max-scaling. Scaling any original value x to x' , within the range (0,1), works as follows:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (19)$$

This min-max-scaling is conducted for every column of the full data frame, including both training and test data. Before doing this, it was ensured that minimum and maximum values of return and price are located in the training data to avoid look-ahead bias ([37]).

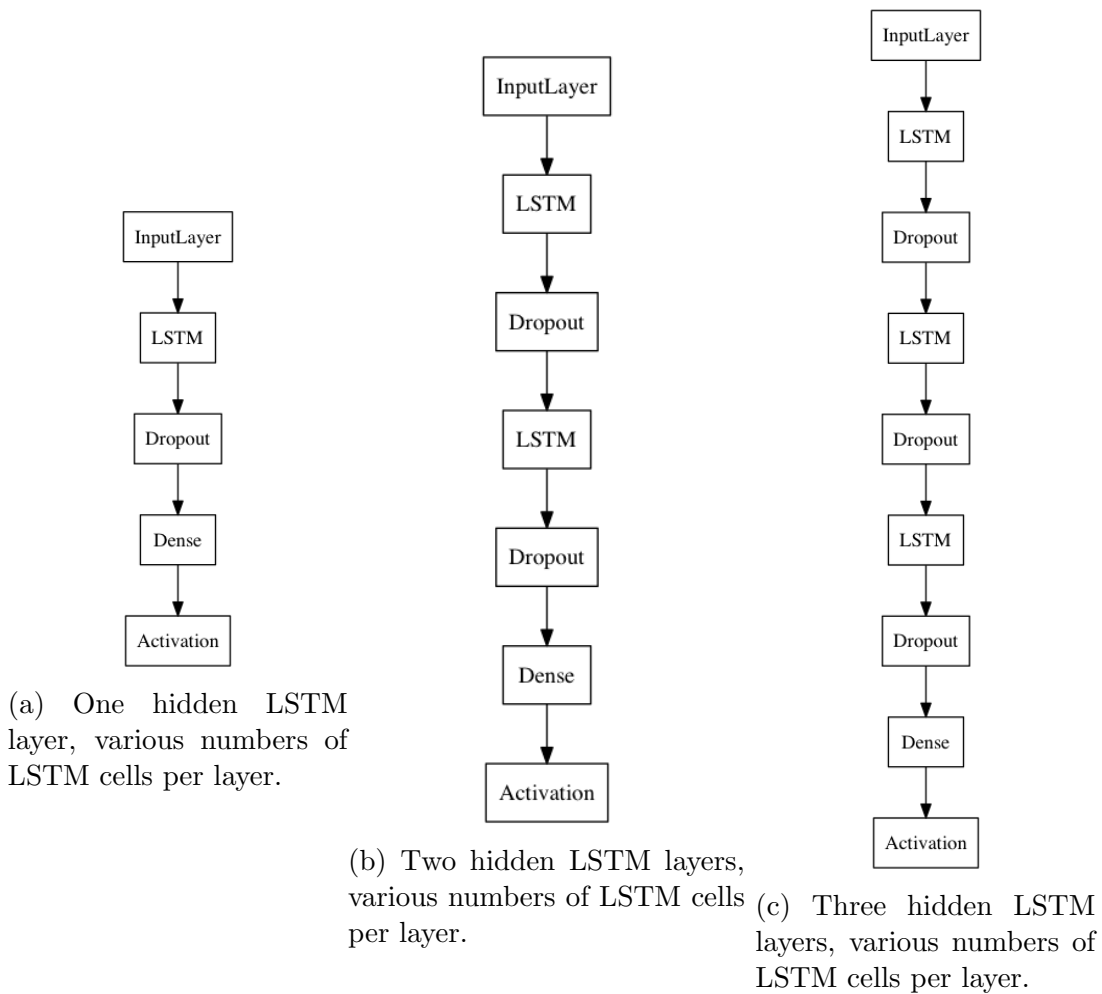


Figure 33: Overview of LSTM architectures, one to three hidden LSTM layers.

- Which activation function to choose?

The choice for the hyperbolic tangent function for this experimental setup has already been mentioned. The arguments supporting this choice are of practical nature: First tests have been done with no scaling similarly to MLP but they did not converge due to problems with exploding gradients. This made adaptations in scale of the input necessary. With the LSTM layers, the hyperbolic tangent is the default choice of the Keras deep learning library ([12]), which builds on Tensorflow ([49]), and is used for the implementation of LSTM models in this work. The default activation function of hyperbolic tangent has not been changed. The last layer, the Dense layer, uses a linear activation function to generate the output.

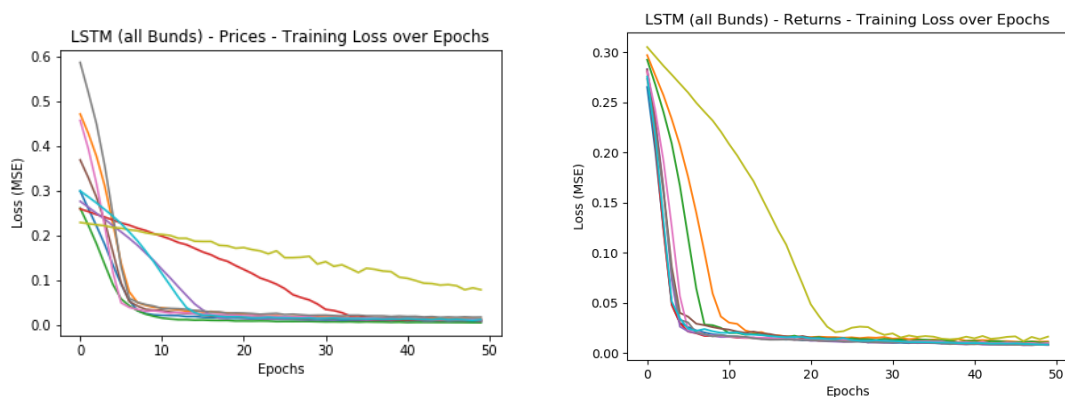
- Which optimizer and which learning rate to choose?

The optimizer is chosen as in the MLP setup, meaning Adam is used as opti-

mizer. Different initial tests for the learning rate showed that a learning rate of 0.001 seems reasonable. Larger learning rates resulted in a non-converging behavior of the networks.

- How many epochs of training?

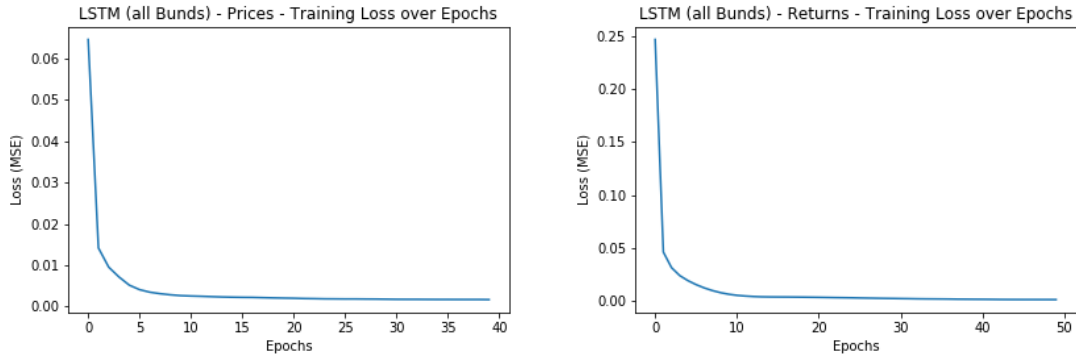
Figure 35, depicting the loss per epoch when forecasting on price or return and trained on either each Bund or all Bunds, is used to derive an idea of how long training should be conducted (in epochs). All of the losses depicted in the figures below decay within the provided scope on the x-axis. Most of the loss curves are declining towards zero already after a few epochs. Few need a little longer and only outliers are not close to zero after the here reported scope of 50 epochs. This results in the selection of 50 epochs of training for the LSTM models.



(a) Losses per epoch, price, each Bund, architecture: (20,10).

(b) Losses per epoch, return, each Bund, architecture: (25,10,10).

Figure 34: Representative example for one selected LSTM architecture: losses per epoch, training on each Bund, different colors indicate different models.



(a) Losses per epoch, price, all Bunds, architecture: (10,10,10).

(b) Losses per epoch, return, on all Bunds, architecture: (30,20,10).

Figure 35: Representative example for one selected LSTM architecture: losses per epoch, training on all Bunds.

Another property of LSTM models has been not mentioned yet, which is the ability to include multiple time steps in the training. Thus, in contrast to MLP networks, the LSTM approach presented in this work makes use of this LSTM characteristic and does not train models solely on the current features, but also on past features, while maintaining their structure. This structural relation is one of the key differences between LSTM and MLP models, as they could, theoretically, also utilize features of earlier points in time, but they do not consider their relation. Then, the question remains of how many points of the past are considered. Here, various different options are possible. Structuring the data in a way, that the model can train well on it seems reasonable. This can be achieved by taking the whole sequence of a Bund of training information with n entries and splitting it into $n-k+1$ k -tuples, where $k \in \mathbb{N}$ can be chosen arbitrarily and the amount of tuples will then be determined by the the size of the training information and the size of the tuples (n). A visualization of this transformation of time series data to LSTM input is depicted in figure 36. These k -tuples are constructed in the way, that if the training time series data is ordered, and each point in the time series is denoted by x_t and its corresponding target (price or return) by y_t , for every $t > k$, there is a tuple constructed with $x_t, x_{t-1}, \dots, x_{t-k}$. Each tuple is then assigned the value of the next point in time, y_{t+1} . This combination of tuples $(x_t, x_{t-1}, \dots, x_{t-k})$ and target (y_{t+1}) is used for LSTM model training. In this work, models with $k \in [3, 10]$ will be implemented.

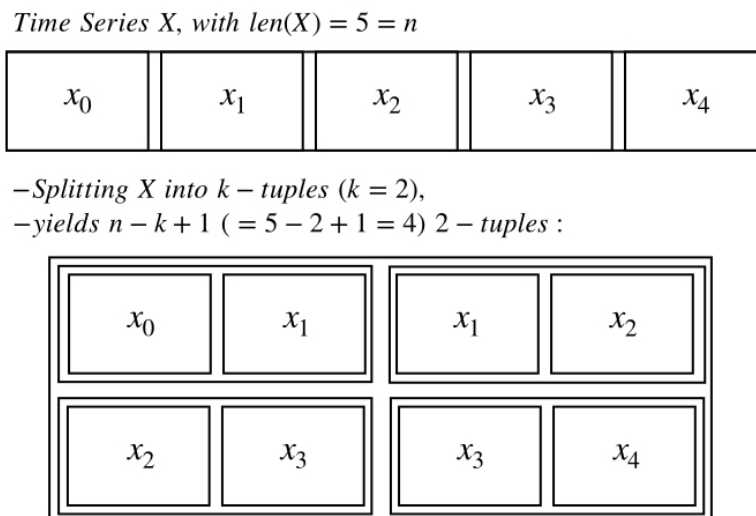


Figure 36: Visualization of time series transformation for LSTM input.

5.4 Features for different models

Implicitly touched until now has been the property that MLP and LSTM models, in contrast to the other ones described, can use other features except the target value itself for forecasting. All other models forecast future price/return development solely with past price/return information. This also means, that only MLP and LSTM can make use of the fused data set which contains additional economic features. For MLP and LSTM models therefore, both options will be implemented and results reported. So, for each MLP and LSTM model configuration, there will be one model with and one without incorporating the additional data in the fused data set.

5.5 Generalizability of different models

One important aspect, which has not been sufficiently touched until now, is, what has been called *generalizability of different models*. This aims at the ability of the neural network based approaches, MLP and LSTM, to be trained for a broader, more general purpose of forecasting on every Bund, even on those which were not included in any training data.

In contrast to the MLP and LSTM approach, naive, mean, median, linear and ARIMA models do not have this capability to generalize or at least not in the context, in which they are used in this work. Naive, mean, median, linear and ARIMA models need to be trained for each and every Bund, resulting in an equivalent num-

ber of models and Bunds in the training data. The neural network based approaches can be used similarly, meaning that one model is trained for each Bund. However, due to their ability to generalize and abstract from a specific Bund, it is also reasonable to train one model for all Bunds. This more general model is then also expected to be able to make reasonable forecasts on yet unseen (meaning unavailable in training data) Bunds. Both approaches, training a model on *all* Bunds and training models on *each* Bund will be tested in this work's experiments.

Furthermore, there is an additional important feature in model training when it comes to generalizability. This is the possibility to already include a validation set in the model training phase. This option is available for both MLP and LSTM networks and will be conducted for both with a validation fraction of 10%, meaning that 10% of the training data will be used to validate the model in each step of the training phase (when updating weights in the networks). This is commonly used to avoid overfitting and achieve further generalizability.

In summary, the naive, mean, median, linear and ARIMA models do not generalize as well as the two neural network techniques mentioned. This is why approaches based on the naive, mean, median, linear and ARIMA methods are trained for *each* Bund whereas MLP and LSTM based approaches are trained in two ways, on *each* Bund and on *all* Bunds.

5.6 Rolling Forecast vs Classical Forecast

Another essential difference lies in the way the forecasting is actually conducted, but, first, how forecasting is actually done has not been explained yet, this needs to be addressed.

By now, it is clear how training and test data are set up. The model performance will be evaluated on the test data (more information on this matter in section 5.7, but how are forecasts for the test data calculated? In this work, there are two different ways of forecasting, which are connected to the model choice. There is, what will be called *rolling forecast* in this work, for naive, mean, median, linear and ARIMA models, and there is a here called *classical forecast*, which is different from a rolling forecast model. How these two forecasting approaches differ, is tackled next.

In essence, the idea of the *classical forecast* refers to the idea of training a model on the training data and then being able to apply the obtained model on every element in the test data set. However, this is not appropriate for naive, mean, median, linear and ARIMA models. A simple example works best to illustrate this: The naive model, for example, will almost certainly produce large errors in forecast

when the naive model of the some random day in the training data set is applied to one of the points in the test set. Assume that the Bund price on day t is 100. The naive model trained on day t would then suggest that the Bund price on day $t + 1$ is also 100. For subsequent days and Bund prices, which are normally not changing drastically from a day to its subsequent day, this will probably work with only a slight error. Nevertheless, using the same model trained at day t , will almost certainly produce a large error when forecasting the Bund price for day $t + 100$ or maybe even $t + 1000$. Due to the nature of the naive, mean, median, linear and ARIMA models, which consider solely a certain, limited number of days of the past, these models need to be re-trained after every forecast made in the test set. This process of re-training and forecasting is the idea behind the *rolling forecast*. A visualization of this process can be found in figure 37.

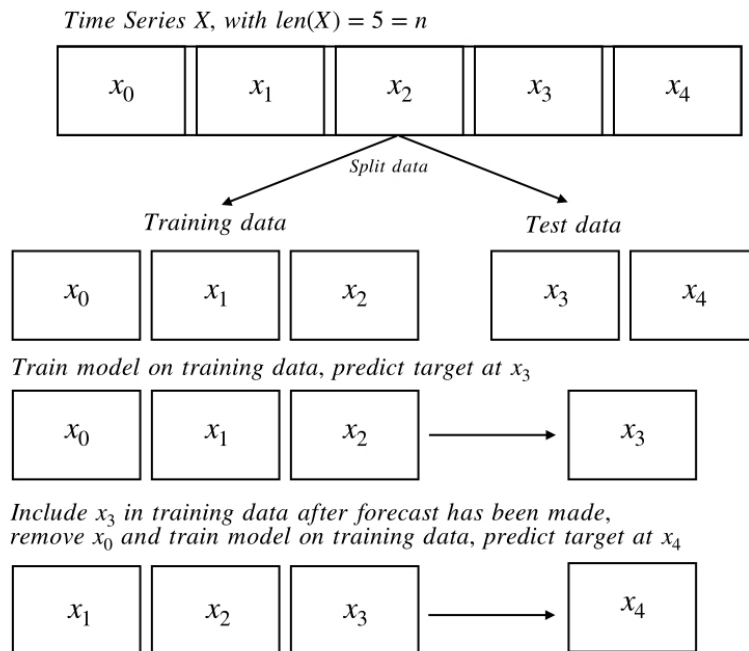


Figure 37: Visualization of rolling forecast method.

On the contrary, MLP and LSTM models are trained on the training data and then the Bund price/return of every element of the test set should be reasonably forecasted by the corresponding model. One might ask, if the rolling forecast could not be used for MLP and LSTM models as well to improve their forecasts as well. It is certainly correct, that a rolling forecast could be used with MLP and LSTM models. However, this will not be executed in this work's experiments due to two considerations. (a) When training on all the training data, which is almost 60,000 elements (3) when training on all Bunds or slightly above 1,500 elements when

training on each Bund, adding one element to the training scope will not change much. (b) Training MLP and LSTM are time-consuming in contrast to naive, mean and linear approaches. Using rolling forecast on this would certainly exceed the experimental means of this work.

5.7 Evaluating model performance

The final subsection, before moving on to the experimental results, will be dedicated to the question of how to actually evaluate model performance. In general, model performances are measured by the distance between prediction and actual value and many adaptations to it. The probably most frequently used one is the *mean squared error* metric (mse), defined as follows:

$$mse = \frac{1}{n} \sum_{t=1}^n (y_t - \hat{y}_t)^2 \quad (20)$$

The mean squared error will also play an important role in the evaluation of model performance in this work. In addition to the mse, another metric, the *mean absolute percentage error* (mape) will be reported. It is defined as below:

$$mape = \frac{1}{n} \sum_{t=1}^n \left| \frac{y_t - \hat{y}_t}{y_t} \right| \cdot 100 \quad (21)$$

The mape avoids two characteristics, which can be seen as a disadvantage of the mse. (a) The mse punishes far distant forecasts harder than absolute errors like mape. (b) The mse can not be interpreted without understanding of the underlying data. A mse of 0.5 can be satisfactory or the very opposite. A mape of 0.5 on the contrary, seems more likely to be satisfying (although this is admittedly context dependent). Hence, these two metrics, mse and mape, will be used for model performance evaluation. Another affair, which needs to be kept in mind, is the fact, that in this work not just one forecast for each model is evaluated. Every model needs to make forecasts for at least the 43 different Bunds (3) in the test scope. Consequently, every model will be associated with a minimum of 43 mean squared errors and mean absolute percentage errors. Making sense of all these results would not be very convenient which is why for each model, the mean and median of the 43 different error results will be reported. This yields four different performance measures for each model, the mean of mses, the median of mses, the mean of mapes and the median of mapes. In the results section (6), those will be denoted as **mse**

(mean), mse (median), mape (mean) and mape (median).

6 Experimental Results

The extensive experimental nature of this work yields a large number of results, which are fully reported in the appendix. Here, only the best performing models per combination of model class, target, without and with economic data will be reported.

Model class in this context refers to the selected models for the experiments of this work, meaning naive, linear, mean and median (which will be often referred to as *less complex models*, ARIMA models, MLP and LSTM models. So, exemplarily, for the model class MLP, results have been calculated for next day and next week forecasts, both for price and return and both with and without the fused economic data set. For each of those combinations the best performing model (where performance is measured as described earlier (5.7) is reported in this section and summarized in the following.

In addition, two visualizations are provided for each task (next day/week and price/return forecast). One depicting the two best performing models on the task on a randomly selected ISIN, a second one depicting the actual value of x vs the prediction for that x , \hat{x} .

The reason behind showing the first visualization is to obtain an understanding of how the development over time is captured by the two best performing models (the best performing in terms of mse (median) and the best performing in terms of mape (median)) of each task. The aim behind exhibiting the second visualization is to grasp, how each of the models generally perform on the given task. In contrast to the representation in a table, this provides insight in additional characteristics of each model, which are lost due to the absolute character of the errors in the tables, e.g. if a model almost always forecasts a value too high or too low.

First, results of next day price and return forecasting are listed, followed by the results of next week price and return forecasting.

6.1 Next Day Forecast: Comparison of Model Classes

The first subsection of this chapter is concerned with the task of next day forecasting and its results. In the following, results for next day price forecasts as well as next day return forecasts are reported. As already addressed, results for each of the task are accompanied by one table reporting the best performing models per class and two visualizations.

6.1.1 Next Day Price Forecast: Best Performing Model Configurations of Each Class

The task of next day price forecasting is solved by many models in the following table (9) in high quality. Best performance with respect to mape (median) was achieved by the naive model. The best performance regarding mse (median) is observed for the MLP model, trained on each Bund, without additional economic data.

Generally, next day price forecast results for different models are similar and there are only two outliers, the LSTM models, trained on each Bund, which yield unsatisfactory results. All the other models perform well on this task, both when measured by mape or mse. Interestingly, the MLP model, achieving the best mse (median) score, is one of the worst, when evaluated by mape (median). One reason for this might be the absence or larger numbers of outliers in the forecast. Other models might yield forecasts with a larger number of outliers, which is heavily punished by the mse score. That the naive model (and others) are better performing with respect to the mape must mean, that these models forecasts very well on parts of the time series, where no outliers are found.

Figure (38) shows the forecasts of the two best performing models, the naive one and the MLP (trained on each Bund) and the first 50 days in the actual test data set of a randomly selected Bund. In this example, both models seem to model the test data also graphically well, with maybe even better performance of the MLP model. Note that this is simply one example of many forecasts on many and larger time horizons as well. However, it seems well suited to provide the reader a visual impression of how well the models work, in addition to the table reported earlier.

Another interesting visualization of the results is the plot of x (on the x-axis) versus \hat{x} (on the y-axis). The points of the optimal model would be placed on the thin black line on the diagonal. This would mean, that every prediction (\hat{x}) is equal to the corresponding actual value x . This plot especially helps to identify, if certain models generally aim too high or too low, or if they forecast more or less constant values. In this specific scenario, it can be observed that most of the LSTM models tend to aim too high whereas MLP models are closer to the optimal black points. The best two models for this task, MLP (trained on each Bund) and the naive model, are highlighted in the plot with different symbols. It can be observed, that points belonging to these two models are rather close to the optimal points.

model	mape (median)	mape (mean)	mse (median)	mse (mean)
average, last 3 days (15)	0.18	0.23	0.07	4.63
linear, last 3 days (15)	1.27	1.4	0.06	11.2
median, last 3 days (15)	0.19	0.24	0.08	3.59
naive, last 1 day (15)	0.14	0.18	0.04	6.73
arima (1,1,0) (17)	1.26	1.46	0.13	9.30
<i>mlp, trained on all Bunds:</i>				
mlp (10) (19)	0.27	0.32	0.12	0.33
mlp (20,10), four ind. (21)	0.41	0.51	0.47	0.94
<i>mlp, trained on each Bund:</i>				
mlp (20,10) (27)	1.25	1.43	0.01	3.29
mlp (10,30), seven ind. (29)	1.22	1.40	0.06	3.36
<i>lstm, trained on all Bunds:</i>				
lstm (30), 3 days (35)	0.19	0.27	0.06	3.55
lstm (25,25,20), 3 days, four ind. (36)	0.19	0.25	0.05	0.323
<i>lstm, trained on each Bund:</i>				
lstm (30,10), 3 days (39)	0.95	1.12	2.18	14.02
lstm (10,20), 3 days, four ind. (40)	2.99	3.38	19.48	25.97

Table 9: Next day price forecast, best performing model configurations per class. Notations as in chapter Experimental Results (6).

6.1.2 Next Day Return Forecast: Best Performing Model Configurations of Each Class

The result for the task of next day return forecasting are reported in the following table (9). Surprisingly, the results are worse than for price forecasting, when considering the mape. Mean squared error scores are low, but this is certainly due to the data itself, which is limited to a much smaller range than the price forecast. Best performance for this task was achieved by neural network based approaches. With respect to mape (median) the LSTM model, which has been trained on all Bunds, performed best. The best performance regarding mse (median) is observed for the MLP model trained on all Bund. Both are trained on the original, technical data without additional economic data.

Next day return forecast results for different models are similar and there is only one slight exception, the naive models. All the other models perform similarly on this task, both when measured by mape or mse.

One reason for the comparable results for most of the models is that all of

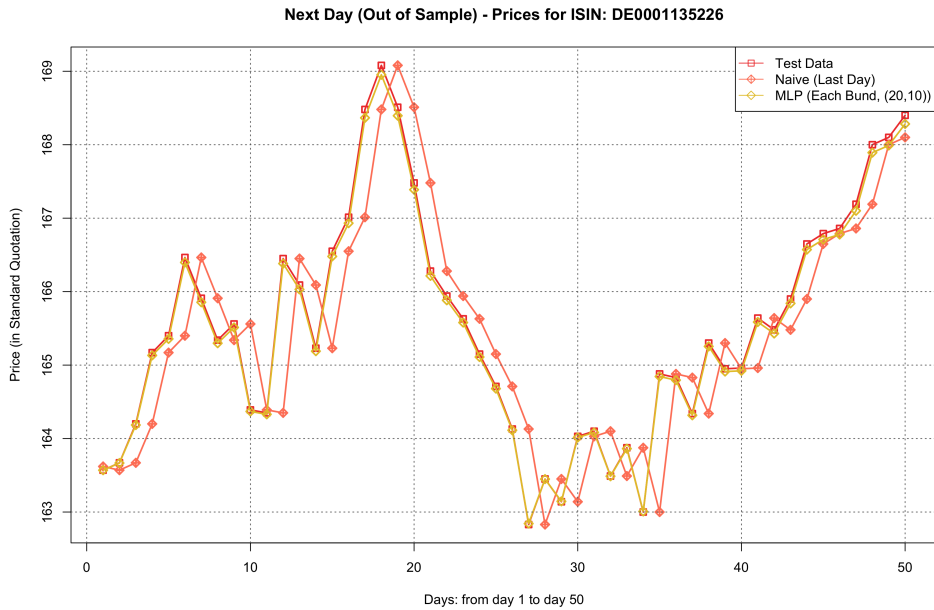


Figure 38: Next day price result visualization of the two best performing models on a randomly selected ISIN, limited to the first 50 days of the test data set.

them tend to forecast an almost constant value for the return, which (for neural network based approaches) seems to minimize their loss function. This behavior is also depicted in the following plot (40), showing the two best performing models for the task of next day return forecast, again with respect to mse (median) and mape (median).

The MLP model for this task and ISIN forecasts value around the constant of 1.11 or 1.12. Spikes in its plot seem to follow larger spikes in the actual data (around day 21 and 22 for example). Spikes for the LSTM model seem to take on values above the more or less constant line at around 1.11/1.12. However, many other spikes seem to be missed. The LSTM model is even more predictable in its behavior and forecasts values slightly below 1.10.

As for next day price forecasts, the following visualization of the results (39) is the plot of x (on the x-axis) versus \hat{x} (on the y-axis). The points of the optimal model would be placed on the thin black line on the diagonal. This would mean that every prediction (\hat{x}) is equal to the corresponding actual value x .

In this specific scenario, it can be observed, that almost all models tends to circle a constant value. This complies with the visualization introduced before (40). The best two models for this task, the LSTM and the MLP (both trained on all Bunds), are highlighted in the plot with different symbols. The impression, which arises from the plot, is, that these two models do not clearly outperform other models plotted.

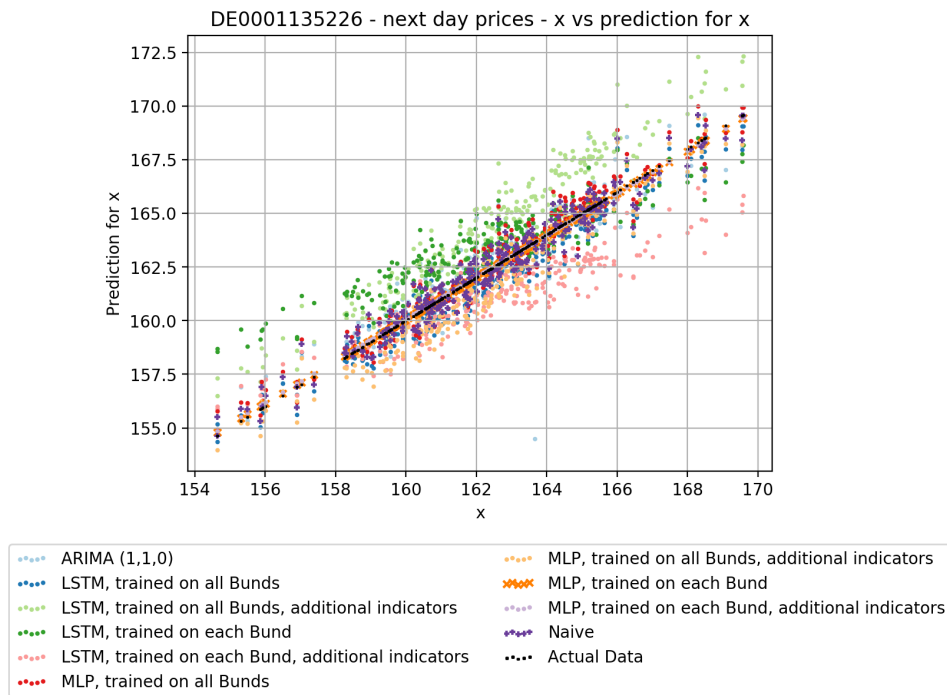


Figure 39: Next day price result visualization, plot of x vs \hat{x} .

This corresponds to the information in table 10.

The only plot, which would differ visually, is the plot for the naive model, but this approach produces largest error, which is why it is not depicted here, as only best performing models per model class are shown.

6.2 Next Week Forecast: Comparison of Model Classes

The second subsection of this chapter copes with the task of next week forecasting and its results. In the following, results for next week price forecasts as well as next week return forecasts are reported. As already addressed, results for each of the task are accompanied by one table (11) reporting the best performing models per class and two visualizations (42,43).

6.2.1 Next Week Price Forecast: Best Performing Model Configurations of Each Class

The task of next week price forecasting is solved by many models in the table 11 in high quality. Best performance with respect to mape (median) was achieved by the LSTM model, trained on the fused data set and on all Bunds. The best performance regarding mse (median) is observed for the MLP model trained on

model	mape (median)	mape (mean)	mse (median)	mse (mean)
average, last 10 days (16)	3.53	3.72	0.0042	0.0045
linear, last 10 days (16)	4.51	4.74	0.0057	0.0060
median, last 10 days (16)	3.50	3.66	0.0041	0.0043
naive, last 1 day (16)	4.66	4.86	0.0075	0.0082
arima (1,0,0) (18)	3.29	3.48	0.0038	0.0041
<i>mlp, trained on all Bunds:</i>				
mlp (20,10,30) (23)	3.27	3.27	0.0035	0.0035
mlp (20,10), four ind. (25)	3.52	3.93	0.0041	0.0046
<i>mlp, trained on each Bund:</i>				
mlp (25,10,10) (31)	3.53	3.75	0.004	0.0044
mlp (20,30,30), four ind. (33)	3.65	3.88	0.0041	0.0045
<i>lstm, trained on all Bunds:</i>				
lstm (30,30,10), 3 days (37)	3.22	3.44	0.004	0.0042
lstm (25,25), 10 days, four ind. (38)	3.23	3.26	0.004	0.0045
<i>lstm, trained on each Bunds:</i>				
lstm (10,10,25), 3 days (41)	3.60	3.93	0.004	0.004
lstm (10,20,30), 3 days, four ind. (42)	3.69	3.99	0.005	0.005

Table 10: Next day return forecast, best performing model configurations per class. Notations as in chapter Experimental Results (6).

each Bund, without additional economic data.

Next week price forecast results mostly are achieving good scores. Except the results of the naive model and the LSTM models, trained on each Bund, which yield unsatisfactory results. All the other models perform well on this task, both when measured by mape or mse. In contrast to the next day price forecast, the best performing models with respect to the mape (median), the LSTM model, also achieves top performance when measured by mse (median). In this metric, it shares the top performance with the mlp model.

In general, next week price forecasts seem to be similar to next day price forecasts in their quality. As expected, measured performances drop, but not deeply. The naive model does not work as well anymore, which can be expected, too, since the difference from one day to the next intuitively is smaller than the difference from one day to the next week.

Also similar to the next day price forecast, is plot 42 of the two best performing models, the MLP and the LSTM model. Both seem to grasp the development of

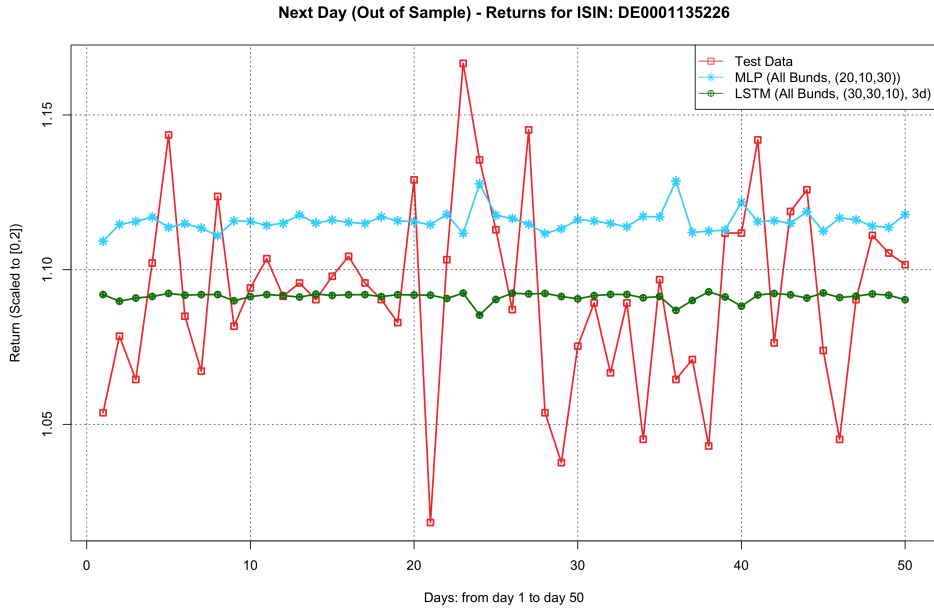


Figure 40: Next day return result visualization of the two best performing models on a randomly selected ISIN, limited to the first 50 days of the test data set.

the Bund price with a shift of around five days. This indicates, that the price at the current point of time still is the best indicator for the price in five days, with small variations. These variations seem to be large enough to not let the naive model work as well as other models.

Again similar to the next day price forecast, are the results depicted in plot 43 of x vs \hat{x} . Here, MLP models seems to forecast values below the optimal diagonal and LSTM models seem to forecast values slightly higher. This complies partially with the plot above (42), at least after day 30, where MLP forecast tend to be too low, whereas the LSTM forecasts are close to the actual data. Collectively, model forecasts are similar, which is why points of certain models are almost completely covered by points of other models, e.g. the LSTM models, which are drawn in shades of green.

6.2.2 Next Week Return Forecast: Best Performing Model Configurations of Each Class

The result for the task of next week return are reported in table 12. The results are similar to the ones for next day return forecasting, when considering mse and mape. Again, mean squared error scores are low for already introduced reasons. Best performance for this task were achieved by the LSTM model, which has been

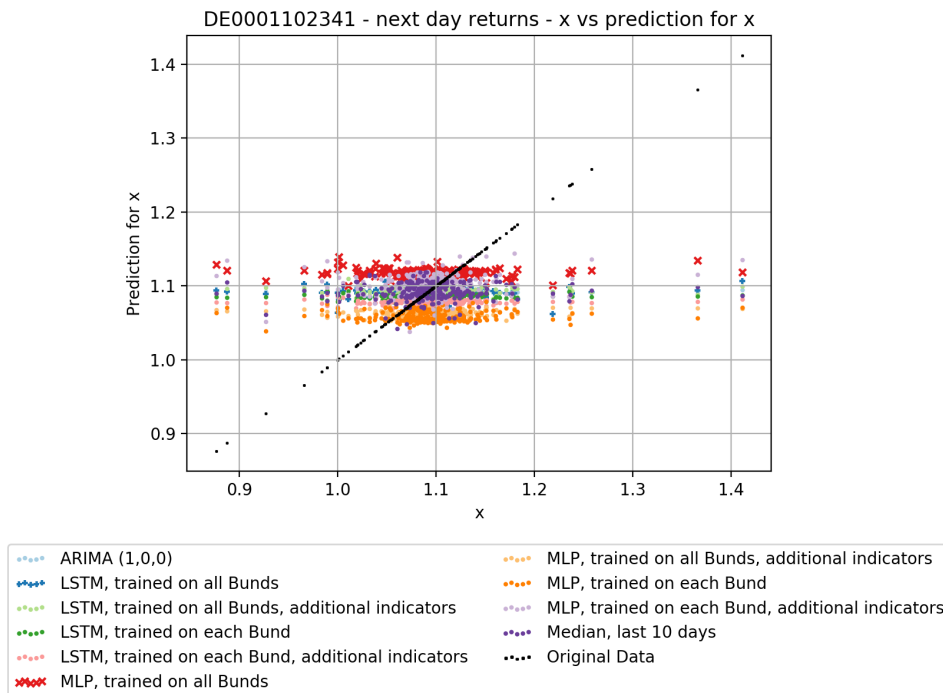


Figure 41: Next day return result visualization, plot of x vs \hat{x} .

trained on all Bunds on the fused data set and the ARIMA (1,0,0) model.

As for the next day return forecast, results are comparable for most of the models. Mape (median) is normally around 3% with exceptions for the linear and the naive model. Values for mse (median) are also close by each other with the same exceptions. The performance of the naive model is not surprising, as it has been argued already, that is it not well suited for both, return and next week forecast. The combination of the two then yields no satisfactory results as expected. Only the linear model performs worse. One explanation might be that the linear approximation and the non-stationarity do not work well together.

In contrast to the comparison between next day and next week price, which were similar, but a tendency of better performance was seen for the next day price forecasts, results for next day return and next week return are even closer. The reason for this however does not seem to be the better quality of the next week return forecasts. Rather, the forecasts for next day and next week return seem to follow the same pattern of circling a constant, which, in contrast to price forecasts, works better for the stationary return.

This behavior of the two best performing models, ARIMA and LSTM, of circling around a constant is also shown in graph (44) of the forecast for first 50 days in the test set.

model	mape (median)	mape (mean)	mse (median)	mse (mean)
average, 3 days (43)	0.344	0.42	0.26	7.55
linear, 3 days (43)	0.410	0.48	0.35	5.14
median, 3 days (43)	0.495	0.57	0.49	4.84
naive, 1 day (43)	1.409	1.70	0.96	89.66
arima (1,1,0) (45)	0.408	0.54	0.40	9.77
<i>mlp, trained on all Bunds:</i>				
mlp (10,10,10) (47)	0.516	0.83	0.53	17.56
mlp (20,10), four ind. (49)	0.643	0.80	0.75	2.27
<i>mlp, trained on each Bund:</i>				
mlp (20,10) (55)	1.231	1.40	0.23	0.81
mlp (10,25), seven ind. (57)	1.207	1.40	0.46	1.12
<i>lstm, trained on all Bunds:</i>				
lstm (10), 3 days (63)	0.385	0.59	0.37	1.53
lstm (30,25,10), 3 days, four ind. (64)	0.338	0.54	0.23	1.38
<i>lstm, trained on each Bund:</i>				
lstm (30,10), 3 days ((67))	1.060	1.50	4.06	5.18
lstm (10,25,10), 3 days, four ind. (68)	1.570	2.84	5.45	22.33

Table 11: Next week price forecast, best performing model configurations per class. Notations as in chapter Experimental Results (6).

The visualization (45) of the results of x (on the x-axis) versus \hat{x} (on the y-axis) is similar to next day return forecasts.

Also in this specific scenario of next week return forecasts, it can be observed, that almost all models tends to circle a constant value, which supports the insight from the earlier plot. The best two models for this task, the ARIMA and the LSTM model (trained on all Bunds), are highlighted in the plot with different symbols. The impression, which arises from the plot, is, that these two models do not clearly outperform other models plotted. This corresponds to the information in the table.

The only plot, which significantly differs visually, is the plot for the MLP model, trained on all Bunds and on the fused data set. This model consequently is not close to a constant. Moreover, when considering the metrics of the table, it does not outperform the models with a more constant behavior.

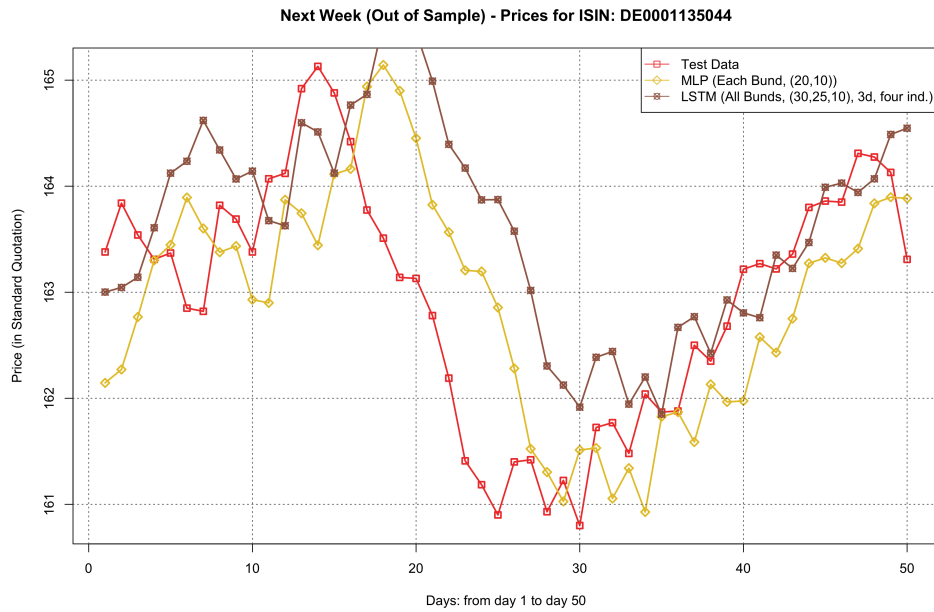


Figure 42: Next week price result visualization of the two best performing models on a randomly selected ISIN, limited to the first 50 days of the test data set.

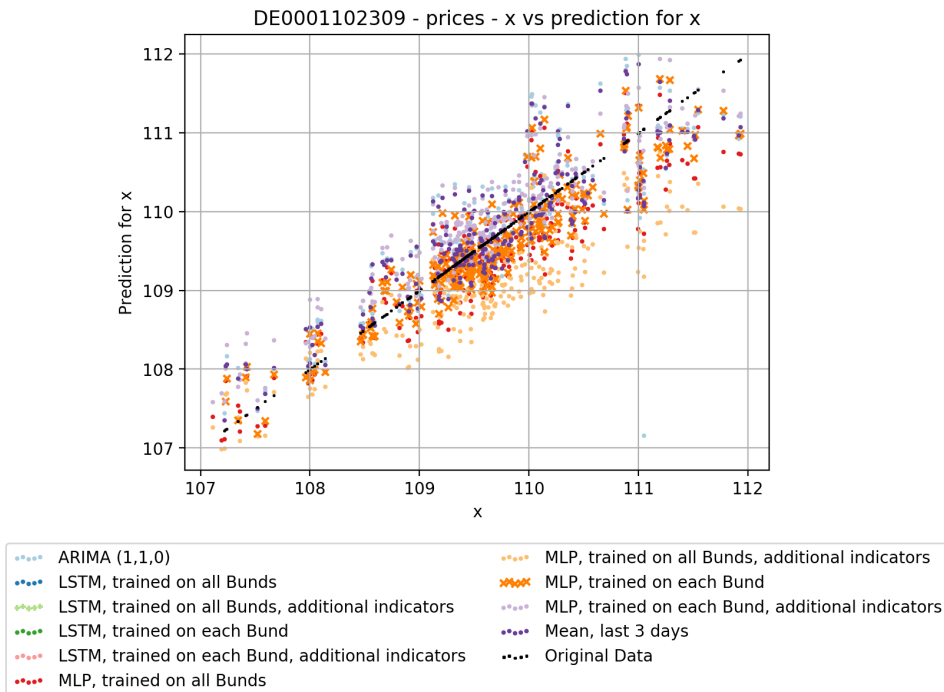


Figure 43: Next week price result visualization, plot of x vs \hat{x} .

model	mape (median)	mape (mean)	mse (median)	mse (mean)
average, 10 days (44)	3.580	3.800	0.00426	0.0046
linear, 10 days (44)	6.130	6.310	0.00971	0.0104
median, 10 days (44)	3.550	3.740	0.00415	0.0045
naive, 1 day (44)	4.930	5.120	0.00805	0.0085
arima (1,0,0) (46)	3.280	3.474	0.00389	0.0042
<i>mlp, trained on all Bunds:</i>				
mlp (30,20,10) (51)	3.630	4.700	0.00442	0.0070
mlp (20,10), four ind. (53)	3.600	4.100	0.00440	0.0048
<i>mlp, trained on each Bund:</i>				
mlp (10,20,10) (59)	3.534	3.760	0.00417	0.0044
mlp (25,20,10), four ind.(61)	3.511	4.093	0.00400	0.0050
<i>lstm, trained on all Bunds:</i>				
lstm (10,10,25), 3 days (65)	3.290	3.472	0.00400	0.0040
lstm (20,25,20), 3 days, four ind. (66)	3.279	3.294	0.00400	0.0039
<i>lstm, trained on each Bund:</i>				
lstm (10,20,30), 3 days (69)	3.571	4.245	0.00440	0.0047
lstm (30,10,30), 10 days, four ind. (70)	3.496	5.483	0.00400	0.0130

Table 12: Next week return forecast, best performing model configurations per class. Notations as in chapter Experimental Results (6).

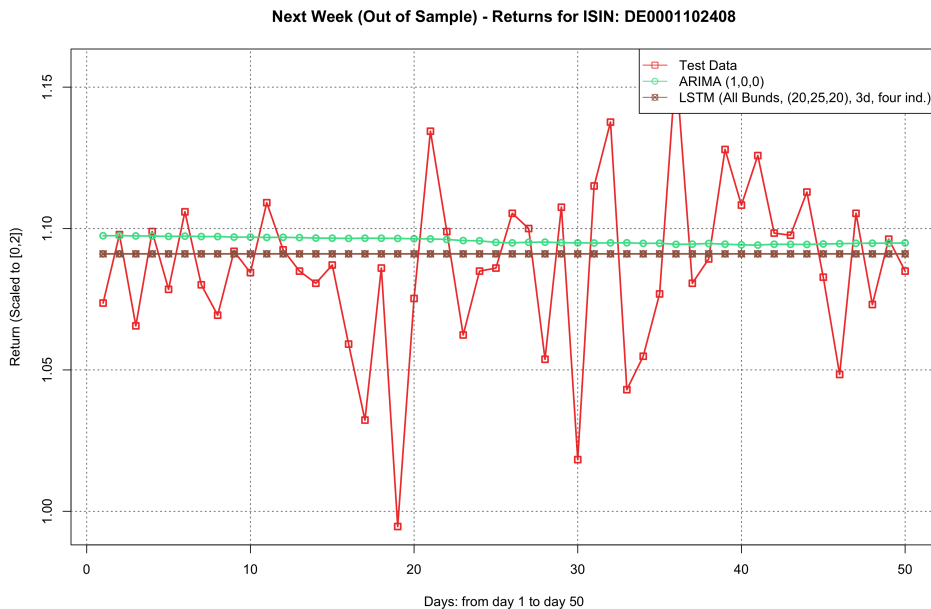


Figure 44: Next week return result visualization of the two best performing models on a randomly selected ISIN, limited to the first 50 days of the test data set.

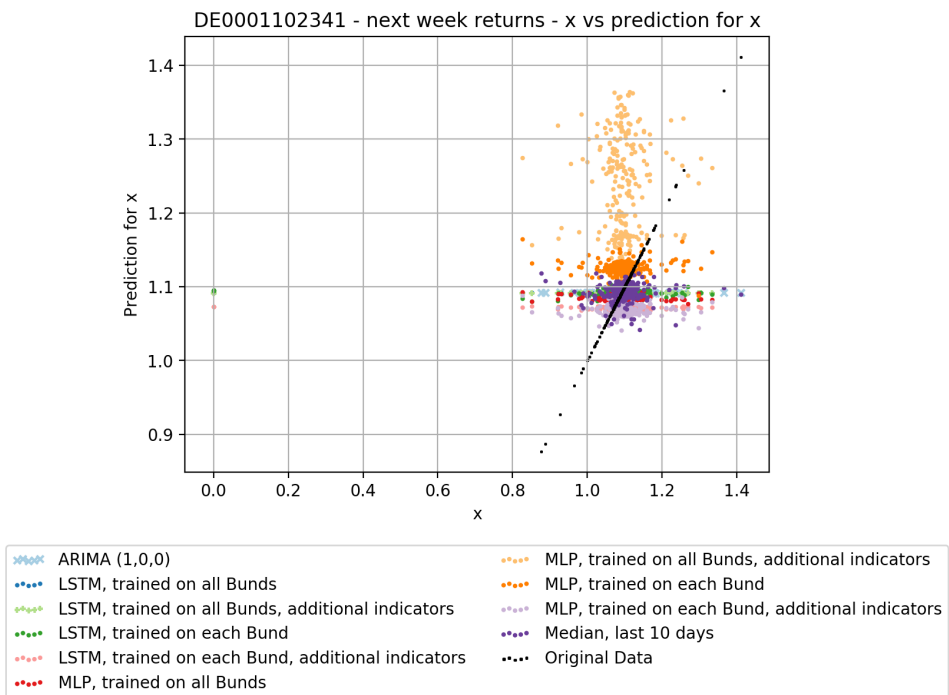


Figure 45: Next week return result visualization, plot of x vs \hat{x} .

7 Discussion

The aim of this chapter is to discuss the large amount of experimental results, which have been introduced in the prior section. Emerging from this discussion, the first important part of this chapter is to assess the hypotheses, which were established in the introduction of this work. Second, additional findings are discussed. Finally, questions, which are still open or arose in the process of developing this work and might be relevant for future research, are formulated.

7.1 Evaluating Introductory Hypotheses

The next lines of this section are concerned with arguing for each of the four initially formulated hypotheses, if they are supported by the experiments of this work or if they should be dropped.

7.1.1 Hypothesis 1

[Hypothesis 1]: Government bond price development can be modelled by LSTMs on the basis of technical bond data and outperforms established computational models, including recent neural network based techniques.

Modelling government bond prices by LSTMs seems generally possible, as long as LSTMs are not trained on each Bund separately. The hypothesis is supported by the experiments and the results obtained in this work. Although not all neural network based approaches (LSTM, on each Bund) performed as expected, most neural network based approaches performed well, in many cases even best for the specific task 6. If the slight improvement in performance is worth the additional complexity is a different question. For certain tasks LSTMs outperform established methods. Generally, the advantages of neural network based models are reflected in this work. Of the eight best performances, two are reported for each combination next {day,week} and {price,return}, six are neural network based.

The following figure (46) depicts the mape for model for a random selection of 10 Bunds for the task of next day price forecasts.

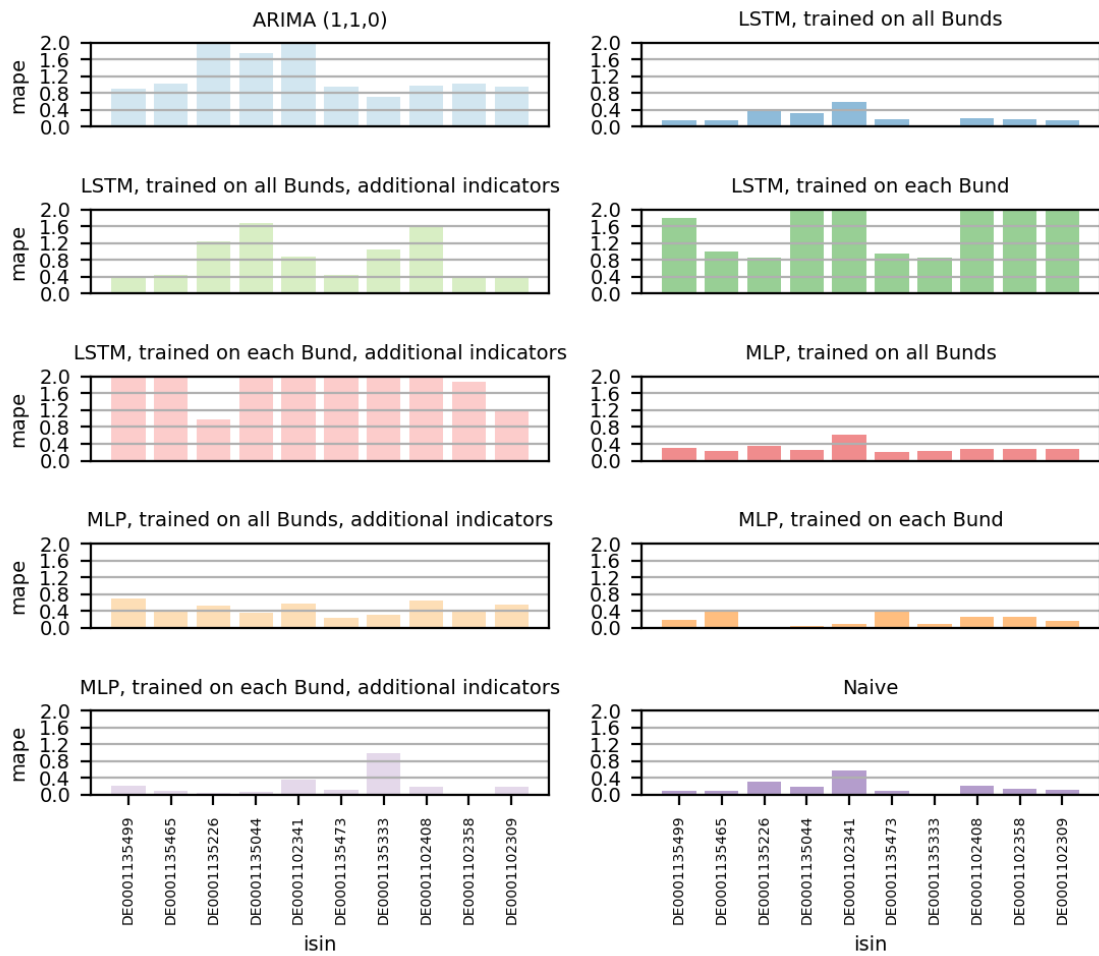


Figure 46: Maape per ISIN per Model for next day price forecasts.

Other figures of the mean absolute percentage error per model for a selection of 10 Bunds can be found in the appendix (61, 62) and in the next paragraph (47). It is observed, that MLPs, trained on each Bund and on all Bunds produce high quality results. When considering the mse (median), MLPs score highest results with a mse (median) of 0.01. Also, the naive model is suited for this task, best performing with respect to a maape (median) of 0.14. LSTMs, if they are trained on all Bunds, achieve second best performance with respect to a maape (median) of 0.19 and second best performance in terms of mse (median) with 0.05 (9). On the contrary, training them on each Bund and consequently evaluating LSTM models for each model for each Bund does not work well. For next day return forecasts, LSTM models achieve the best maape (median) score with 3.22 (10). For next week price forecasts, LSTM models are most successful in terms of maape (median), 0.338, and with respect to mse (median), 0.23 (11). Also, for next week return forecasts, LSTM models accomplish the best maape (median) score of 3.279.

Certainly surprising is the large difference between MLP and LSTM models trained on each Bund. While MLP models (on each Bund) perform best in certain cases and often very well, LSTM models (on each Bund) are in many cases (just as the one depicted in the figure above) the ones with worst performance. A convincing explanation is still to be sought. One possible reason might be the rather small number of training epochs for the LSTM models. The LSTM, trained on all Bunds, might be able to avoid this problem due to the larger number of training samples in every epoch.

Another noteworthy fact about the neural network based approaches is the one, that architectural choices seem less important than expected. Best performing MLP and LSTM models for different tasks have a variety of different units per layer and also a different number of layers. When consulting the full experimental results in the appendix for each of the results tables, it becomes clear, that the results of different model architectures for a specific task and training setup is often rather small. This indicates, that the tasks as posed in this work, and with the data available, can be solved by many of the tested architectures, at least to the extent presented in this work.

All in all, the neural network based approaches achieve better performances than ARIMA or less complex models. Two questions however remain open, which are discussed later again: (a) is the minor gain in performance obtained by the neural networks based performances worth their higher complexity and consequently their significantly longer computation times and (b) do the neural network based approaches really learn much when it comes to return forecasting? Their almost constant forecasts answer this question in the negative.

The superiority of neural network based approaches seems evident to a certain extent. As the hypothesis suggests, however, LSTM models do not seem to perform better than MLP models. Although LSTM models show best performance for some tasks, MLP models are performing often similarly, sometimes even better (6). That LSTM models generally outperform MLP models, is not supported by the results of this work.

7.1.2 Hypothesis 2

[Hypothesis 2]: Enriching the technical bond data by economic features improves the overall model performance of neural network based models in forecasting future bond development.

For MLP model forecasts, this does not enhance performance significantly. For next day price forecast for example, MLP models trained on all Bunds obtain a mape (median) score of 0.27 on the original data and a score of 0.41 on the fused data (9). Trained on each Bund, there is improvement, but it is small. From a mape (median) of 1.25, it declines to 1.22. For LSTM models, trained on all Bunds, next day price forecast performances are almost identical for both, training on fused and original data with a mape (median) of 0.06 and 0.05 (9). For next day return forecasting, results for both, MLP and LSTM models, are similar. Training on the fused data does not increase performance. On the contrary, next week forecasts seem to profit from training on the fused data set. With respect to mape (median), the LSTM models, trained on all Bunds and on the fused data set, accomplish top performance with a mape of 0.338 for next week price forecasts and 3.279 for next week return forecasts. Enriching the feature space with economic data, at least in the way it has been done in this work, might help in certain scenarios, especially for longer forecasting horizons. Not much evidence however is obtained, that this adds value in general.

7.1.3 Hypothesis 3

[Hypothesis 3]: Neural network based models outperform other established methods for an increased forecasting horizon.

How do next day and next week forecasts compare, both in price and return? Already mentioned is the apparent superiority of price forecasts over return forecasts. The difference between next day and next week forecasts is on the contrary rather small.

Target: Price When targeting price, a small difference in performance metrics can be observed (9, 11). Forecasting the next day seems easier than forecasting the next week, which answers the expectations. Best next day price forecasting performance with respect to mse (median) is 0.01 obtained by the MLP model, trained on each Bund. For next week price forecasts, LSTM and MLP models perform equally well in terms of mse (median) with a value of 0.23. When trying to probe the causes of these results, and especially when looking at the corresponding plots for next day and next week price forecasts (38, 40), the reason behind this,

is possibly the characteristic, that best performing models are similar to the naive approach of forecasting values close to the current value. For next week forecasts, with larger variations than for next day forecasts. This *rolling forward* of the current value seems to be the best strategy, naturally combined with small adaptations, for the tasks at hand.

Target: Return The *rolling forward* of the current value is not successful for return values. Both, for next day and next week. Models forecasting the return on the contrary, are inclined to circle around constants (40, 44). The almost constant forecast then does not yield a large discrepancy in next day and next week return forecast, which manifests itself in the top performances. Top next day return performance, measured in mape (median) is 3.22 by the LSTM model (10). Best next week return mape (median) scores, are also obtained by LSTM models with a value of 3.28.

In next week forecasts, neural network based approaches perform better than other approaches. The difference between mape and mse for neural network based approaches and easier methods also increases slightly (9, 11). For next week forecasts, LSTM models also outperform MLP models. This could indicate, that a longer time horizon improves neural network based approaches' performance, especially LSTM models, compared to less complex models, ARIMA or MLP models. However, the difference is rather small and at least for the chosen time horizons, the variation might be negligible.

7.1.4 Hypothesis 4

[Hypothesis 4]: Neural networks for the task of return forecasting outperform neural network aiming for price forecasting.

The forecasts for both, next day and next week price, are, in terms of the mean absolute percentage error, better than the forecasts for the corresponding return forecasts (9, 10, 11, 12). Next day price forecasts range from a mape (median) of 0.19 to 2.99. Price forecasts for the next week achieve a mape (median) between 0.338 and 1.5 for neural network based models. Next day return forecasts result in a mape (median) of in between 3.22 and 3.69. Next week return forecasts scores range from 3.279 to 3.630.

This is surprising and does comply with the expectations formulated in the in-

roduction to this work. Among others, missing stationarity was expected to cause problems for neural network based price forecasts, while return forecast should have benefited from this features. On the contrary, return forecasts perform worse than expected, which is also clearly shown in the plots (38, 39, 40, 41). The circling around constant values of neural network based approaches (and others) does not convey the impression, that much is learned. The general superiority of price forecast over return forecasts can be inferred from the plots of the mean absolute percentage errors per model for a random selection of Bunds. The corresponding price plot (46) shows lower mape scores than the corresponding return plot (47) below, which can be seen easily, when consulting the y-axes of both plots.

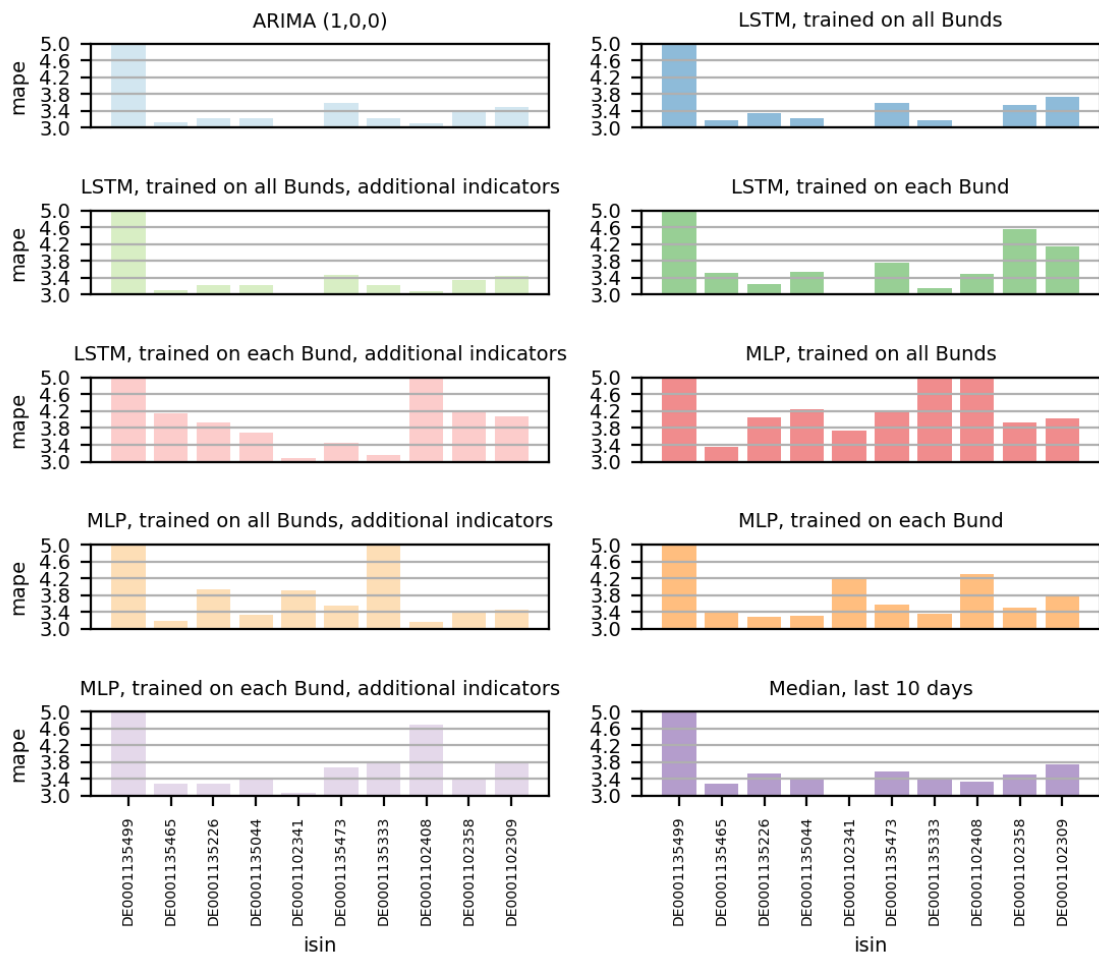


Figure 47: Mape per ISIN per Model for next day return forecasts.

Evidence collected in the experiments of this work, points in the very opposite direction of this [Hypothesis 4]. Although theoretically, there a good reasons to believe, that this hypothesis is valid, in this work’s experiments, the results indicate

the opposite. Price forecasting achieved more satisfactory results than return forecasting (9, 10). First ideas for this unexpected behavior have been discussed, but further explanations for this are sought.

7.2 Open Questions

The presented results and the evaluation of the initially formulated hypotheses pose some further questions, which are briefly addressed in this subsection. For a few questions, first answers are provided, whereas some remain open for future research in this area.

1. *Can differences in performances for specific Bunds be observed?*

This important question has not been addressed yet, which is why a few thoughts on this are explained. The theoretical background for not distinguishing between the obvious difference in 10 and 30 year Bunds has been provided in earlier sections of this work (3). In practice, one difference between 10 year and 30 year Bunds is the increased variance in prices of 30 year Bunds due to their longer runtime and higher exposure to risk. Figure (48) shows for each model and ten randomly selected Bunds, the Bunds' variance in price and the mape of the best model per class. Although the difference is not large, it seems that, especially for well performing models like the MLP (on each Bund), the Naive, and the LSTM (on all Bunds), the mean absolute percentage errors increases with increasing variance. When thinking about this, this might not be as surprising and it does not conflict with the earlier statement, that theoretically 10 year and 30 year Bunds can be modelled with one model. Instead, what might be useful to incorporate in future models, is the variance, which seems to be connected to the remaining time to maturity. This does not directly connect to 10 and 30 year Bunds, as in the given training data set, there are 10 year Bunds with larger maturity than 30 year Bunds and vice versa. Consequently, the development of different models for ranges of remaining time to maturity might be interesting. For the return, the observations of the variance of the price data, do not hold (Figure 49). Variance in return seems not connected to runtimes of Bunds. Error metrics also do not seem to be strongly effected by variance of returns.

2. *Can certain scenarios be identified, where some models seem to be more suitable*

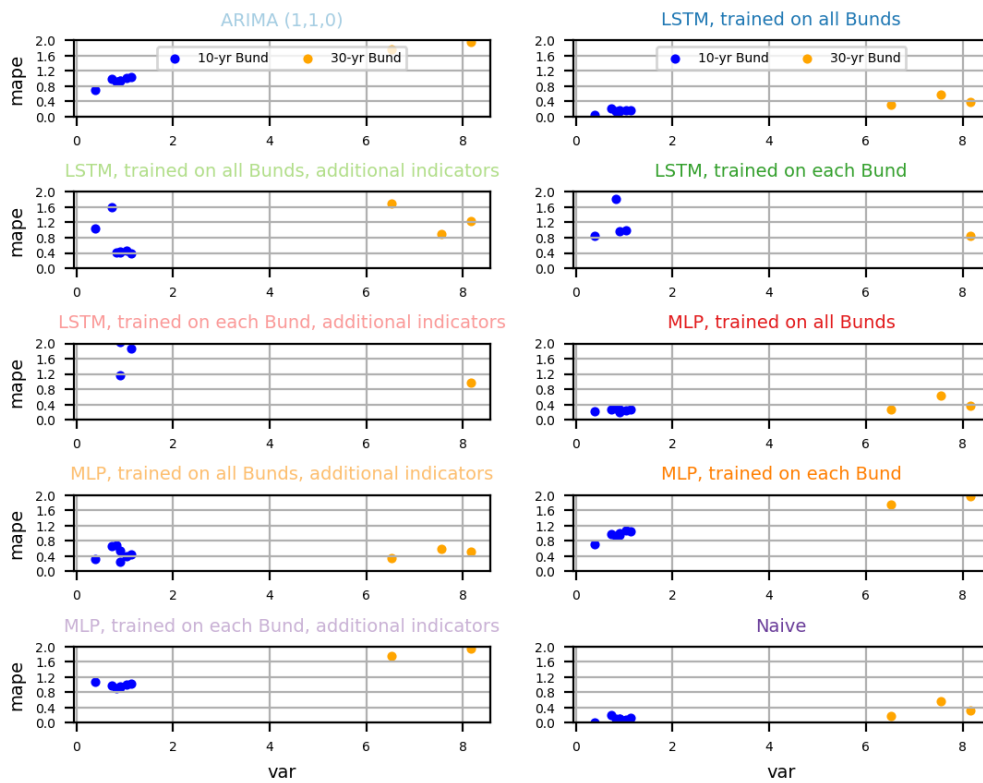


Figure 48: Plot of variance vs mape per model of next day price forecasts.

than others?

Apart from the models, which have performed best, for the given task, are there more scenarios, in which certain models are to be preferred over others? Already mentioned is the idea, that certain models might be more successful for shorter remaining time to maturity, other might be more successful for longer remaining time to maturity. Are there other factors? Possibly, lower interest rates or coupons might be modelled by other models than higher interest rates or coupons. Further differentiation of this rather global approach in this work might be useful.

3. *Is it desirable to test more or different models for further improvement of results?*

The number of models tested per model class was already large for the scope of this work. There is no evidence, that simply adding more layers and/or units will drive performance in the desired directions. Maybe different model types yield better performances. Recent work uses convolutional neural networks and reinforcement learning for time series forecasting ([40]). Also, a combination of LSTM to capture time dependency and convolutional networks to

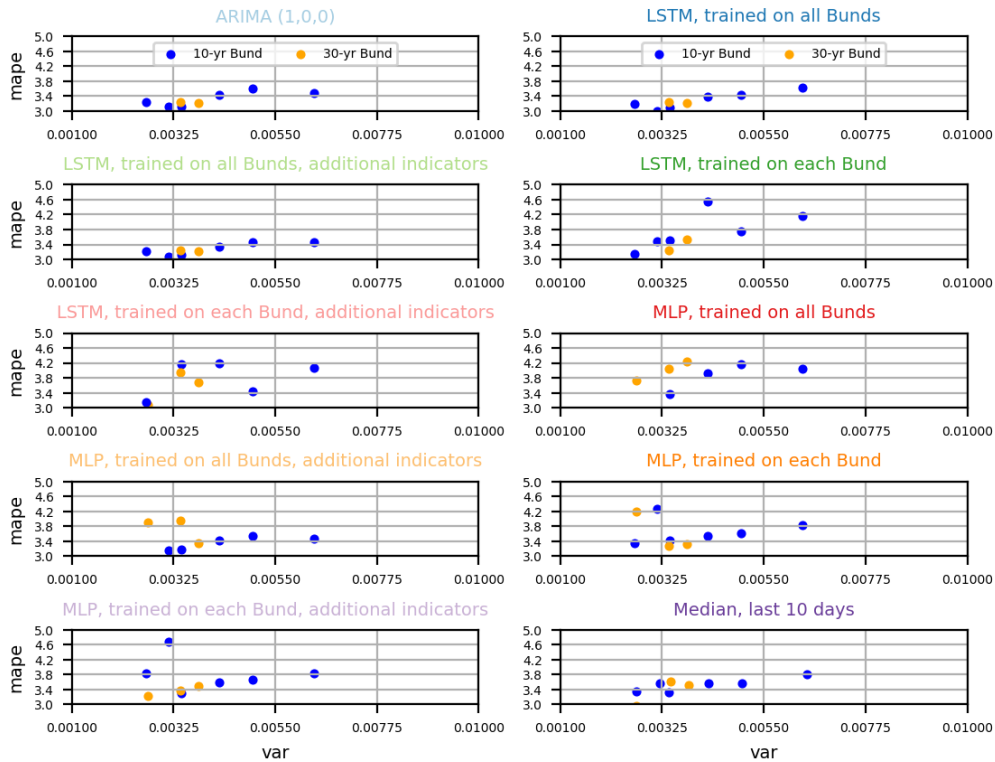


Figure 49: Plot of variance vs mape per model of next day return forecasts.

capture other aspects might be worth trying.

4. *Is the choice of economic indicators correct? Should they be modelled differently?*

The choice of economic indicators certainly could be changed. This work has not presented much evidence, that this does improve model quality notably. One problem could be the different resolution of days and months in the different data sets and the chosen approach to fusion the data. Different approaches to address this problem might be more successful.

5. *How do general models like certain neural network based perform on unseen Bunds?* This has not been evaluated within the scope of this work. However, it is expected, that neural networks trained on all Bunds perform similarly on unseen Bunds and are comparable to the provided results. This is certainly an advantage of these models, as many other models are not able to handle unseen Bunds.

6. *Could different scaling of the data before training change the results as desired?* Min-max-scaling as used for the input for the LSTM models loses more infor-

mation than normalization, for example. This is why, it is a reasonable idea to try this different approach to scale the input data. For the specific task of next day price forecasting, an LSTM model, trained on all Bunds, has been trained. The top 5 performances with respect to mape (median) are reported in table 13. When comparing the results for the two different inputs for the LSTM model, the normalized input performs worse than the min-max-scaled input. Consequently, changing the scaling method for the input does not seem to yield major benefits.

7. *One last question is regarding the number of training epochs, especially for return forecasting. Would longer training epochs be beneficial for return forecast and impact the approximately constant forecast?*

A first idea of how more epochs change the forecast is tested for the LSTM model for next day return forecasting, trained on all Bunds. The behavior of the loss function, which has been seen for the first 50 epochs already in the experimental setup, is not surprising (figure 50). It can be expected that the loss declines quickly at the beginning (the orange box in the plot). Maybe, it is expected to not decline as quickly as depicted here. However, the reason for no further improvement is, that the loss is constant after it has reached a certain (near 0) threshold (the purple box in the plot). So, as least as the task is constituted in this work, longer training does not seem to help.

model	mape (median)
lstm (10), 3 days	1.04
lstm (20), 3 days	1.07
lstm (25), 3 days	1.10
lstm (30), 3 days	1.12
lstm (10,10), 3 days	1.26

Table 13: Next day price forecast, normalization instead of scaling, trained on *all* Bunds. Notations as in chapter Experimental Results (6).

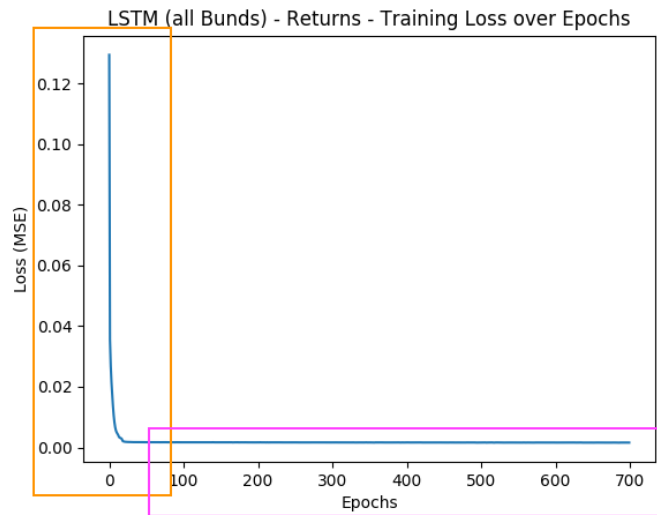


Figure 50: Longer Training does not improve LSTM model performance, trained on *all* Bunds, forecasting next day return.

8 Conclusion

The motivation of this work has been to set an example of the large potential of Artificial Intelligence in the financial industry with the help of currently in vogue techniques of neural networks and especially LSTMs. The financial industry's most prominent data type is time series data, which is notoriously hard to model. Much effort has been invested into the modelling and forecasting of stock price time series data. This work investigated the related, but far less explored topic of government bond development. In contrast to stock data, government bond data is expected to be largely driven by the economic development of the bond issuing government. Therefore, relevant data for the modelling and forecasting of government bond development is accessible to the public. Understanding government bonds in greater detail, with their large impact on global economy, and to improving capabilities of modelling their development, would be of enormous benefit to governments worldwide, to adjust economic policies and prevent financial crises.

With the publicly available bond data, and state-of-the-art methods of Artificial Intelligence, German government bonds (Bunds) have been modelled with different time horizons, targets and modelling approaches. Neural network based approaches, especially LSTM models due to their design for sequence modelling, were expected to be able to model the Bund development with high quality. Furthermore, LSTM were anticipated to outperform already introduced neural network based approaches and other, well established methods for government bond price forecasting including ARIMA models. These expectations were formulated in the following four hypotheses, which investigated within this work. The outcome of this research is summarized for each of the initially formulated hypotheses:

[Hypothesis 1]:

Government bond price development can be modelled by LSTMs on the basis of technical bond data and outperforms established computational models, including recent neural network based techniques.

It has been shown, that LSTMs are capable of modelling German government bonds. Depending on the specific task, they outperform established computational models, also neural network based ones, of which MLP models have been compared in this work. LSTMs were especially useful for next week price forecast, for which they achieved better scores than the MLP model (second best) and the average model (best of remaining tested models), as shown in table (14).

ID	model	mape (median)	std (over mape)
1	average, 3 days (43)	0.344	0.33
2	mlp (10,10,10) (55)	0.516	0.26
3	lstm (30,25,10), 3 days, four ind. (64)	0.338	0.62

Table 14: Results of next week price forecast, extract of table 11. Best performing LSTM, MLP and Average with respect to the median of the mean absolute percentage error over all Bunds are reported. Standard deviation of mape scores over all Bunds per model is listed as well.

In most of the tested scenarios, neural network based approaches outperformed established methods. However, the improvement of performance has been minor in most cases. It is questionable, if the added complexity justifies the small advancements.

[Hypothesis 2]:

Enriching the technical bond data by economic features improves the overall model performance of neural network based models in forecasting future bond development.

Interestingly, it is not obvious, that the implemented models benefit significantly from the large amounts of available economic data. Table 14 above presents LSTM models, which perform best on the fused data set. However, there the scores do not seem to be a significant improvement to training models on the original data set. This might be caused by the selection of economic data for model building in this work and the fusion technical and economical data into one data set. Certainly, further research in this area seems promising, as economic research clearly identified data, which drives government bond development, and first steps for modelling this development by modern computational methods as LSTMs are made.

[Hypothesis 3]:

Neural network based models outperform other established methods for an increased forecasting horizon.

More distant forecasting horizons do not effect model performance as strongly as one might expect. Observations show a decline in performance for more distant events in the future, but larger discrepancies might have been expected. LSTM models, trained on all Bunds and on the fused data set, achieved a mape (median)

of 0.19 and a mse (median) of 0.05. For next week price forecast, performance of the same model declines to 0.338 (mape (median)) and 0.23 (mse (median)). Consequently, mape (median) performance increases by 0.148 and mse (median) rises by 0.18. Difference in performance of the average model, is 0.164 in mape (median) and 0.19 in mse (median). Hence, more distant forecasting horizons do not only show no significant effects on model performance, in addition, LSTMs do not profit as expected from an increased forecasting horizon in comparison with established methods.

[Hypothesis 4]:

Neural networks for the task of return forecasting outperform neural networks aiming for price forecasting.

Another surprising aspect is the unanswered expectation, that return development is easier to model than price development. This is manifested in the fact, that the majority of models for price forecasting obtain a mape (median) score between close to 0% and around 1.5%. Return forecasts mostly achieve mae (median) scores of around 3%. This contrasts initially formulated expectations. First ideas, why this might be the case, such as an insufficient number of training epochs, do not seem to remedy this fact.

This thesis answered a variety of questions regarding LSTMs' ability to model German bond development. However, as discussed in chapter 7.2 in detail, a number of questions for further research in this area emerged. There is uncertainty, if the training including the scaling could be improved. First ideas did not enhance the presented results, but further investigation might lead to an advance in performance. Another technical aspect is the possibility of implementing further, deep learning inspired models, such as a combination of LSTM and convolutional networks. Furthermore, the decision for economic data, which is used in addition to technical bond data, is not set in stone. Other choices might yield improvements for bond development forecasting. Lastly, different scenarios, in which certain models forecast in satisfactory quality and also in which they leave room for improvement, need to be investigated: Which models perform well on government bonds with long remaining time to maturity? Which perform well with short remaining time? Which models capture volatile bonds? Are there models, which model Bunds with higher interest rates better than Bunds with lower interest rates, or potentially the other way around? All these aspects, which are concerned with the nature of government-

tal bonds and their impact on modelling might be worth further investigation.

In conclusion, on the one hand this extensive experimental work on Bund development forecasting confirmed expectations of LSTM models being able to model Bund development. On the other hand, other expectations remain unanswered and interesting directions for future research emerged.

9 References

- [1] Magnus Andersson, Lars Jul Overby, and Szabolcs Sebestyén. “Which news moves the euro area bond market?” In: *German economic review* 10.1 (2009), pp. 1–31.
- [2] Adebiyi A Ariyo, Adewumi O Adewumi, and Charles K Ayo. “Stock price prediction using the ARIMA model”. In: *Computer Modelling and Simulation (UKSim), 2014 UKSim-AMSS 16th International Conference on*. IEEE. 2014, pp. 106–112.
- [3] Pierluigi Balduzzi, Edwin J Elton, and T Clifton Green. “Economic news and bond prices: Evidence from the US Treasury market”. In: *Journal of financial and Quantitative analysis* 36.4 (2001), pp. 523–543.
- [4] George EP Box et al. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
- [5] Peter J Brockwell and Richard A Davis. *Introduction to time series and forecasting*. springer, 2016.
- [6] Peter J Brockwell and Richard A Davis. *Time series: theory and methods*. Springer Science & Business Media, 2013.
- [7] Bundesbank. *Bundesbank - Kurse und Renditen*. https://www.bundesbank.de/Navigation/DE/Service/Bundeswertpapiere/Kurse_und_Renditen/kurse_und_renditen.html. [Online; accessed 4-May-2018]. 2018.
- [8] StatsLab Cambridge. *Time Series*. <http://www.statslab.cam.ac.uk/~rrw1/timeseries/t.pdf>. [Online; accessed 10-Aug-2018]. 2018.
- [9] Lijuan Cao. “Support vector machines experts for time series forecasting”. In: *Neurocomputing* 51 (2003), pp. 321–339.
- [10] Rodolfo C Cavalcante et al. “Computational intelligence and financial markets: A survey and future directions”. In: *Expert Systems with Applications* 55 (2016), pp. 194–211.
- [11] Yingjun Chen and Yongtao Hao. “A feature weighted support vector machine and K-nearest neighbor algorithm for stock market indices prediction”. In: *Expert Systems with Applications* 80 (2017), pp. 340–355.
- [12] François Chollet et al. *Keras*. <https://keras.io>. 2015.

- [13] Eunsuk Chong, Chulwoo Han, and Frank C Park. “Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies”. In: *Expert Systems with Applications* 83 (2017), pp. 187–205.
- [14] Rama Cont. “Empirical properties of asset returns: stylized facts and statistical issues”. In: (2001).
- [15] Xiao Ding et al. “Deep learning for event-driven stock prediction.” In: *Ijcai*. 2015, pp. 2327–2333.
- [16] Statistical Office of the European Union (Eurostat). *Statistical Office of the European Union (Eurostat)*. <http://ec.europa.eu/eurostat/about/overview>. [Online; accessed 29-July-2018]. 2018.
- [17] WEN Fenghua et al. “Stock price prediction based on SSA and SVM”. In: *Procedia Computer Science* 31 (2014), pp. 625–631.
- [18] Deutsche Finanzagentur. *Federal Bonds*. <https://www.deutsche-finanzagentur.de/en/institutional-investors/federal-securities/federal-bonds/>. [Online; accessed 4-May-2018]. 2018.
- [19] Deutsche Finanzagentur. *Federal notes*. <https://www.deutsche-finanzagentur.de/en/institutional-investors/federal-securities/federal-notes/>. [Online; accessed 4-May-2018]. 2018.
- [20] Deutsche Finanzagentur. *Federal Treasury notes*. <https://www.deutsche-finanzagentur.de/en/institutional-investors/federal-securities/federal-treasury-notes/>. [Online; accessed 4-May-2018]. 2018.
- [21] Banque France. *Taux indicatifs des bons du Trésor et OAT*. <https://www.banque-france.fr/statistiques/taux-et-cours/taux-indicatifs-des-bons-du-tresor-et-oat>. [Online; accessed 4-May-2018]. 2018.
- [22] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York, 2001.
- [23] Swetava Ganguli and Jared Dunnmon. “Machine Learning for Better Models for Predicting Bond Prices”. In: *arXiv preprint arXiv:1705.01142* (2017).
- [24] Linda S Goldberg and Deborah Leonard. “What moves sovereign bond markets? The effects of economic news on US and German yields”. In: (2003).
- [25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [26] Alex Graves. “Generating sequences with recurrent neural networks”. In: *arXiv preprint arXiv:1308.0850* (2013).

- [27] T Clifton Green. “Economic news and the impact of trading on bond prices”. In: *The Journal of Finance* 59.3 (2004), pp. 1201–1233.
- [28] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.
- [29] Robert D. Hof. *Deep Learning*. <https://www.technologyreview.com/s/513696/deep-learning/>. [Online; accessed 10-Oct-2018]. 2013.
- [30] Ming-Wei Hsu et al. “Bridging the divide in financial market forecasting: machine learners vs. financial economists”. In: *Expert Systems with Applications* 61 (2016), pp. 215–234.
- [31] Cheng-Lung Huang and Cheng-Yi Tsai. “A hybrid SOFM-SVR with a filter-based feature selection for stock market forecasting”. In: *Expert Systems with applications* 36.2 (2009), pp. 1529–1539.
- [32] Wei Huang, Yoshiteru Nakamori, and Shou-Yang Wang. “Forecasting stock market movement direction with support vector machine”. In: *Computers & Operations Research* 32.10 (2005), pp. 2513–2522.
- [33] John C Hull and Sankarshan Basu. *Options, futures, and other derivatives*. Pearson Education India, 2016.
- [34] John Hull, Mirela Predescu, and Alan White. “The relationship between credit default swap spreads, bond yields, and credit rating announcements”. In: *Journal of Banking & Finance* 28.11 (2004), pp. 2789–2811.
- [35] Jochen R. Andritzky IMF. *Government Bonds and Their Investors: What Are the Facts and Do They Matter?* <https://www.imf.org/external/pubs/ft/wp/2012/wp12158.pdf>. [Online; accessed 10-Oct-2018]. 2012.
- [36] Investopedia. *Government Bond Definition*. <https://www.investopedia.com/terms/g/government-bond.asp>. [Online; accessed 30-Apr-2018]. 2018.
- [37] Investopedia. *Look-Ahead Bias*. <https://www.investopedia.com/terms/l/lookaheadbias.asp>. [Online; accessed 19-Sep-2018]. 2018.
- [38] Investopedia. *Time Series*. <https://www.investopedia.com/terms/t/timeseries.asp>. [Online; accessed 08-Aug-2018]. 2018.
- [39] Marko Jerkic. *Neural Network Output Units*. <https://markojerkic.com/neural-network-output-units/>. [Online; accessed 28-May-2018]. 2018.
- [40] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. “A deep reinforcement learning framework for the financial portfolio management problem”. In: *arXiv preprint arXiv:1706.10059* (2017).

- [41] Charles M Jones, Owen Lamont, and Robin L Lumsdaine. “Macroeconomic news and bond market volatility¹”. In: *Journal of Financial Economics* 47.3 (1998), pp. 315–337.
- [42] D Ashok Kumar and S Murugan. “Performance analysis of Indian stock market index using neural network time series model”. In: *Pattern Recognition, Informatics and Mobile Engineering (PRIME), 2013 International Conference on*. IEEE. 2013, pp. 72–78.
- [43] Monica Lam. “Neural network techniques for financial performance prediction: integrating fundamental and technical analysis”. In: *Decision support systems* 37.4 (2004), pp. 567–581.
- [44] Martin Långkvist, Lars Karlsson, and Amy Loutfi. “A review of unsupervised feature learning and deep learning for time-series modeling”. In: *Pattern Recognition Letters* 42 (2014), pp. 11–24.
- [45] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), p. 436.
- [46] Fei-Fei Li, Justin Johnson, and Serena Yeoung. *Convolutional Neural Networks for Visual Recognition - Training Neural Networks I*. <https://youtu.be/wEoyxE0GP2M>. [Online; accessed 29-May-2018]. 2018.
- [47] Chi-Jie Lu, Tian-Shyug Lee, and Chih-Chou Chiu. “Financial time series forecasting using independent component analysis and support vector regression”. In: *Decision Support Systems* 47.2 (2009), pp. 115–125.
- [48] Frederick R Macaulay et al. “Some theoretical problems suggested by the movements of interest rates, bond yields and stock prices in the United States since 1856”. In: *NBER Books* (1938).
- [49] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: <https://www.tensorflow.org/>.
- [50] Moody’s. *Moody’s Sovereign Ratings*. https://www.moody.com/research/Sovereign-Supranational-Rating-List--PBC_186519. [Online; accessed 30-Apr-2018]. 2018.
- [51] Andrew NG. *What data scientists should know about deep learning*. <https://www.youtube.com/watch?v=00VN0pGgBZM>. [Online; accessed 20-Apr-2018]. 2015.

- [52] Christopher Olah. *Understanding LSTM Networks*. <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. [Online; accessed 27-Feb-2018]. 2015.
- [53] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [54] PIMCO. *Everything you need to know about bonds*. <http://europe.pimco.com/EN/Education/Pages/Everythingyouneedtoknowaboutbonds.aspx>. [Online; accessed 30-Apr-2018]. 2012.
- [55] Saimadhu Polamuri. *Difference Between Softmax Function and Sigmoid Function*. <https://dataaspirant.com/2017/03/07/difference-between-softmax-function-and-sigmoid-function/>. [Online; accessed 28-May-2018]. 2018.
- [56] StatsModels Statistics in Python. *StatsModels ARIMA*. http://www.statsmodels.org/devel/generated/statsmodels.tsa.arima_model.ARIMA.html. [Online; accessed 20-Sep-2018]. 2018.
- [57] Thomson Reuters. *Thomson Reuters Eikon*. <https://amers1.login.cp.thomsonreuters.net/>. [Online; accessed 29-July-2018]. 2018.
- [58] Duke University Robert Nau Fuqua School of Business. *Identifying the numbers of AR or MA terms in an ARIMA model*. <https://people.duke.edu/~rnau/411arim3.htm>. [Online; accessed 17-May-2018]. 2018.
- [59] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016).
- [60] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), p. 533.
- [61] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural networks* 61 (2015), pp. 85–117.
- [62] Dr. Holger Schmidt. *Künstliche Intelligenz rückt auf der IT-Agenda der Unternehmen weit nach oben*. <https://netzoekonom.de/2018/05/10/kuenstliche-intelligenz-rueckt-auf-der-it-agenda-der-unternehmen-weit-nach-oben/>. [Online; accessed 22-Aug-2018]. 2018.
- [63] Marco Schreyer et al. “Detection of Anomalies in Large Scale Accounting Data using Deep Autoencoder Networks”. In: *arXiv preprint arXiv:1709.05254* (2017).

- [64] U.S. Securities and Exchange Commission. *Interest Rate Risk - When Interest Rates Go Up, Prices of Fixed-Rate Bonds Fall*. https://www.sec.gov/investor/alerts/ib_interestraterisk.pdf. [Online; accessed 06-Mar-2018]. 2013.
- [65] Bank For International Settlements. *BIS Quartely Review*. https://www.bis.org/publ/qtrpdf/r_qt1809.pdf. [Online; accessed 10-Oct-2018]. 2018.
- [66] Robert J Shiller and Andrea E Beltratti. “Stock prices and bond yields: Can their comovements be explained in terms of present value models?” In: *Journal of Monetary Economics* 30.1 (1992), pp. 25–46.
- [67] Robert Shumway and David Stoffer. *Time Series Analysis and Its Applications*. Vol. 3. Springer New York Dordrecht Heidelberg London, 2010.
- [68] Yain-Whar Si and Jiangling Yin. “OBST-based segmentation approach to financial time series”. In: *Engineering Applications of Artificial Intelligence* 26.10 (2013), pp. 2581–2596.
- [69] Benchmark Solutions. *Benchmark Bond Trade Price Challenge*. <https://www.kaggle.com/c/benchmark-bond-trade-price-challenge>. [Online; accessed 27-Feb-2018]. 2012.
- [70] Federal Office of Statistics for Germany (Statistisches Bundesamt). *Statistisches Bundesamt*. <https://www.destatis.de/DE/Startseite.html>. [Online; accessed 29-July-2018]. 2018.
- [71] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.
- [72] Graham William Taylor. *Composable, distributed-state models for high-dimensional time series*. University of Toronto Toronto, 2009.
- [73] Tensorflow. *Recurrent Neural Networks*. <https://www.tensorflow.org/tutorials/sequences/recurrent>. [Online; accessed 10-Oct-2018]. 2018.
- [74] Robert Tibshirani et al. *An introduction to statistical learning-with applications in R*. 2013.
- [75] U.S. Department of Treasury. *Daily Treasury Bill Rates*. <https://www.treasury.gov/resource-center/data-chart-center/interest-rates/Pages/TextView.aspx?data=billrates>. [Online; accessed 29-July-2018]. 2018.

- [76] Agence France Trésor. *Medium and Long-Term OATs*. http://www.aft.gouv.fr/rubriques/medium-and-long-term-oats_172.html. [Online; accessed 29-July-2018]. 2018.
- [77] Udacity. *Cross Entropy*. https://youtu.be/tRsSi_sqXjI. [Online; accessed 28-May-2018]. 2018.
- [78] Bruce Vanstone and Gavin Finnie. “An empirical methodology for developing stockmarket trading systems using artificial neural networks”. In: *Expert systems with applications* 36.3 (2009), pp. 6668–6680.
- [79] Baohua Wang, Hejiao Huang, and Xiaolong Wang. “A novel text mining approach to financial time series forecasting”. In: *Neurocomputing* 83 (2012), pp. 136–145.
- [80] Jian-Zhou Wang et al. “Forecasting stock indices with back propagation neural network”. In: *Expert Systems with Applications* 38.11 (2011), pp. 14346–14355.
- [81] Bin Weng, Mohamed A Ahmed, and Fadel M Megahed. “Stock market one-day ahead movement prediction using disparate data sources”. In: *Expert Systems with Applications* 79 (2017), pp. 153–163.
- [82] Wikipedia. *Autoregressive–moving-average model*. https://en.wikipedia.org/wiki/Autoregressive%E2%80%93moving-average_model. [Online; accessed 9-May-2018]. 2018.
- [83] Wikipedia. *List of countries by rating*. https://en.wikipedia.org/wiki/List_of_countries_by_credit_rating. [Online; accessed 4-May-2018]. 2018.
- [84] G Peter Zhang. “Time series forecasting using a hybrid ARIMA and neural network model”. In: *Neurocomputing* 50 (2003), pp. 159–175.
- [85] Xiao Zhong and David Enke. “Forecasting daily stock market return using dimensionality reduction”. In: *Expert Systems with Applications* 67 (2017), pp. 126–139.

10 Appendix

10.1 Full Experimental Results: Next Day Forecasting

This subsection reports the results of the next day forecasts. It includes forecasts for both *price* and *return* as well as forecasts based solely on the Bund data and forecasts based on Bund data *fused* with economic data. The latter are only available for the neural network based approaches of MLP and LSTM.

The extensive experimental nature of this work yields a large number of results which are fully reported here in the appendix. For the vast variety of models with different target and, for those models where it is reasonable, with additional economic data, a maximum of ten best performing results is reported per combination of model class, target, without and with economic data. Model class in this context refers to the selected models for the experiments of this work, meaning naive, linear, mean and median (which will be often referred to as *less complex models*, ARIMA model, MLP and LSTM models. So, exemplarily, for the model class MLP, there will be results reported for next day, both for price and return and both with and without the fused economic data set. For each of those combinations the ten best performing (where performance is measured as described earlier (5.7), ordered by mean of mean squared errors over all tested Bunds, are reported in this section. Below, an overview of the results found in this section as they occur is provided to help the reader navigate through the large amount of reported results.

Results: Next day forecasting 10.1:

Reporting all the results of next day forecasts.

1. Less complex models (10.1.1)
 - i. Target: price
 - ii. Target: return
2. ARIMA (10.1.2)
 - i. Target: price
 - ii. Target: return
3. MLP, trained on *all* Bunds (10.1.3)
 - i. Target: price
 - ii. Target: return
 - iii. Target: price, model with additional economic features
 - iv. Target: return, model with additional economic features

4. MLP, trained on *each* Bund (10.1.4)
 - i. Target: price
 - ii. Target: return
 - iii. Target: price, model with additional economic features
 - iv. Target: return, model with additional economic features

5. LSTM, trained on *all* Bunds (10.1.5)
 - i. Target: price
 - ii. Target: return
 - iii. Target: price, model with additional economic features
 - iv. Target: return, model with additional economic features

6. LSTM, trained on *each* Bund
 - i. Target: price
 - ii. Target: return
 - iii. Target: price, model with additional economic features
 - iv. Target: return, model with additional economic features

10.1.1 Comparison of Less Complex Models

In this part of this work, results of mean (= average), linear, median and naive models are reported, denoted here as less complex models. First, results for next day price forecasts will be listed, second results for next day return forecasts will be reported.

Next Day Price Forecast The following table (15) lists the results of the forecasts for the next day price obtained with average, linear, median and naive models (in this order) with their different configurations of how many days of the past are considered. The best performing model (in terms of the median of mean squared errors over all Bunds) in this case the **naive** approach - forecasting the next time step to be equivalent to the current one for any time step - has been the most successful model.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
average, last 3 days	4.63	0.07	0.23	0.18
average, last 5 days	4.31	0.11	0.28	0.23
average, last 10 days	4.31	0.21	0.38	0.32
linear, last 3 days	11.20	0.06	1.40	1.27
linear, last 5 days	7.11	0.06	1.40	1.27
linear, last 10 days	5.14	0.09	1.41	1.27
median, last 3 days	3.59	0.08	0.24	0.19
median, last 5 days	3.71	0.12	0.28	0.23
median, last 10 days	4.03	0.23	0.38	0.33
naive, last 1 day	6.73	0.04	0.18	0.14

Table 15: Next day price forecast results of less complex models.

Next Day Return Forecast The following table (16) lists the results of the forecasts for the next day return obtained with average, linear, median and naive models (in this order) with their different configurations of how many days of the past are considered. The best performing model (in terms of the median of mean squared errors over all Bunds) in this case the **median** approach - taking the last 10 days at any given point of time into account - has been the most successful model.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
average, last 3 days	0.00545	0.00501	4.09	3.93
average, last 5 days	0.00495	0.00482	3.88	3.70
average, last 10 days	0.00454	0.00421	3.72	3.53
linear, last 3 days	0.01365	0.01204	6.89	6.66
linear, last 5 days	0.00851	0.00785	5.62	5.45
linear, last 10 days	0.00604	0.00570	4.74	4.51
median, last 3 days	0.00548	0.00527	4.10	3.92
median, last 5 days	0.00499	0.00483	3.91	3.71
median, last 10 days	0.00439	0.00412	3.66	3.50
naive, last 1 day	0.00821	0.00755	4.86	4.66

Table 16: Next day return forecast results of less complex models.

10.1.2 Comparison of Performance of ARIMA Models

In this part of this work, results of different ARIMA models will be reported. First, results for next day price forecasts will be listed, second results for next day return forecasts will be reported. ARIMA models for the next day price forecast include ARIMA models with $p \in 1, 3$, $d = 1$, and $q \in 0, 1$. ARIMA models for the next day return forecast include ARIMA models with $p \in 1, 3$, $d = 0$, and $q \in 0, 1$.

Next Day Price Forecast The following table (17) lists the results of the the next day price forecast obtained with ARIMA (1,1,0), ARIMA (1,1,1), ARIMA (3,1,0) and ARIMA (3,1,1) models (in this order). The best performing model (in terms of the median of mean squared errors over all Bunds) in this case the **ARIMA 1,1,0** model has been the most successful model.

model	mse (mean)	mse (median)	mape (mean)	mape (median)	failed
arima (1,1,0)	9.302	0.139	1.461	1.2653	0
arima (1,1,1)	9.310	0.140	1.460	1.2653	32
arima (3,1,0)	12.180	0.140	1.482	1.2651	0
arima (3,1,1)	12.183	0.141	1.481	1.2655	12

Table 17: Next day price forecast results of arima models. Ordered by mse (median) and mape (median)

Next Day Return Forecast The following table (18) lists the results of the the next day return forecast obtained with ARIMA (1,0,0), ARIMA (1,0,1), ARIMA (3,0,0) and ARIMA (3,0,1) models (in this order). The best performing model (in terms of the median of mean squared errors over all Bunds) in this case the **ARIMA 1,0,0** model has been the most successful model.

model	mse (mean)	mse (median)	mape (mean)	mape (median)	failed
arima (1,0,0)	0.00416	0.0038	3.485	3.297	0
arima (1,0,1)	0.00417	0.0039	3.495	3.302	20
arima (3,0,0)	0.0041895	0.0039	3.515	3.305	0
arima (3,0,1)	0.0041984	0.0040	3.529	3.327	9

Table 18: Next day return forecast results of arima models. Ordered by mse (median) and mape (median)

10.1.3 Comparison of Performance of MLP Models, Trained on All Bunds

In this part of this work, results of different MLP models, which were trained on *all* Bunds, will be reported. First, results for next day price forecasts will be listed, second results for next day return forecasts will be reported. For both of the two different targets, results will be reported for forecasts on the original Bund data set as well as for forecasts on the fused data set, including economic features. In addition, for each best performing model per task, results for the same model configuration but with a different random initialization will be reported.

As described in detail in the prior section of experimental setup (5.3.5), MLP architectures range from one to three hidden layers containing 10, 20, 25 or 30 units each. Reporting all 84 architectures would require unnecessary attention of the reader, which is why it has been decided to only report the top 10 performing models. Performance hereby is ranked again by the median of mean squared errors over all the tested Bunds. Other performance metrics as introduced in the experimental setup (5.7) are reported as well.

Next Day Price Forecast The following table (19) lists the results of the the next day price forecast obtained with MLP models and different architectures. The best performing model (in terms of the median of mean squared errors over all Bunds) in this case the **MLP (10)** model has been the most successful model. Models in the table are ordered by the column *mse (median)*. Training is conducted on *all* Bunds.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (10)	0.337	0.121	0.328	0.271
mlp (25,20,10)	0.376	0.209	0.368	0.313
mlp (25,25)	0.467	0.19	0.4	0.331
mlp (20,20,30)	0.465	0.257	0.423	0.344
mlp (30,30)	0.597	0.231	0.435	0.349
mlp (30,30,30)	0.577	0.238	0.445	0.363
mlp (20,20,20)	0.741	0.353	0.488	0.39
mlp (20,10,25)	0.725	0.329	0.489	0.393
mlp (25,20,20)	1.312	0.407	0.615	0.407
mlp (30,20,10)	0.904	0.47	0.517	0.408

Table 19: Ten best performing next day price forecast results of MLP models trained on **all** Bunds, ordered by mape (median). MLP (x) stands for a MLP with one hidden layer and x units. MLP (x,y) stands for a MLP with two hidden layers and x units in the first hidden layer and y in the second.

Different Random Initializations for Best Next Day Price Forecast Model

The following table (20) lists the results of the the next day price forecast obtained with the already seen MLP (10) model, which performed best on the task of next day price forecasts, when training started with five other random initializations. Training is conducted on *all* Bunds.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (10), seed=5555	0.2667	0.1044	0.2835	0.2360
mlp (10), seed=9999	0.4325	0.1149	0.3389	0.2444
mlp (10), seed=1234	0.3374	0.1215	0.3282	0.2705
mlp (10), seed=6789	0.4868	0.2236	0.4202	0.3154
mlp (10), seed=5678	1.0689	0.2838	0.5800	0.3899

Table 20: Five different initializations of best performing next day price forecast MLP model, trained on **all** Bunds, ordered by mape (median).

Next Day Price Forecast on Fused Data The following table (21) lists the results of the the next day price forecast obtained with MLP models and different

architectures but in addition to prior results trained on the fused data set. The best performing model (in terms of the median of mean squared errors over all Bunds) in this case the **MLP (10), seven ind.** model has been the most successful model. Models in the table are ordered by the column *mse (median)*. Training is conducted on *all* Bunds.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (20,10), four ind.	0.945	0.474	0.515	0.412
mlp (10), seven ind.	0.994	0.384	0.572	0.466
mlp (10), four ind.	2.413	0.717	0.871	0.474
mlp (20,10,20), four ind.	1.951	0.522	0.728	0.503
mlp (25,30,30), four ind.	2.536	0.479	0.83	0.517
mlp (25,10,30), four ind.	1.081	0.737	0.612	0.533
mlp (20,10), seven ind.	1.687	0.694	0.723	0.541
mlp (20,10,10), seven ind.	6.115	0.723	1.087	0.551
mlp (20,25,30), four ind.	1.347	0.727	0.709	0.551
mlp (30,10), seven ind.	1.186	0.495	0.701	0.552

Table 21: Ten best performing next day price forecast results MLP models trained on **all** Bunds and fused data, ordered by mape (median). MLP (x) stands for a MLP with one hidden layer and x units. MLP (x,y) stands for a MLP with two hidden layers and x units in the first hidden layer and y in the second.

Different Random Initializations for Best Next Day Price Forecast Model, Trained on Fused Data The following table (22) lists the results of the the next day price forecast obtained with the already seen MLP (10), seven ind., model, which performed best (with regards to mse) on the task of next day price forecasts on the fused data set, when training started with five other random initializations. Training is conducted on *all* Bunds.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (10), seed=4567	2.0807	0.3567	0.6879	0.4037
mlp (10), seed=5555	1.0378	0.3760	0.6001	0.4586
mlp (10), seed=1234	0.9944	0.3840	0.5725	0.4659
mlp (10), seed=6666	3.9058	0.4036	1.0435	0.5558
mlp (10), seed=2222	2.1842	0.6033	0.7844	0.5512

Table 22: Five different initializations of best performing next day price forecast MLP model, trained on **all** Bunds and fused data with seven indicators.

Next Day Return Forecast The following table (23) lists the results of the the next day return forecast obtained with MLP models and different architectures. The best performing model (in terms of the median of mean squared errors over all Bunds) in this case the **MLP (30,20,10)** model has been the most successful model. Models in the table are ordered by the column *mse (median)*. Models are trained on *all* Bunds.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (20,10,30)	0.0035	0.0035	3.2751	3.2751
mlp (10,20,10)	0.0035	0.0035	3.3561	3.3561
mlp (30,20,10)	0.0036	0.0036	3.4263	3.4263
mlp (20,20)	0.0039	0.0039	3.4640	3.4640
mlp (30,20,20)	0.0036	0.0036	3.6517	3.6517
mlp (25,10,10)	0.0048	0.0048	4.3220	4.3220
mlp (10,10,20)	0.0046	0.0046	4.3549	4.3549
mlp (30,30,10)	0.0051	0.0051	4.4460	4.4460
mlp (25,30,10)	0.0042	0.0042	4.4809	4.4809
mlp (30,30,30)	0.0046	0.0046	4.4831	4.4831
mlp (25,20)	0.0051	0.0051	4.4906	4.4906

Table 23: Ten best performing next day return forecast results MLP models trained on **all** Bunds, ordered by mape (median). MLP (x) stands for a MLP with one hidden layer and x units. MLP (x,y) stands for a MLP with two hidden layers and x units in the first hidden layer and y in the second.

Different Random Initializations for Best Next Day Return Forecast Model The following table (24) lists the results of the the next day return forecast obtained with the already seen MLP (30,20,10) model, which performed best on the task of next day return forecast (with regards to mse), when training started with five other random initializations. Models are trained on *all* Bunds.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (30,20,10), seed=4444	0.00480	0.00434	4.13020	3.78059
mlp (30,20,10), seed=1234	0.00607	0.00436	4.59131	3.64575
mlp (30,20,10), seed=8888	0.00660	0.00476	4.93531	3.73337
mlp (30,20,10), seed=2222	0.00697	0.00550	5.59091	4.68932
mlp (30,20,10), seed=6666	0.00770	0.00612	5.96564	4.88125

Table 24: Five different initializations of best performing next day return forecast MLP model, trained on **all** Bunds, ordered by mape (median).

Next Day Return Forecast on Fused Data The following table (25) lists the results of the the next day return forecast obtained with MLP models and different architectures, trained on the fused data set. The best performing model (in terms of the median of mean squared errors over all Bunds) in this case the **MLP (20,10), four ind.** model has been the most successful model. Models in the table are ordered by the column *mse (median)*. Models are trained on *all* Bunds.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (20,10), four ind.	0.0046	0.0041	3.934	3.5292
mlp (20,10,10), seven ind.	0.0046	0.0043	3.9591	3.7835
mlp (20,20,25), four ind.	0.0357	0.0074	11.1314	5.0617
mlp (25,20,10), four ind.	0.016	0.0071	8.3463	5.8197
mlp (10,20), seven ind.	0.0392	0.008	9.2284	6.5168
mlp (20,10,30), seven ind.	0.0528	0.0146	14.7329	9.1361
mlp (30,30,10), seven ind.	0.0214	0.0165	11.0573	10.5383
mlp (25,30,25), four ind.	0.0396	0.0193	13.9474	10.554
mlp (20,20,20), seven ind.	0.4852	0.013	38.9305	10.7692
mlp (20,10,25), four ind.	0.0278	0.016	12.5895	10.8193

Table 25: Ten best performing next day return forecast results MLP models trained on **all** Bunds and fused data, ordered by mape (median). MLP (x) stands for a MLP with one hidden layer and x units. MLP (x,y) stands for a MLP with two hidden layers and x units in the first hidden layer and y in the second.

Different Random Initializations for Best Next Day Return Forecast Model, Trained on Fused Data The following table (26) lists the results of the the next day return forecast obtained with the already seen MLP (20,10), four ind. model, which performed best on the task of next day return forecast (with regards to mse), when training started with five other random initializations. Models are trained on *all* Bunds.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (20,10), seed=1234	0.00463	0.00415	3.93395	3.52920
mlp (20,10), seed=1111	0.03865	0.00475	8.79669	4.18204
mlp (20,10), seed=5678	0.24254	0.02372	25.99458	12.50295
mlp (20,10), seed=5555	0.05101	0.03035	17.95088	15.19884
mlp (20,10), seed=6789	0.19274	0.04139	27.79356	17.76645

Table 26: Five different initializations of best performing next day return forecast MLP model, trained on **all** Bunds and fused data with four indicators, ordered by mape (median).

10.1.4 Comparison of Performance of MLP Models, Trained on Each Bund

Similarly to the part before, this part reports results for various different MLP models. First, reporting the results for the next day price forecast and, second, reporting the results for the next day return forecast. Also as before, results for models trained solely on the Bund data but also results for models trained on the fused data set will be reported. The difference is that in this part of the work, models reported are trained on *each* Bund and not on all Bunds as before. Other configurations are identical, meaning that also different random initializations are reported as well as the top 10 architectures per task.

Next Day Price Forecast 27 Here, results for the task of next day price forecasting with MLP models, trained on *each* Bund are reported (27). Results are ordered by the median of mean squared error per Bund and best performance for this metric is obtained by the mlp (20,10) model.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (20,10)	3.2961	0.0144	1.4331	1.2583
mlp (10,30)	3.4322	0.0154	1.4553	1.2619
mlp (25,20,10)	3.3223	0.0169	1.4337	1.2619
mlp (30,30)	3.6965	0.3096	1.4791	1.2647
mlp (30,25)	3.4567	0.1005	1.4584	1.2673
mlp (10,25)	3.3632	0.0218	1.4355	1.2703
mlp (20)	3.2874	0.0299	1.4315	1.2706
mlp (20,20)	3.3363	0.0525	1.4423	1.2726
mlp (10,30,30)	3.4002	0.0414	1.4549	1.2734
mlp (10,20,30)	3.4847	0.0593	1.4601	1.2735

Table 27: Ten best performing next day price forecast results of MLP models trained on **each** Bund, ordered by mape (median). MLP (x) stands for a MLP with one hidden layer and x units. MLP (x,y) stands for a MLP with two hidden layers and x units in the first hidden layer and y in the second.

Different Random Initializations for Best Next Day Price Forecast Model, Trained on Each Bund When initializing the mlp (20,10) network with other, also random, states although the task of forecasting then next day price, the results

can differ as can be seen in the following table (28).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (20,10), seed=1111	3.2861	0.0085	1.4384	1.2699
mlp (20,10), seed=1234	3.2961	0.0144	1.4331	1.2583
mlp (20,10), seed=3456	3.278	0.0152	1.4315	1.262
mlp (20,10), seed=4567	3.2831	0.0223	1.4305	1.2691
mlp (20,10), seed=2222	3.3395	0.0284	1.4297	1.2624

Table 28: Five different initializations of best performing (with regards to mse) next day price forecast MLP model, trained on **each** Bund, ordered by mape (median).

Next Day Price Forecast on Fused Data The following table (29) reports the results for the task of next day price forecasting with MLP models, trained on *each* Bund. In addition to the results reported just before this paragraph, models were trained on the fused data set. The results in the table are ordered by the median of the mean squared errors of forecasts for all Bunds. The best performing model for this task when evaluating with the mentioned metric is the MLP (20,20,10).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (10,30), seven ind.	3.3604	0.0636	1.4094	1.2294
mlp (20,25), seven ind.	3.417	0.0846	1.431	1.2307
mlp (20,10), seven ind.	3.479	0.1609	1.4178	1.2344
mlp (10,10), seven ind.	3.3373	0.0399	1.422	1.2383
mlp (30), four ind.	3.803	0.2469	1.4703	1.243
mlp (10,20), seven ind.	3.35	0.0563	1.4092	1.2446
mlp (20,20,10), four ind.	3.4185	0.0235	1.4497	1.247
mlp (10), seven ind.	3.3407	0.0397	1.4135	1.251
mlp (30,2 0), seven ind.	3.4407	0.1823	1.4301	1.2523
mlp (20,2 0), seven ind.	3.4788	0.1272	1.4348	1.2558

Table 29: Ten best performing next day price forecast results MLP models trained on **each** Bund and fused data, ordered by mape (median). MLP (x) stands for a MLP with one hidden layer and x units. MLP (x,y) stands for a MLP with two hidden layers and x units in the first hidden layer and y in the second.

Different Random Initializations for Best Next Day Price Forecast Model, Trained on Each Bund and Fused Data Also, for the task of next day price forecasting on fused data, the best performing (with regards to mse) model will be re-trained with different random initializations to further assess its quality. Results are listed in table (30)

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (20,20,10), four ind., seed=1234	3.4185	0.0235	1.4497	1.247
mlp (20,20,10), four ind., seed=4444	3.3111	0.0301	1.4437	1.2837
mlp (20,20,10), four ind., seed=6543	3.3935	0.0865	1.4716	1.3252
mlp (20,20,10), four ind., seed=3456	3.6321	0.1292	1.474	1.327
mlp (20,20,10), four ind., seed=3333	3.4917	0.1575	1.4452	1.3291

Table 30: Five different initializations of best performing next day price forecast MLP model, trained on **each** Bund and fused data.

Next Day Return Forecast The following table (31) lists the results of the the next day return forecast obtained with MLP models and different architectures. The best performing model (in terms of the median of mean squared errors over all Bunds) in this case the **MLP (25,10,10)** model has been the most successful model. Models in the table are ordered by the column *mse (median)*. Models are trained on *each* Bund.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (25,10,10)	0.0044	0.0040	3.7514	3.5354
mlp (10,20,10)	0.0046	0.0041	3.9264	3.5374
mlp (20,20,10)	0.0052	0.0046	4.2842	3.5494
mlp (20,10,30)	0.0050	0.0043	4.1643	3.5690
mlp (30,30,10)	0.0047	0.0042	3.9992	3.6237
mlp (10,30,30)	0.0052	0.0043	4.2472	3.6677
mlp (30,20,10)	0.0055	0.0045	4.3221	3.6832
mlp (25,10,25)	0.0051	0.0044	4.2248	3.6933
mlp (20,30,20)	0.0049	0.0044	4.2212	3.7076
mlp (10,30,20)	0.0051	0.0046	4.3797	3.7133

Table 31: Ten best performing next day return forecast results MLP models trained on **each** Bund, ordered by mape (median). MLP (x) stands for a MLP with one hidden layer and x units. MLP (x,y) stands for a MLP with two hidden layers and x units in the first hidden layer and y in the second.

Different Random Initializations for Best Next Day Return Forecast Model, Trained on Each Bund Table (32), which can be found below, reports results for the best performing MLP model, MLP (25,10,10) for the task of next day return forecasts, trained on each Bund. Results are ordered by the median of mean squared errors over the forecasts for all Bunds.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (25,10,10), seed=1234	0.00439	0.00401	3.75139	3.53538
mlp (25,10,10), seed=7777	0.00465	0.00418	3.79992	3.51382
mlp (25,10,10), seed=4321	0.00472	0.00428	4.06478	3.64749
mlp (25,10,10), seed=5678	0.00474	0.00430	4.09051	3.69136
mlp (25,10,10), seed=4567	0.00493	0.00437	4.06607	3.63823

Table 32: Five different initializations of best performing next day return forecast MLP model, trained on **each** Bund, ordered by mape (median).

Next Day Return Forecast on Fused Data The following table (33) lists the results of the the next day return forecast obtained with MLP models and different architectures trained on the fused data set. The best performing model (in terms of the median of mean squared errors over all Bunds) in this case the MLP (20,30,30) model has been the most successful model. Models in the table are ordered by the column *mse (median)*. Models are trained on *each* Bund.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (20,30,30), four ind.	0.0045	0.0041	3.8863	3.6514
mlp (25,25,10), seven ind.	0.0048	0.0043	4.0880	3.7180
mlp (30,10,10), four ind.	0.0051	0.0044	4.0837	3.6429
mlp (30,25,10), four ind.	0.0050	0.0045	4.2054	3.8102
mlp (20,30), four ind.	0.0051	0.0045	4.2401	3.7652
mlp (20,10,20), seven ind.	0.0058	0.0045	4.6355	3.7510
mlp (10,20), seven ind.	0.0048	0.0045	4.1372	3.7541
mlp (10,25,30), four ind.	0.0055	0.0045	4.6164	4.2220
mlp (25,30,30), seven ind.	0.0065	0.0045	5.0582	3.8656
mlp (30,30,10), four ind.	0.0054	0.0046	4.4792	3.9782

Table 33: Ten best performing next day return forecast results MLP models trained on **each** Bund and fused data, ordered by mape (median). MLP (x) stands for a MLP with one hidden layer and x units. MLP (x,y) stands for a MLP with two hidden layers and x units in the first hidden layer and y in the second.

Different Random Initializations for Best Next Day Return Forecast Model, Trained on Each Bund and Fused Data Again, the best performing model, which is the MLP (20,30,30), for the task of next day return forecasting on fused data (trained on each Bund), is re-trained with different random initializations. Results can be found in table (34).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (20,30,30), seed=1234	0.0045	0.00414	3.88632	3.65141
mlp (20,30,30), seed=6789	0.0059	0.00426	4.43678	3.58897
mlp (20,30,30), seed=7654	0.00603	0.00446	4.68675	3.743
mlp (20,30,30), seed=2222	0.00528	0.00449	4.46158	3.79794
mlp (20,30,30), seed=2345	0.00547	0.00449	4.53185	3.66439

Table 34: Five different initializations of best performing next day return forecast MLP model, trained on **each** Bund and fused data with four indicators, ordered by mape (median).

10.1.5 Comparison of Best Performance of LSTM Models, Trained on All Bunds

In this part of this work, results of different LSTM models, which were trained on *all* Bunds, will be reported. First, results for next day price forecasts will be listed, second results for next day return forecasts will be reported. For both of the two different targets, results will be reported for forecasts on the original Bund data set as well as for forecasts on the fused data set, including economic features.

As described in detail in the prior section of experimental setup (5.3.6), LSTM architectures range from one to three hidden layers containing 10, 20, 25 or 30 units each. Reporting all 84 architectures would require unnecessary attention of the reader, especially since for the LSTM models there are two different input scenarios (incorporating last 3 or last 10 days) tested, which would equal 168 models to be reported, which is why it has been decided to only report the top 10 performing models. Performance is here ranked again by the median of mean squared errors over all the tested Bunds. Other performance metrics as introduced in the experimental setup (5.7) are reported as well.

Next Day Price Forecast The following table (35) reports the ten best performing next day price forecast results of LSTM models trained on **all** Bunds, ordered

by mape (median).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
lstm (30), 3 days	3.55	0.064	0.27	0.192
lstm (20,25), 3 days	3.81	0.068	0.33	0.192
lstm (20,30,20), 3 days	4.25	0.100	0.47	0.272
lstm (25,20,30), 3 days	5.70	0.104	0.66	0.283
lstm (30,25,30), 3 days	4.13	0.112	0.39	0.294
lstm (30,20,20), 3 days	5.73	0.121	0.62	0.306
lstm (20,20,25), 3 days	4.51	0.127	0.51	0.287
lstm (30,20,30), 3 days	3.72	0.131	0.35	0.292
lstm (25,30,25), 3 days	3.94	0.148	0.41	0.305
lstm (20,20), 3 days	4.09	0.168	0.42	0.332

Table 35: Ten best performing next day price forecast results of LSTM models trained on **all** Bunds, ordered by mape (median). LSTM (x) stands for a LSTM with one hidden layer of x LSTM cells. LSTM (x,y) stands for a LSTM with two hidden layers and x LSTM cells in the first hidden layer and y LSTM cells in the second.

Next Day Price Forecast on Fused Data The table below (36) reports the ten best performing next day price forecast results of LSTM models trained on **all** Bunds, ordered by mape (median).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
lstm (25,25,20), 3 days, four ind.	0.323	0.054	0.252	0.190
lstm (25,25,30), 3 days, four ind.	1.100	0.074	0.391	0.198
lstm (30,20,30), 3 days, four ind.	0.549	0.075	0.313	0.198
lstm (30,30), 3 days, four ind.	0.320	0.073	0.242	0.201
lstm (30), 10 days, four ind.	0.970	0.078	0.386	0.205
lstm (10,20), 3 days, four ind.	0.460	0.067	0.305	0.206
lstm (25,20,10), 3 days, four ind.	0.545	0.082	0.334	0.208
lstm (25,25), 3 days, four ind.	0.268	0.075	0.240	0.212
lstm (20,30,10), 3 days, four ind.	4.260	0.083	0.677	0.222
lstm (20,30,25), 3 days, four ind.	0.513	0.076	0.320	0.226

Table 36: Ten best performing next day price forecast results of LSTM models trained on **all** Bunds and fused data, ordered by mape (median). LSTM (x) stands for a LSTM with one hidden layer of x LSTM cells. LSTM (x,y) stands for a LSTM with two hidden layers and x LSTM cells in the first hidden layer and y LSTM cells in the second.

Next Day Return Forecast The following table (37) lists the ten best performing next day return forecast results of LSTM models trained on **all** Bunds, ordered by mape (median).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
lstm (30,30,10), 3 days	0.0042	0.0040	3.4427	3.2282
lstm (30,25,10), 3 days	0.0041	0.0039	3.4399	3.2283
lstm (30,20,30), 3 days	0.0041	0.0039	3.4405	3.2310
lstm (25,10,25), 3 days	0.0041	0.0039	3.4412	3.2311
lstm (25,25,20), 3 days	0.0041	0.0039	3.4393	3.2325
lstm (25,25,30), 3 days	0.0042	0.0039	3.4429	3.2329
lstm (30,25,30), 3 days	0.0042	0.0039	3.4482	3.2334
lstm (25,20,20), 3 days	0.0042	0.0039	3.4414	3.2345
lstm (30,25,20), 3 days	0.0042	0.0040	3.4416	3.2353
lstm (30,10,10), 3 days	0.0041	0.0039	3.4418	3.2354

Table 37: Ten best performing next day return forecast results of LSTM models trained on **all** Bunds, ordered by mape (median). LSTM (x) stands for a LSTM with one hidden layer of x LSTM cells. LSTM (x,y) stands for a LSTM with two hidden layers and x LSTM cells in the first hidden layer and y LSTM cells in the second.

Next Day Return Forecast on Fused Data Next (38), the ten best performing next day price forecast results of LSTM models trained on **all** Bunds and fused data are reported. They are ordered by mape (median).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
lstm (25,25), 10 days, four ind.	0.0045	0.0040	3.2661	3.2306
lstm (20,30), 10 days, four ind.	0.0045	0.0040	3.2624	3.2313
lstm (10,30), 10 days, four ind.	0.0045	0.0041	3.2625	3.2315
lstm (25,25,10), 10 days, four ind.	0.0045	0.0040	3.2620	3.2334
lstm (25,20,30), 3 days, four ind.	0.0045	0.0040	3.2622	3.2337
lstm (30,25,20), 10 days, four ind.	0.0045	0.0040	3.2635	3.2339
lstm (20,10,10), 10 days, four ind.	0.0045	0.0041	3.2641	3.2342
lstm (10,30,20), 10 days, four ind.	0.0045	0.0040	3.2657	3.2352
lstm (30,30,20), 10 days, four ind.	0.0045	0.0041	3.2640	3.2360
lstm (10,10,25), 10 days, four ind.	0.0045	0.0040	3.2603	3.2372

Table 38: Ten best performing next day return forecast results of LSTM models trained on **all** Bunds and fused data, ordered by mape (median). LSTM (x) stands for a LSTM with one hidden layer of x LSTM cells. LSTM (x,y) stands for a LSTM with two hidden layers and x LSTM cells in the first hidden layer and y LSTM cells in the second.

10.1.6 Comparison of Best Performance of LSTM Models, Trained on Each Bund

In this part of this work, results of different LSTM models, which were trained on *each* Bund, will be reported. First, results for next day price forecasts will be listed, second results for next day return forecasts will be reported. For both of the two different targets, results will be reported for forecasts on the original Bund data set as well as for forecasts on the fused data set, including economic features.

As described in detail in the prior section of experimental setup (5.3.6), LSTM architectures range from one to three hidden layers containing 10, 20, 25 or 30 units each. Reporting all 84 architectures would require unnecessary attention of the reader, especially since for the LSTM models there are two different input scenarios (incorporating last 3 or last 10 days) tested, which would equal 168 models to be reported, which is why it has been decided to only report the top 10 performing models. Performance is here ranked again by the median of mean squared errors over all the tested Bunds. Other performance metrics as introduced in the experimental setup (5.7) are reported as well.

Next Day Price Forecast The following table (39) reports the ten best performing next day price forecast results of LSTM models trained on **each** Bund, ordered by mape (median).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
lstm (30,10), 3 days	14.022	2.184	1.128	0.957
lstm (20,10), 3 days	12.005	3.007	1.005	1.106
lstm (25,10), 3 days	13.946	3.109	1.332	1.187
lstm (25,20), 3 days	14.493	2.870	1.410	1.209
lstm (20,25), 3 days	15.646	3.626	1.403	1.274
lstm (25), 3 days	14.279	3.539	1.346	1.290
lstm (30,20), 3 days	14.628	3.829	1.372	1.319
lstm (30), 3 days	15.750	5.857	1.529	1.568
lstm (25,25), 3 days	15.894	6.316	1.638	1.610
lstm (25,20,10), 3 days	16.767	5.469	1.540	1.642

Table 39: Ten best performing next day price forecast results of LSTM models trained on **each** Bund, ordered by mape (median). LSTM (x) stands for a LSTM with one hidden layer of x LSTM cells. LSTM (x,y) stands for a LSTM with two hidden layers and x LSTM cells in the first hidden layer and y LSTM cells in the second.

Next Day Price Forecast on Fused Data The table below (40) reports the ten best performing next day price forecast results of LSTM models trained on **each** Bund, ordered by mape (median).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
lstm (10,20), 3 days, four ind.	25.97	19.48	3.38	2.99

Table 40: Ten best performing next day price forecast results of LSTM models trained on **each** Bund and fused data, ordered by mape (median). LSTM (x) stands for a LSTM with one hidden layer of x LSTM cells. LSTM (x,y) stands for a LSTM with two hidden layers and x LSTM cells in the first hidden layer and y LSTM cells in the second.

Next Day Return Forecast The following table (41) lists the ten best performing next day return forecast results of LSTM models trained on **each** Bund, ordered

by mape (median).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
lstm (10,10,25), 3 days	0.004	0.004	3.931	3.601
lstm (10,10,30), 3 days	0.008	0.004	5.186	3.797
lstm (30,25), 3 days	0.046	0.004	9.394	3.840
lstm (20,30), 3 days	0.005	0.005	4.140	3.870
lstm (10,10,20), 3 days	0.005	0.004	4.233	3.877
lstm (25,30), 3 days	0.020	0.004	6.907	3.883
lstm (20,25), 3 days	0.006	0.005	4.954	3.967
lstm (20,20), 3 days	0.008	0.004	5.435	3.971
lstm (10,20,25), 3 days	0.006	0.005	4.691	4.066
lstm (10,25), 3 days	0.020	0.005	7.316	4.123

Table 41: Ten best performing next day return forecast results of LSTM models trained on **each** Bund, ordered by mape (median). LSTM (x) stands for a LSTM with one hidden layer of x LSTM cells. LSTM (x,y) stands for a LSTM with two hidden layers and x LSTM cells in the first hidden layer and y LSTM cells in the second.

Next Day Return Forecast on Fused Data Next (42), the ten best performing next day price forecast results of LSTM models trained on **each** Bund and fused data are reported. They are ordered by mape (median).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
lstm (10,20,30), 3 days, four ind.	0.005	0.005	3.996	3.689

Table 42: Ten best performing next day return forecast results of LSTM models trained on **each** Bund and fused data, ordered by mape (median). LSTM (x) stands for a LSTM with one hidden layer of x LSTM cells. LSTM (x,y) stands for a LSTM with two hidden layers and x LSTM cells in the first hidden layer and y LSTM cells in the second.

10.2 Full Experimental Results: Next Week Forecasting

This subsection reports the results of the next week forecasts. It includes forecasts for both *price* and *return* as well as forecasts based solely on the Bund data and

forecasts based on Bund data *fused* with economic data. The latter are only available for the neural network based approaches of MLP and LSTM.

The extensive experimental nature of this work yields a large number of results which are fully reported here the appendix. For the vast variety of models with different target and, for those models where it is reasonable, with additional economic data, a maximum of ten best performing results is reported per combination of model class, target, without and with economic data. Model class in this context refers to the selected models for the experiments of this work, meaning naive, linear, mean and median (which will be often referred to as *less complex models*, ARIMA model, MLP and LSTM models. So, exemplarily, for the model class MLP, there will be results reported for next day, both for price and return and both with and without the fused economic data set. For each of those combinations the ten best performing (where performance is measured as described earlier (5.7), ordered by median of mean squared errors or by median of mean absolute percentage errors over all tested Bunds, are reported in this section. Below, an overview of the results found in this section as they occur is provided to help the reader navigate through the large amount of reported results.

Results: Next week forecasting (10.2):

Reporting all the results of next week forecasts.

1. Less complex models (10.2.1)
 - i. Target: price
 - ii. Target: return
2. ARIMA (10.2.2)
 - i. Target: price
 - ii. Target: return
3. MLP, trained on *all* Bunds (10.2.3)
 - i. Target: price
 - ii. Target: return
 - iii. Target: price, model with additional economic features
 - iv. Target: return, model with additional economic features
4. MLP, trained on *each* Bund (10.2.4)
 - i. Target: price
 - ii. Target: return
 - iii. Target: price, model with additional economic features
 - iv. Target: return, model with additional economic features

5. LSTM, trained on *all* Bunds (10.2.5)
 - i. Target: price
 - ii. Target: return
 - iii. Target: price, model with additional economic features
 - iv. Target: return, model with additional economic features

6. LSTM, trained on *each* Bund
 - i. Target: price
 - ii. Target: return
 - iii. Target: price, model with additional economic features
 - iv. Target: return, model with additional economic features

10.2.1 Comparison of Less Complex Models

In this part of this work, results of mean (= average), linear, median and naive models are reported, denoted here as less complex models. First, results for next week price forecasts will be listed, second results for next week return forecasts will be reported.

Next Week Price Forecasts 43 The following table (43) lists the results of the forecasts for the next week price obtained with average, linear, median and naive models (in this order) with their different configurations of how many days of the past are considered. The best performing model (in terms of the median of mean squared errors over all Bunds) in this case the **average, 3 days** approach - forecasting the next time step to be equivalent to the mean of the last 3 - has been the most successful model.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
average, 3 days	7.554	0.260	0.415	0.344
average, 5 days	5.441	0.303	0.446	0.376
average, 10 days	4.403	0.310	0.451	0.379
linear, 3 days	5.144	0.351	0.482	0.410
linear, 5 days	4.546	0.359	0.485	0.412
linear, 10 days	5.105	0.467	0.567	0.491
median, 3 days	4.841	0.486	0.567	0.495
median, 5 days	9.673	0.563	1.557	1.331
median, 10 days	27.548	0.664	1.607	1.370
naive, 1 day	89.657	0.955	1.703	1.409

Table 43: Next week price forecast results of less complex models. Ordered by mse (median) and mape (median)

Next Week Return Forecast The following table (44) lists the results of the forecasts for the next week return obtained with average, linear, median and naive models (in this order) with their different configurations of how many days of the past are considered. The best performing model (in terms of the median of mean squared errors over all Bunds) in this case the **median, 10 days** approach - forecasting the next time step to be equivalent to the median of the last 10 days - has been the most successful model.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
average, 3 days	0.00558	0.00534	4.23	4.01
average, 5 days	0.00503	0.00470	4.00	3.81
average, 10 days	0.00460	0.00426	3.80	3.58
linear, 3 days	0.10681	0.09982	17.82	17.55
linear, 5 days	0.03176	0.03031	10.38	10.02
linear, 10 days	0.01037	0.00971	6.31	6.13
median, 3 days	0.00561	0.00551	4.25	4.11
median, 5 days	0.00498	0.00481	4.01	3.89
median, 10 days	0.00449	0.00415	3.74	3.55
naive, 1 day	0.00846	0.00805	5.12	4.93

Table 44: Next week return forecast results of less complex models. Ordered by mse (median) and mape (median).

10.2.2 Comparison of Performance of ARIMA Models

In this part of this work, results of different ARIMA models will be reported. First, results for next week price forecasts will be listed, second results for next week return forecasts will be reported. ARIMA models for the next week price forecast include ARIMA models with $p \in 1, 3$, $d = 1$, and $q \in 0, 1$. ARIMA models for the next week return forecast include ARIMA models with $p \in 1, 3$, $d = 0$, and $q \in 0, 1$.

Next Week Price Forecast The following table (45) lists the results of the the next week price forecast obtained with ARIMA (1,1,0), ARIMA (1,1,1), ARIMA (3,1,0) and ARIMA (3,1,1) models. The best performing model (in terms of the median of mean absolute percentage error over all Bunds) in this case the **ARIMA 1,1,0** model has been the most successful model.

model	mse (mean)	mse (median)	mape (mean)	mape (median)	failed
arima (3,1,0)	10.017	0.397	0.552	0.410	0
arima (3,1,1)	10.023	0.399	0.554	0.413	12
arima (1,1,0)	9.772	0.400	0.539	0.408	0
arima (1,1,1)	9.780	0.400	0.541	0.408	32

Table 45: Next week price forecast results of arima models. Ordered by mse (median) and mape (median).

Next Week Return Forecast The following table (46) reports next week return forecast results of ARIMA models. Ordered by mse (median) and mape (median).

model	mse (mean)	mse (median)	mape (mean)	mape (median)	failed
arima (1,0,1)	0.00419933	0.00389447	3.4772	3.2802	20
arima (1,0,0)	0.00419586	0.00389448	3.4737	3.2802	0
arima (3,0,0)	0.00419492	0.00389459	3.4732	3.2819	0
arima (3,0,1)	0.00420215	0.00394997	3.4810	3.3027	8

Table 46: Next week return forecast results of arima models. Ordered by mse (median) and mape (median).

10.2.3 Comparison of Performance of MLP Models, Trained on All Bunds

In this part of this work, results of different MLP models, which were trained on *all* Bunds, will be reported. First, results for next week price forecasts will be listed, second results for next week return forecasts will be reported. For both of the two different targets, results will be reported for forecasts on the original Bund data set as well as for forecasts on the fused data set, including economic features.

As described in detail in the prior section of experimental setup (5.3.5), MLP architectures range from one to three hidden layers containing 10, 20, 25 or 30 units each. Reporting all 84 architectures would require unnecessary attention of the reader, which is why it has been decided to only report the top 10 performing models. Performance is here ranked again by the median of mean squared errors over all the tested Bunds. Other performance metrics as introduced in the experimental

setup (5.7) are reported as well.

Next Week Price Forecast The following table (47) reports the ten best performing next week price forecast results of MLP models trained on **all** Bunds, ordered by mape (median). Best performance is achieved by the MLP (10,10,10).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (10,10,10)	17.555	0.527	0.833	0.516
mlp (10,20,25)	12.403	0.543	0.779	0.565
mlp (20,10,20)	6.656	0.569	0.835	0.556
mlp (25,25,20)	7.322	0.602	0.836	0.557
mlp (10,30,10)	9.293	0.612	0.828	0.634
mlp (10,20,20)	11.345	0.617	1.002	0.598
mlp (30,30,30)	12.148	0.639	0.887	0.621
mlp (25,20,10)	10.668	0.661	0.830	0.562
mlp (25,30,30)	13.513	0.679	0.922	0.622
mlp (25,20)	10.779	0.690	0.909	0.620

Table 47: Ten best performing next week price forecast results of MLP models trained on **all** Bunds, ordered by mape (median). MLP (x) stands for a MLP with one hidden layer and x units. MLP (x,y) stands for a MLP with two hidden layers and x units in the first hidden layer and y in the second.

Different Random Initializations for Best Next Week Price Forecast Model

Following table (48) lists five different initializations of the best performing next week price forecast MLP model, trained on **all** Bunds, ordered by mape (median).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (10,10,10), seed=1234	17.5551	0.5273	0.8330	0.5156
mlp (10,10,10), seed=5678	7.4704	0.5288	0.8194	0.5382
mlp (10,10,10), seed=2222	13.7014	0.5501	0.8521	0.5372
mlp (10,10,10), seed=2345	3.1637	0.6455	0.8996	0.5900
mlp (10,10,10), seed=9876	12.8707	0.6689	0.9367	0.6146

Table 48: Five different initializations of best performing next week price forecast MLP model, trained on **all** Bunds, ordered by mape (median).

Next Week Price Forecast on Fused Data Below (49), the ten best performing next week price forecast results of MLP models trained on **all** Bunds and fused data are reported. They are ordered by mape (median). Best performing model is the MLP (20,10).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (20,10), four ind.	2.271	0.747	0.799	0.643
mlp (20,10,25), seven ind.	2.322	0.898	0.846	0.66
mlp (20,25) four ind.	4.508	1.005	0.931	0.678
mlp (30,25,10), four ind.	5.581	1.042	1.156	0.713
mlp (30,20), seven ind.	12.065	0.761	1.664	0.72
mlp (30), four ind.	2.576	0.851	0.868	0.72
mlp (20,10), seven ind.	6.401	1.219	1.345	0.739
mlp (10,20,20), four ind.	4.914	1.8	1.164	0.752
mlp (30,20,10), seven ind.	4.155	1.19	1.109	0.753
mlp (30,30), seven ind.	3.622	1.047	1.037	0.78

Table 49: Ten best performing next week price forecast results MLP models trained on **all** Bunds and fused data, ordered by mape (median). MLP (x) stands for a MLP with one hidden layer and x units. MLP (x,y) stands for a MLP with two hidden layers and x units in the first hidden layer and y in the second.

Different Random Initializations for Best Next Week Price Forecast Model, Trained on Fused Data In the following table (50), five different initializations of best performing next week price forecast of MLP models are listed. They are again trained on **all** Bunds and fused data.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (20,10), four ind., seed=2345	1.8291	0.5932	0.6999	0.5719
mlp (20,10), four ind., seed=1234	2.2706	0.7469	0.7986	0.6427
mlp (20,10), four ind., seed=5555	3.0339	1.1029	0.9739	0.7571
mlp (20,10), four ind., seed=3333	3.4532	1.2058	1.0456	0.7700
mlp (20,10), four ind., seed=6789	3.7495	1.5254	1.0742	0.9137

Table 50: Five different initializations of best performing next week price forecast MLP model, trained on **all** Bunds and fused data.

Next Week Return Forecast The table below (51) shows the ten best performing next week return forecast results of MLP models trained on **all** Bunds, ordered by mse (median). Best performing in this scenario is the MLP (30,20,10).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (30,20,10)	0.00703726	0.00442275	4.70	3.63
mlp (20,10,30)	0.00742564	0.00465743	4.67	3.83
mlp (25,10,10)	0.00790437	0.00472544	4.88	3.68
mlp (10,10,30)	0.00647998	0.00484183	4.75	3.76
mlp (30,20,20)	0.01062210	0.00510542	5.67	4.36
mlp (10,20,10)	0.00573115	0.00512370	4.53	4.39
mlp (20,10,25)	0.00698273	0.00514286	5.21	4.21
mlp (25,30,10)	0.00938351	0.00514931	5.64	4.47
mlp (25,25)	0.00933842	0.00565894	6.06	4.54
mlp (20,25,25)	0.01012944	0.00569645	6.59	4.66

Table 51: Ten best performing next week return forecast results MLP models trained on **all** Bunds, ordered by mape (median). MLP (x) stands for a MLP with one hidden layer and x units. MLP (x,y) stands for a MLP with two hidden layers and x units in the first hidden layer and y in the second.

Different Random Initializations for Best Next Week Return Forecast Model Next (52), five different initializations of best performing next week return forecast of MLP models, trained on **all** Bunds, are shown.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (30,20,10), seed=1234	0.00704	0.00442	4.70801	3.63684
mlp (30,20,10), seed=8888	0.00878	0.00473	5.24696	3.94292
mlp (30,20,10), seed=5432	0.00958	0.00528	6.08744	4.34309
mlp (30,20,10), seed=9999	0.03330	0.00590	9.72051	5.12001
mlp (30,20,10), seed=2222	0.00914	0.00606	5.64456	4.60663

Table 52: Five different initializations of best performing next week return forecast MLP models, trained on **all** Bunds.

Next Week Return Forecast on Fused Data The following table (53) lists the ten best performing next week return forecast results of MLP models trained on **all** Bunds and fused data, ordered by mse (median). Best performance is achieved by the MLP (20,10), trained on the fused data set with four additional features.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (20,10), four ind.	0.0048	0.0044	4.1	3.6
mlp (25,20,10), four ind.	0.0141	0.0063	8.0	5.4
mlp (10,20), seven ind.	0.0393	0.0067	9.0	5.1
mlp (20,10,10), seven ind.	0.0091	0.0093	7.3	7.5
mlp (20,30,25), seven ind.	0.2736	0.0110	25.3	9.2
mlp (20,10,30), seven ind.	0.0481	0.0114	14.2	9.0
mlp (30,30,10), seven ind.	0.0148	0.0133	9.1	9.3
mlp (20,20,20), seven ind.	0.5098	0.0159	39.7	12.2
mlp (20,20,25), four ind.	0.0434	0.0165	14.7	10.7
mlp (25,10,25), four ind.	0.1466	0.0178	21.8	11.1

Table 53: Ten best performing next week return forecast results MLP models trained on **all** Bunds and fused data, ordered by mape (median). MLP (x) stands for a MLP with one hidden layer and x units. MLP (x,y) stands for a MLP with two hidden layers and x units in the first hidden layer and y in the second.

Different Random Initializations for Best Next Week Return Forecast Model, Trained on Fused Data Below (54), five different initializations of best performing next week return forecast of MLP models, trained on **all** Bunds and fused data, are reported.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (20,10), four ind., seed=1234	0.00480	0.00443	4.08290	3.59340
mlp (20,10), four ind., seed=5678	0.18641	0.02604	24.60491	13.48816
mlp (20,10), four ind., seed=4444	0.42957	0.02713	37.28984	17.19765
mlp (20,10), four ind., seed=6789	0.19973	0.02966	27.13669	13.69185
mlp (20,10), four ind., seed=5432	1.60648	0.03803	61.25495	17.83457

Table 54: Five different initializations of best performing next week return forecast MLP model, trained on **all** Bunds and fused data.

10.2.4 Comparison of Performance of MLP Models, Trained on Each Bund

In this part of this work, results of different MLP models, which were trained on *each* Bund, will be reported. First, results for next week price forecasts will be listed, second results for next week return forecasts will be reported. For both of the two different targets, results will be reported for forecasts on the original Bund data set as well as for forecasts on the fused data set, including economic features.

As described in detail in the prior section of experimental setup (5.3.5), MLP architectures range from one to three hidden layers containing 10, 20, 25 or 30 units each. Reporting all 84 architectures would require unnecessary attention of the reader, which is why it has been decided to only report the top 10 performing models. Performance is here ranked again by the median of mean squared errors over all the tested Bunds. Other performance metrics as introduced in the experimental setup (5.7) are reported as well.

Next Week Price Forecast The following table (55) lists the ten best performing next week price forecast results of MLP models trained on **each** Bund, ordered by mse (median).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (25,10)	0.8282	0.1996	1.4077	1.2704
mlp (30)	0.815	0.2004	1.3962	1.26
mlp (20)	0.7766	0.2108	1.3925	1.238
mlp (10,25)	0.7921	0.2123	1.398	1.2392
mlp (20,10)	0.8093	0.2252	1.3957	1.2307
mlp (25,10,10)	0.878	0.2264	1.3983	1.2744
mlp (20,30)	0.8783	0.2308	1.4048	1.2579
mlp (10)	0.8722	0.2323	1.4146	1.2809
mlp (10,20)	0.832	0.2471	1.3986	1.2562
mlp (20,20)	0.9133	0.2474	1.4143	1.2538

Table 55: Ten best performing next week price forecast results of MLP models trained on **each** Bund, ordered by mape (median). MLP (x) stands for a MLP with one hidden layer and x units. MLP (x,y) stands for a MLP with two hidden layers and x units in the first hidden layer and y in the second.

Different Random Initializations for Best Next Week Price Forecast Model, Trained on Each Bund Below (56), five different initializations of best performing next week price forecast of MLP models, trained on **each** Bund, are reported.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (25,10), seed=2345	0.7997	0.1981	1.404	1.2508
mlp (25,10), seed=1234	0.8282	0.1996	1.4077	1.2704
mlp (25,10), seed=5678	0.7888	0.2096	1.3918	1.2322
mlp (25,10), seed=4321	0.8492	0.2228	1.4092	1.2559
mlp (25,10), seed=4444	0.8228	0.2246	1.4019	1.2382

Table 56: Five different initializations of best performing next week price forecast MLP model, trained on **each** Bund.

Next Week Price Forecast on Fused Data The next table (57) reports the ten best performing next week price forecast results of MLP models trained on **each** Bund and fused data, ordered by mse (median). In this scenario, the MLP (20,20), trained on the fused data set with seven additional features, performed best.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (20,20), seven ind.	1.2069	0.3734	1.4064	1.2315
mlp (25,20), four ind.	1.1510	0.3808	1.4259	1.3048
mlp (20,10), seven ind.	1.2229	0.3821	1.3962	1.2415
mlp (25,20,10), four ind.	1.1242	0.3822	1.4181	1.2828
mlp (10,10,20), seven ind.	2.1403	0.4090	1.4805	1.2305
mlp (25,25,20), seven ind.	1.3561	0.4312	1.4483	1.2918
mlp (10,10), seven ind.	1.1540	0.4382	1.4118	1.2459
mlp (20), seven ind.	1.1628	0.4400	1.3826	1.2206
mlp (10,25,30), seven ind.	1.5715	0.4423	1.4697	1.3810
mlp (25,25), four ind.	1.4077	0.4423	1.4488	1.3361

Table 57: Ten best performing next week price forecast results MLP models trained on **each** Bund and fused data, ordered by mape (median). MLP (x) stands for a MLP with one hidden layer and x units. MLP (x,y) stands for a MLP with two hidden layers and x units in the first hidden layer and y in the second.

Different Random Initializations for Best Next Week Price Forecast Model, Trained on Each Bund and Fused Data Below (58), five different initializations of best performing next week price forecast MLP models, trained on **each** Bund and fused data, are listed.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (20,20), seven ind., seed=4321	0.787	0.2009	1.3835	1.2312
mlp (20,20), seven ind., seed=4444	0.7951	0.2021	1.3978	1.2489
mlp (20,20), seven ind., seed=6666	0.8037	0.2134	1.3868	1.2195
mlp (20,20), seven ind., seed=7654	0.7838	0.223	1.386	1.224
mlp (20,20), seven ind., seed=5678	0.8717	0.229	1.3967	1.2534

Table 58: Five different initializations of best performing next week price forecast MLP model, trained on **each** Bund and fused data.

Next Week Return Forecast The table below (59) lists the ten best performing next week return forecast results of MLP models trained on **each** Bund, ordered by mape (median). Best performance is achieved by the MLP (10,20,10).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (10,20,10)	0.00439	0.00417	3.75994	3.53382
mlp (25,25,25)	0.00590	0.00424	4.54022	3.62975
mlp (25,10,10)	0.00444	0.00426	3.76122	3.54142
mlp (20,25,25)	0.00580	0.00432	4.61424	3.72162
mlp (30,10,10)	0.00464	0.00433	3.91071	3.75160
mlp (10,20,20)	0.00493	0.00433	4.13358	3.82689
mlp (20,10,30)	0.00509	0.00436	4.24335	3.70199
mlp (25,25,20)	0.00641	0.00437	4.82253	3.78202
mlp (25,10,25)	0.00537	0.00437	4.36444	3.69489
mlp (25,10,30)	0.00499	0.00438	4.19517	3.76597

Table 59: Ten best performing next week return forecast results MLP models trained on **each** Bund, ordered by mape (median). MLP (x) stands for a MLP with one hidden layer and x units. MLP (x,y) stands for a MLP with two hidden layers and x units in the first hidden layer and y in the second.

Different Random Initializations for Best Next Week Return Forecast Model, Trained on Each Bund Below (60), five different initializations of best performing next week return forecast MLP model, trained on **each** Bund, are listed.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (10,20,10), seed=4321	0.00449	0.00417	3.84822	3.66612
mlp (10,20,10), seed=1234	0.00439	0.00417	3.75994	3.53382
mlp (10,20,10), seed=5678	0.00481	0.00427	4.06681	3.76117
mlp (10,20,10), seed=2345	0.00507	0.00429	4.09095	3.58847
mlp (10,20,10), seed=8765	0.00463	0.00431	3.98316	3.6404

Table 60: Five different initializations of best performing next week return forecast MLP model, trained on **each** Bund.

Next Week Return Forecast on Fused Data In the following (61), ten best performing next week return forecast results of MLP models trained on **each** Bund and fused data, ordered by mape (median). Here, the MLP (25,25,20) outperforms other configurations.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (25,25,20), four ind.	0.00501	0.00413	4.14350	3.52428
mlp (10,30,10), seven ind.	0.00437	0.00420	3.83148	3.63571
mlp (30,20,20), four ind.	0.00505	0.00425	4.34088	3.68780
mlp (10,30,10), four ind.	0.00559	0.00426	4.41108	3.63522
mlp (25), four ind.	0.00489	0.00427	4.15359	3.65914
mlp (25,20,10), four ind.	0.00490	0.00430	4.09301	3.51057
mlp (30,10,30), four ind.	0.00646	0.00430	4.98482	3.82334
mlp (10,10,30), seven ind.	0.00479	0.00431	4.07299	3.58836
mlp (25,25), seven ind.	0.00477	0.00431	4.12372	3.84349
mlp (30,10), four ind.	0.00653	0.00435	4.91454	3.78474

Table 61: Ten best performing next week return forecast results MLP models trained on **each** Bund and fused data, ordered by mape (median). MLP (x) stands for a MLP with one hidden layer and x units. MLP (x,y) stands for a MLP with two hidden layers and x units in the first hidden layer and y in the second.

Different Random Initializations for Best Next Week Return Forecast Model, Trained on Each Bund and Fused Data The following table (62), reports five different initializations of the best performing next week return forecast MLP model, trained on **each** Bund and fused data.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
mlp (25,25,20), four ind., seed=7777	0.00509	0.00418	4.19271	3.56795
mlp (25,25,20), four ind., seed=3456	0.00512	0.00426	4.3252	3.67348
mlp (25,25,20), four ind., seed=2345	0.00453	0.00433	3.85467	3.61208
mlp (25,25,20), four ind., seed=4444	0.00596	0.00451	4.80887	3.91811
mlp (25,25,20), four ind., seed=5678	0.00662	0.00458	5.01262	3.78129

Table 62: Five different initializations of best performing next week return forecast MLP model, trained on **each** Bund and fused data.

10.2.5 Comparison of Performance of LSTM Models, Trained on All Bunds

Next Week Price Forecast Following table (63) lists the ten best performing next week price forecast results of LSTM models trained on **all** Bunds, ordered by mape (median). Best performance is achieved by the LSTM (25,10) in this case.

model	mse (mean)	mse (median)	mape (mean)	mape (median)
lstm (10), 3 days	1.5295	0.3682	0.5868	0.3847
lstm (25,10), 3 days	1.2879	0.3285	0.5874	0.3996
lstm (20,20), 3 days	2.0446	0.3477	0.6439	0.4223
lstm (10,30), 3 days	1.5407	0.3544	0.5449	0.4263
lstm (25,30), 3 days	1.6357	0.3435	0.5828	0.4343
lstm (25,25), 3 days	1.6242	0.3494	0.6143	0.4360
lstm (25,20), 3 days	1.3365	0.3754	0.5477	0.4418
lstm (30), 3 days	1.1821	0.4466	0.5486	0.4512
lstm (20,10,20), 3 days	1.3835	0.3660	0.5935	0.4627
lstm (20,30), 3 days	1.5607	0.4393	0.6267	0.4718

Table 63: Ten best performing next week price forecast results of LSTM models trained on **all** Bunds, ordered by mape (median). LSTM (x) stands for a LSTM with one hidden layer and x units. LSTM (x,y) stands for a LSTM with two hidden layers and x units in the first hidden layer and y in the second.

Next Week Price Forecast on Fused Data Below (64), the ten best performing next week price forecast results of LSTM models trained on **all** Bunds and fused data, ordered by mape (median).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
lstm (30,25,10), 3 days, four ind.	1.382	0.230	0.538	0.338
lstm (10,20,20), 10 days, four ind.	1.973	0.238	0.623	0.342
lstm (20,30), 3 days, four ind.	1.864	0.251	0.603	0.344
lstm (10,10,30), 3 days, four ind.	3.129	0.258	0.767	0.356
lstm (30,25), 3 days, four ind.	1.461	0.275	0.557	0.358
lstm (30,30), 10 days, four ind.	1.385	0.252	0.570	0.359
lstm (25,30), 3 days, four ind.	2.159	0.269	0.661	0.361
lstm (20,30), 10 days, four ind.	1.315	0.254	0.562	0.362
lstm (10,10), 3 days, four ind.	1.479	0.262	0.548	0.365
lstm (30,20), 3 days, four ind.	1.660	0.278	0.627	0.369

Table 64: Ten best performing next week price forecast results of LSTM models trained on **all** Bunds and fused data, ordered by mape (median). LSTM (x) stands for a LSTM with one hidden layer and x units. LSTM (x,y) stands for a LSTM with two hidden layers and x units in the first hidden layer and y in the second.

Next Week Return Forecast The next table (65) reports the ten best performing next week return forecast results of LSTM models trained on **all** Bunds, ordered by mape (median).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
lstm (10,10,25), 3 days	0.004	0.004	3.472	3.290
lstm (10,25), 3 days	0.004	0.004	3.487	3.292
lstm (25,25,10), 10 days	0.004	0.004	3.487	3.294
lstm (10,10,25), 10 days	0.004	0.004	3.474	3.295
lstm (30,20,25), 10 days	0.004	0.004	3.473	3.296
lstm (25,30,10), 3 days	0.004	0.004	3.468	3.297
lstm (20,25,10), 3 days	0.004	0.004	3.469	3.297
lstm (25,10,20), 10 days	0.004	0.004	3.469	3.298
lstm (10,25,30), 3 days	0.004	0.004	3.487	3.298
lstm (10,25,10), 3 days	0.004	0.004	3.470	3.298

Table 65: Ten best performing next week return forecast results of LSTM models trained on **all** Bunds, ordered by mape (median). LSTM (x) stands for a LSTM with one hidden layer and x units. LSTM (x,y) stands for a LSTM with two hidden layers and x units in the first hidden layer and y in the second.

Next Week Return Forecast on Fused Data The table below (66) lists the ten best performing next week return forecast results of LSTM models trained on **all** Bunds and fused data, ordered by mape (median).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
lstm (20,25,20), 3 days, four ind.	0.0039	0.0040	3.2935	3.2788
lstm (20,20,25), 10 days, four ind.	0.0039	0.0040	3.3358	3.2789
lstm (30,20,10), 10 days, four ind.	0.0039	0.0040	3.3020	3.2790
lstm (25,25,20), 10 days, four ind.	0.0039	0.0040	3.2960	3.2790
lstm (10,25), 3 days, four ind.	0.0039	0.0040	3.3101	3.2790
lstm (30,30,10), 10 days, four ind.	0.0039	0.0040	3.2912	3.2790
lstm (10,25,10), 3 days, four ind.	0.0039	0.0040	3.2928	3.2791
lstm (20,25,10), 10 days, four ind.	0.0039	0.0040	3.2980	3.2791
lstm (30,25,10), 3 days, four ind.	0.0039	0.0040	3.2928	3.2791
lstm (30,20,30), 10 days, four ind.	0.0039	0.0040	3.2957	3.2792

Table 66: Ten best performing next week return forecast results of LSTM models trained on **all** Bunds and fused data, ordered by mape (median). LSTM (x) stands for a LSTM with one hidden layer and x units. LSTM (x,y) stands for a LSTM with two hidden layers and x units in the first hidden layer and y in the second.

10.2.6 Comparison of Performance of LSTM Models, Trained on Each Bunds

Next Week Price Forecast Following table (67) lists the ten best performing next week price forecast results of LSTM models trained on **each** Bunds, ordered by mape (median).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
lstm (30,10), 3 days	5.18	4.06	1.50	1.06
lstm (30,25), 3 days	90.75	4.40	4.17	1.19
lstm (10,25,20), 3 days	7.29	4.85	1.79	1.25
lstm (20,20,25), 3 days	130.41	4.65	5.20	1.31
lstm (25,10), 3 days	62.27	4.90	4.08	1.31
lstm (20,10,25), 3 days	11.24	5.80	2.28	1.32
lstm (20,30,20), 3 days	104.15	5.58	4.96	1.33
lstm (10,20,30), 3 days	48.30	5.51	3.83	1.37
lstm (10,25,10), 3 days	96.70	6.30	5.17	1.39
lstm (20,30,25), 3 days	12.39	5.30	2.26	1.39

Table 67: Ten best performing next week price forecast results of LSTM models trained on **each** Bunds, ordered by mape (median). LSTM (x) stands for a LSTM with one hidden layer and x units. LSTM (x,y) stands for a LSTM with two hidden layers and x units in the first hidden layer and y in the second.

Next Week Price Forecast on Fused Data Below (68), the ten best performing next week price forecast results of LSTM models trained on **each** Bunds and fused data, ordered by mape (median).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
lstm (10,25,10), 3 days, four ind.	22.33	5.45	2.84	1.57
lstm (10,30,10), 3 days, four ind.	46.52	6.27	3.55	1.70
lstm (10,20), 3 days, four ind.	120.50	5.43	5.18	1.90
lstm (20,30,30), 10 days, four ind.	11.13	9.05	2.39	2.12
lstm (25), 3 days, four ind.	37.50	6.22	3.73	2.13
lstm (10,20,10), 3 days, four ind.	36.65	7.49	3.38	2.13
lstm (30,10), 3 days, four ind.	50.98	11.85	4.58	2.16
lstm (30,20,10), 3 days, four ind.	43.77	10.40	3.54	2.18
lstm (25,10), 3 days, four ind.	99.02	7.03	4.88	2.22
lstm (20,10,10), 10 days, four ind.	230.13	8.37	6.24	2.22

Table 68: Ten best performing next week price forecast results of LSTM models trained on **each** Bunds and fused data, ordered by mape (median). LSTM (x) stands for a LSTM with one hidden layer and x units. LSTM (x,y) stands for a LSTM with two hidden layers and x units in the first hidden layer and y in the second.

Next Week Return Forecast The next table (69) reports the ten best performing next week return forecast results of LSTM models trained on **each** Bund, ordered by mape (median).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
lstm (10,20,30), 3 days	0.0047	0.0044	4.2454	3.5708
lstm (20,10,30), 3 days	0.0046	0.0044	4.1620	3.5765
lstm (25,10,30), 3 days	0.0046	0.0043	4.1683	3.5805
lstm (10,10,30), 3 days	0.0048	0.0045	4.3522	3.6107
lstm (20,10,20), 3 days	0.0048	0.0043	4.3215	3.6898
lstm (10,25,30), 3 days	0.0048	0.0044	4.3117	3.6960
lstm (10,10,30), 10 days	0.0048	0.0043	4.2209	3.7233
lstm (20,20,30), 3 days	0.0048	0.0044	4.3010	3.7268
lstm (20,10,25), 3 days	0.0049	0.0045	4.4159	3.7436
lstm (25,20,30), 3 days	0.0052	0.0047	4.6244	3.7480

Table 69: Ten best performing next week return forecast results of LSTM models trained on **each** Bund, ordered by mape (median). LSTM (x) stands for a LSTM with one hidden layer and x units. LSTM (x,y) stands for a LSTM with two hidden layers and x units in the first hidden layer and y in the second.

Next Week Return Forecast on Fused Data The table below (70) lists the ten best performing next week return forecast results of LSTM models trained on **each** Bund and fused data, ordered by mape (median).

model	mse (mean)	mse (median)	mape (mean)	mape (median)
lstm (30,10,30), 10 days, four ind.	0.013	0.004	5.483	3.496
lstm (20,30,30), 3 days, four ind.	0.013	0.004	5.604	3.531
lstm (10,25,25), 3 days, four ind.	0.009	0.004	4.857	3.546
lstm (10,25,30), 3 days, four ind.	0.007	0.004	4.554	3.587
lstm (30,10,30), 3 days, four ind.	0.013	0.004	5.479	3.598
lstm (20,10,30), 10 days, seven ind.	0.010	0.004	5.040	3.601
lstm (30,10,20), 10 days, seven ind.	0.010	0.004	5.069	3.608
lstm (20,20,30), 3 days, four ind.	0.013	0.004	5.533	3.612
lstm (20,20,30), 10 days, seven ind.	0.010	0.004	5.120	3.614
lstm (20,10,20), 3 days, four ind.	0.006	0.004	4.180	3.614

Table 70: Ten best performing next week return forecast results of LSTM models trained on **each** Bund and fused data, ordered by mape (median). LSTM (x) stands for a LSTM with one hidden layer and x units. LSTM (x,y) stands for a LSTM with two hidden layers and x units in the first hidden layer and y in the second.

10.3 Development of R-Shiny-App

10.3.1 Motivation

The development of the R-Shiny-App was motivated by the difficulty, when trying to quickly evaluate results of a certain forecast for a certain model. Since model evaluation always needs to consider forecast for different Bunds, a comparison of forecasting results for single Bunds between different models was missing. This gap is closed by the R-Shiny-App, which by user selection, depicts certain Bund performances for a certain task for a user selection of models. This helps the user to quickly identify, what the user is searching for.

10.3.2 Components

The developed Shiny App consists of the following major components.

Start Page When opening the Shiny app, the user will be provided with this view (51) of the start page. It displays the next day forecast (out of sample), the graph of the next 50 days of the actual price development and many options to manipulate the graph which can be found on the left. Each of the single components of the app will be addressed next.

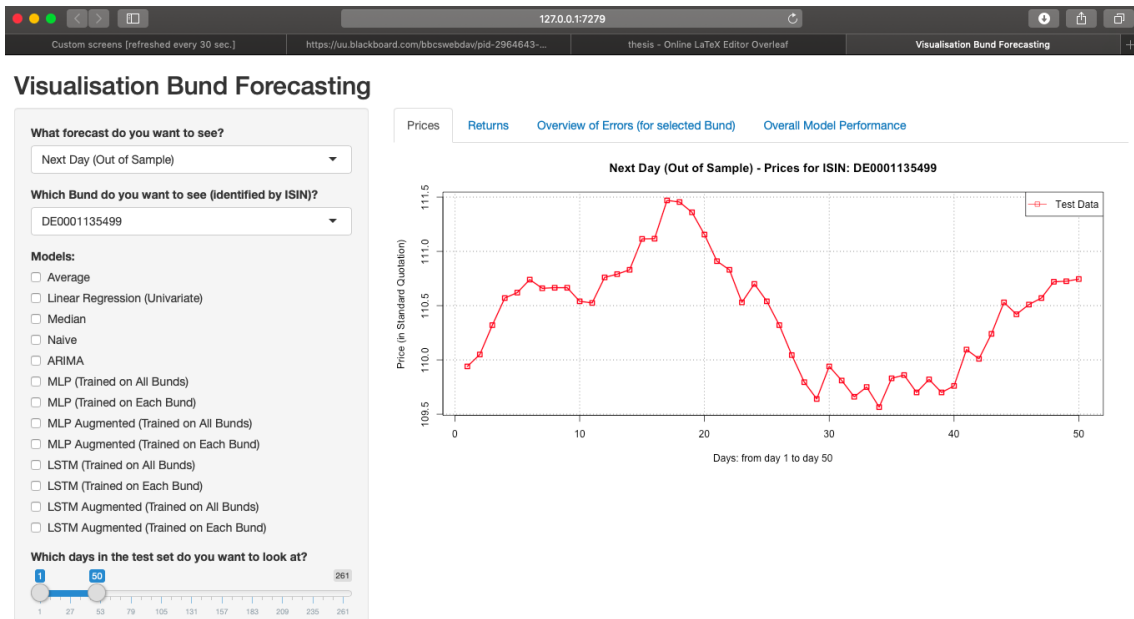


Figure 51: Start Page Shiny App

Main Navigation Bar on the Left The following figure (52) depicts the major part of the navigation bar on the left of the Shiny app. It can be used for various purposes, like choosing the Bund, the model, exports et cetera. Detailed information on each component can be found below.

The screenshot shows a Shiny App navigation bar with the following elements:

- What forecast do you want to see?**: A dropdown menu with "Next Day (Out of Sample)" selected.
- Which Bund do you want to see (identified by ISIN)?**: A dropdown menu with "DE0001135499" selected.
- Models:**: A list of radio button options:
 - Average
 - Linear Regression (Univariate)
 - Median
 - Naive
 - ARIMA
 - MLP (Trained on All Bunds)
 - MLP (Trained on Each Bund)
 - MLP Augmented (Trained on All Bunds)
 - MLP Augmented (Trained on Each Bund)
 - LSTM (Trained on All Bunds)
 - LSTM (Trained on Each Bund)
 - LSTM Augmented (Trained on All Bunds)
 - LSTM Augmented (Trained on Each Bund)
- Which days in the test set do you want to look at?**: A slider range from 1 to 261. The current range is from 1 to 50, with a blue bar and a slider knob at 50.

Figure 52: Shiny App Navigation Bar

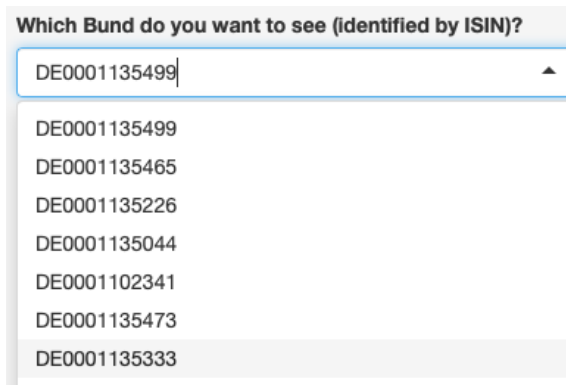
Choosing the depicted forecasting horizon This figure (53), displaying a dropdown menu, can be used to select the forecast horizon which is considered for graphs and model performance for the selected models. The user can choose between next day and next week as well as in sample and out of sample horizons.

The screenshot shows a Shiny App dropdown menu with the following options:

- What forecast do you want to see?**: A dropdown menu with "Next Day (Out of Sample)" selected.
- Next Day (Out of Sample)
- Next Day (In Sample)
- Next Week (Out of Sample)
- Next Week (In Sample)

Figure 53: Shiny App Choosing Forecast Horizon

Choosing the Bund to display This figure (54), displaying a dropdown menu, can be used to select the Bund which is considered for graphs and model performance for the selected models.



Which Bund do you want to see (identified by ISIN)?

DE0001135499 ▲

DE0001135499

DE0001135465

DE0001135226

DE0001135044

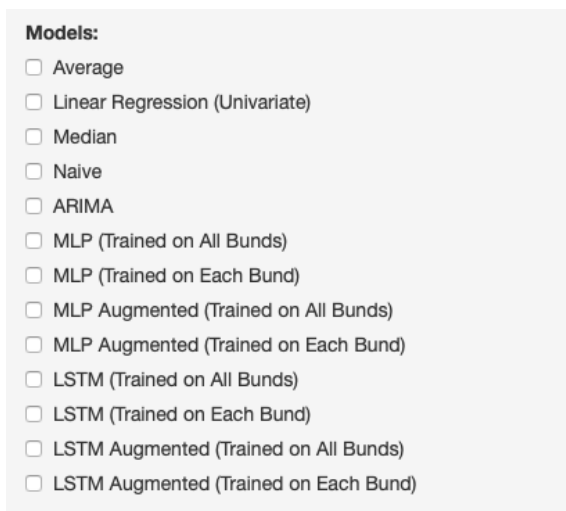
DE0001102341

DE0001135473

DE0001135333

Figure 54: Shiny App Choosing Bund

Choosing the Model(s) to display This figure (55), displaying a multi-check-box, can be used to select the models which are considered for graphs and model performance for the selected Bund.



Models:

Average

Linear Regression (Univariate)

Median

Naive

ARIMA

MLP (Trained on All Bunds)

MLP (Trained on Each Bund)

MLP Augmented (Trained on All Bunds)

MLP Augmented (Trained on Each Bund)

LSTM (Trained on All Bunds)

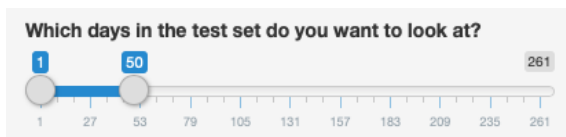
LSTM (Trained on Each Bund)

LSTM Augmented (Trained on All Bunds)

LSTM Augmented (Trained on Each Bund)

Figure 55: Shiny App Choosing Model

Choosing the number of days to display The next image (56) shows the slider which allows to specify the date range which is considered for the drawing of the graphs as well as for the calculating model performance for the specified Bund and time scope.



Which days in the test set do you want to look at?

1 50 261

1 27 53 79 105 131 157 183 209 235 261

Figure 56: Shiny App Choosing Days Range

Exporting the current graph The following figure (57) shows the export function of the R-Shiny-App. The two buttons displayed allow to either export the current, active graph for the chosen Bund or export the overall model performance over all Bunds per model.

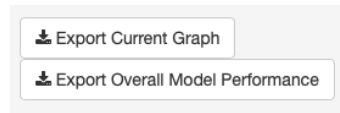


Figure 57: Shiny App Export Function

Current graph: prices The following figure (58) shows the actual return and return forecasts for each model for the chosen ISIN and the chosen time frame selected by the slider which allows to select the days for which the forecasts are depicted.

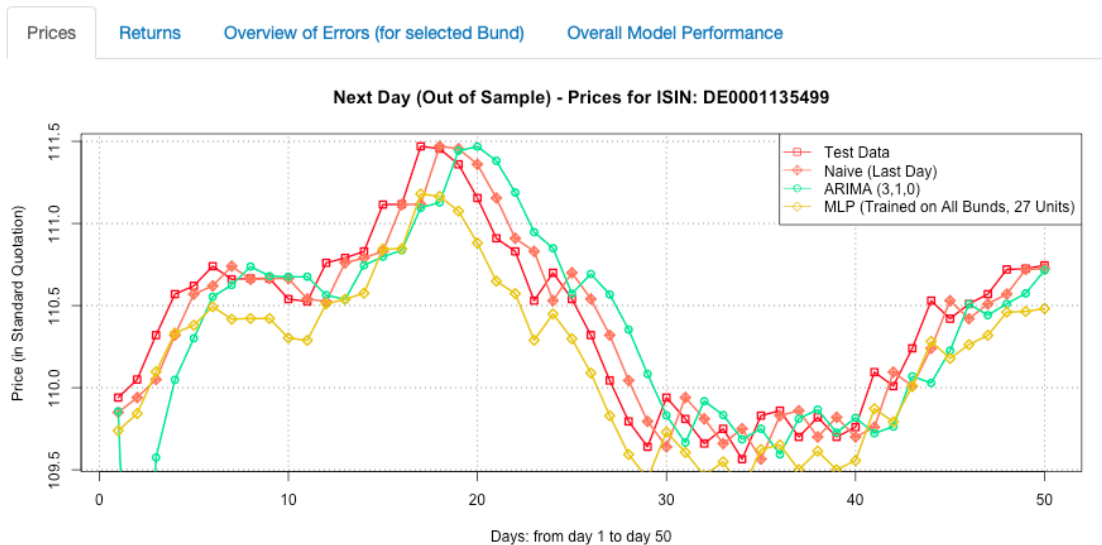


Figure 58: Shiny App Price Graph

Current graph: returns The following figure (59) shows the actual return and return forecasts for each model for the chosen ISIN and the chosen time frame selected by the slider which allows to select the days for which the forecasts are depicted.

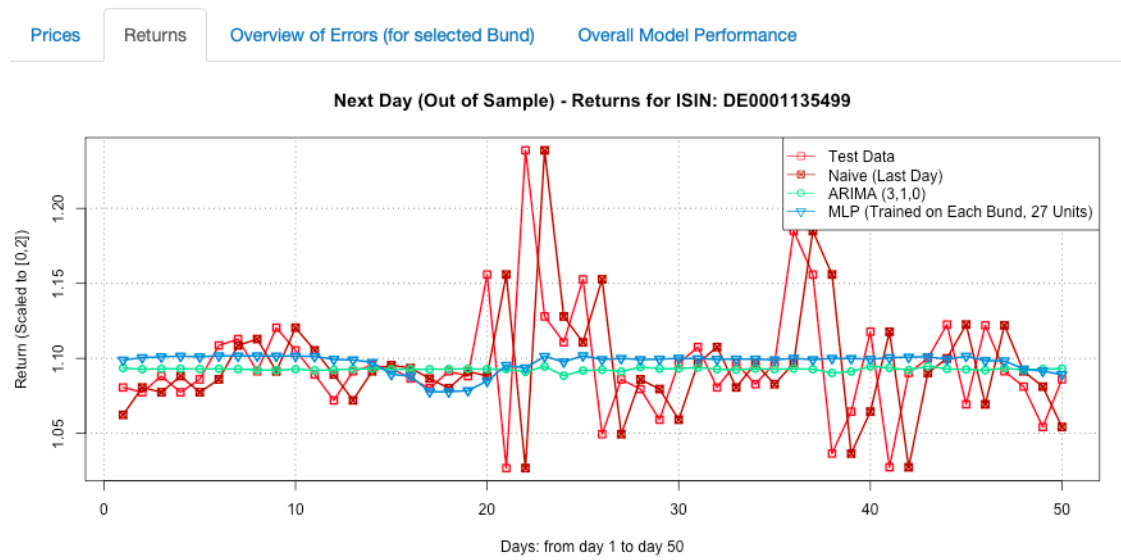


Figure 59: Shiny App Return Graph

Current performance The following table (60) reports performances for each model for the chosen ISIN and the chosen time frame selected by the slider which allows to select the days for which the forecasts are depicted.

Prices		Returns	Overview of Errors (for selected Bund)		Overall Model Performance
model	mse.prices.next.day	mape.prices.next.day	mse.returns.next.day	mape.returns.next.day	
ARIMA	0.26	0.26	0.00	2.15	
Average	0.06	0.18	0.00	2.33	
Linear Univariate (Last 10 Days)	0.04	0.14	0.00	2.92	
LSTM (Trained on All Bunds)	0.06	0.18	0.00	2.14	
LSTM (Trained on Each Bund)	7.35	2.44	0.00	2.73	
LSTM Augmented (Trained on All Bunds)	0.21	0.38	0.00	2.11	
LSTM Augmented (Trained on Each Bund)	7.35	2.44	0.00	2.73	
Median	0.06	0.19	0.00	2.59	
MLP (Trained on All Bunds)	0.14	0.30	0.00	2.81	
MLP (Trained on Each Bund)	0.06	0.21	0.00	2.27	
MLP Augmented (Trained on All Bunds)	0.66	0.72	0.00	2.12	
MLP Augmented (Trained on Each Bund)	0.08	0.25	0.00	3.46	
Naive	0.03	0.13	0.00	3.27	

Figure 60: Shiny App Current Performance

10.4 MAPE per Model for a Selection of 10 Bunds

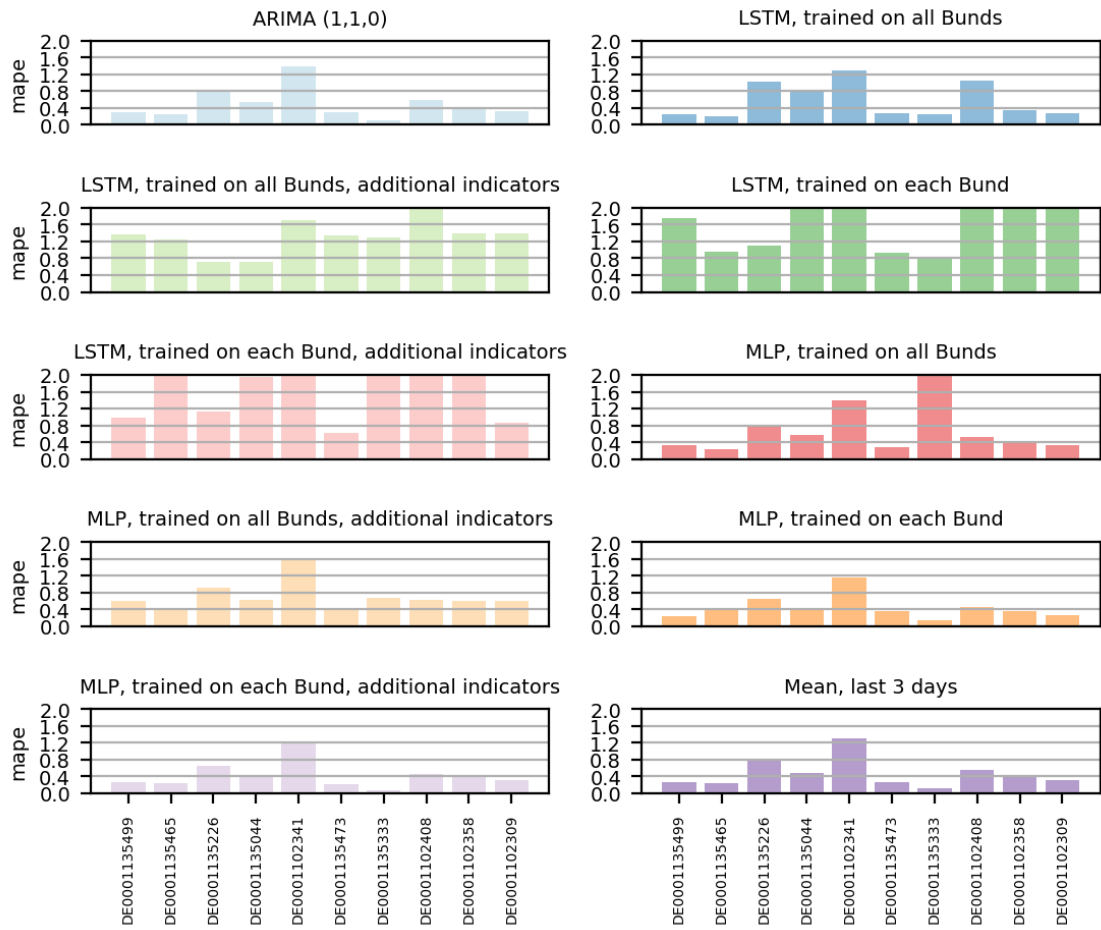


Figure 61: Mape per ISIN per Model for next week price forecasts

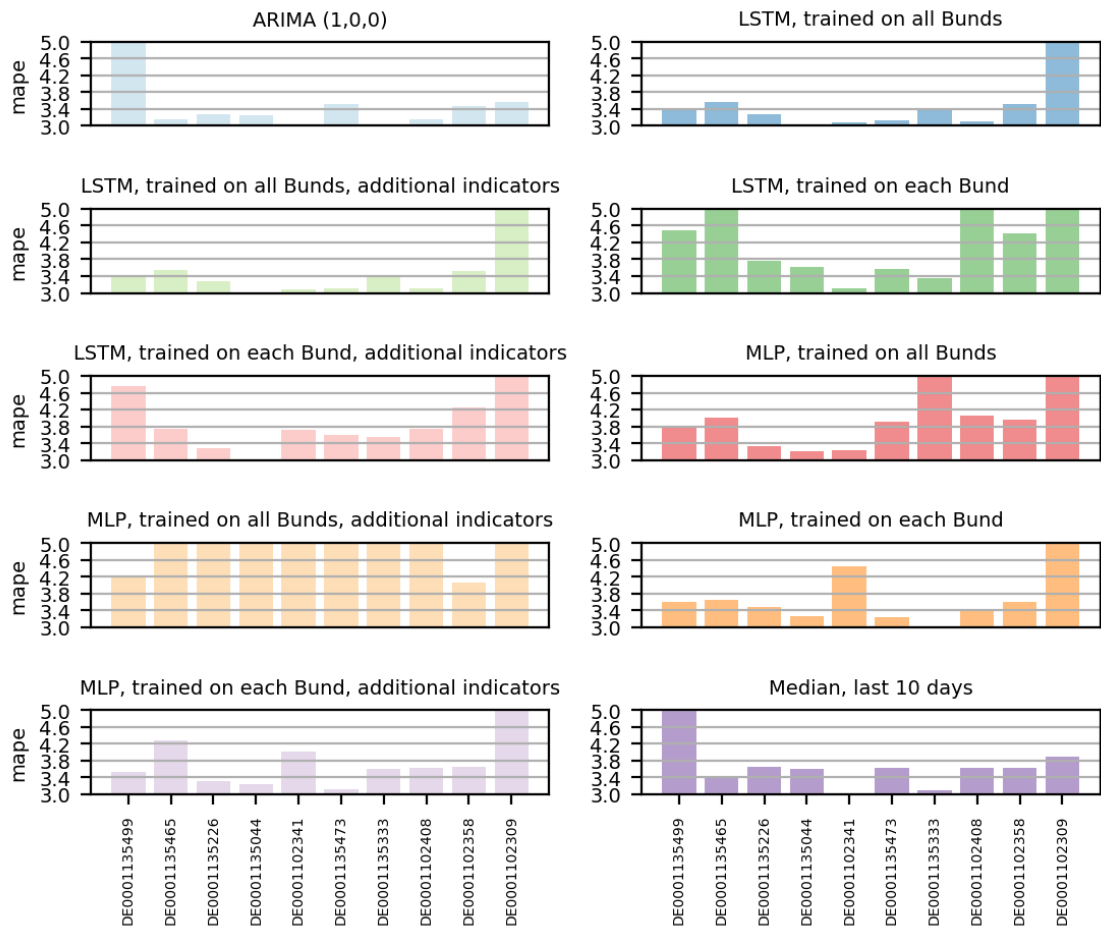


Figure 62: MAPE per ISIN per Model for next week return forecasts