Hybrid Methods for Time Series Forecasting

Master Thesis

by

Daniel Berberich

At the Department of Economics and Management

Karlsruhe Service Research Institute

Advisor:	Prof. Dr. Hansjörg Fromm
Second Advisor:	Prof. Dr. Gerhard Satzger
External Advisor:	Dr. Florian Wilhelm (inovex GmbH)
Date of Submission:	June 30, 2020

Declaration of Academic Integrity

I hereby confirm that the present thesis is solely my own work and that if any text passages or diagrams from books, papers, the Web or other sources have been copied or in any other way used, all references—including those found in electronic media—have been acknowledged and fully cited.

Karlsruhe, June 30, 2020 *Daniel Berberich*

Abstract

Time series forecasting is a crucial task in various fields of business and science. There are two coexisting approaches to time series forecasting, which are statistical methods and machine learning methods. Both come with different strengths and limitations. Statistical methods such as the Holt-Winters' Method or ARIMA have been practiced for decades. They stand out due to their robustness and flexibility. Furthermore, these methods work well when few data is available and can exploit a priori knowledge. However, statistical methods assume linear relationships in the data, which is not necessarily the case in real-world data, inhibiting forecasting performance.
On the other hand, machine learning methods such as Multilayer Perceptrons or Long Short-Term Memory Networks do not have the assumption of linearity and have the exceptional advantage of universally approximating almost any function. In addition to that, machine learning methods can exploit cross-series information to enhance an individual forecast. Besides these strengths, machine learning methods face several limitations in terms of data and computation requirements.

Hybrid methods promise to advance time series forecasting by combining the best of statistical and machine learning methods. The fundamental idea is that the combination compensates for the limitations of one approach with the strengths of the other. This thesis shows that the combination of a Holt-Winters' Method and a Long Short-Term Memory Network is promising when the periodicity of a time series can be precisely specified. The precise specification enables the Holt-Winters' Method to simplify the forecasting task for the Long Short-Term Memory Network and, consequently, facilitates the hybrid method to obtain accurate forecasts.

The research question to be answered is which characteristics of a time series determine the superiority of either statistical, machine learning, or hybrid approaches. The result of the conducted experiment shows that this research question can not be answered generally. Nevertheless, the results propose findings for specific forecasting methods. The Holt-Winters' Method provides reliable forecasts when the periodicity can be precisely determined. ARIMA, however, handles overlying seasonalities better than the Holt-Winters' Method due to its autoregressive approach. Furthermore, the results suggest the hypothesis that machine learning methods have difficulties extrapolating time series with trend. Finally, the Multilayer Perceptron can conduct accurate forecasts for various time series despite its simplicity, and the Long Short-Term Memory Network proves that it needs relevant datasets of adequate length to conduct accurate forecasts.

Contents

Acronyms

ACF	Autocorrelation Function
ADF	Augmented Dickey-Fuller Test
AI	Artificial Intelligence
ANN	Artificial Neural Network
AR	Autoregressive
ARIMA	Autoregressive Integrated Moving Average
ARIMAX	Autoregressive Integrated Moving Average with explanatory variables
ARMA	Autoregressive Moving Average
CNN	Convolutional Neural Network
DL	Deep Learning
DSR	Design Science Research
DWT	Discrete Wavelet Transform
FFNN	Feedforward Neural Network
GRU	Gated Recurrent Unit
HW	Holt-Winters' Method
IOB	Improvement over baseline
LSTM	Long Short-Term Memory Network
MA	Moving Average
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MASE	Mean Absolute Scaled Error
MIMO	Multi-input multi-output
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NN	Neural Network
NNAR	Neural Network Autoregression
OWA	Overall Weighted Average
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SARIMA	Seasonal Autoregressive Integrated Moving Average
SARIMAX	Seasonal Autoregressive Integrated Moving Average with explanatory variables
SES	Simple Exponential Smoothing
SGD	Stochastic Gradient Descent

sMAPE	Symmetric Mean Absolute Percentage Error
STL	Seasonal and Trend Decomposition using Loess
SVR	Support Vector Regression
TDNN	Time-delay Neural Network
TES	Triple Exponential Smoothing

List of Figures

List of Tables

1 Introduction

This master thesis is dedicated to time series forecasting. It elaborates on statistical, machine learning, and hybrid approaches in this domain. Each of these approaches has individual strengths and limitations, which are covered in this work. Furthermore, this thesis presents an experiment whose results aim to answer the research question.

1.1 Motivation

The literature for time series forecasting distinguishes two elemental approaches to time series forecasting: statistical methods and machine learning methods. These two approaches coexist in the domain but are fundamentally different. Some authors even describe a clash of cultures that divides the forecasting community between these two approaches (Breiman, 2001b). Each of these approaches has unique strengths that established them in the various fields of applications for time series forecasting. However, neither statistical nor machine learning methods are free from limitations that inhibit excellent forecasting performance.

In the past, well-established statistical methods showed dominance over emerging machine learning methods in forecasting competitions. While the machine learning approaches improved and delivered promising results, they still face significant challenges in forecasting competitions (Makridakis et al., 2020).

Hybrid methods, which are a combination of statistical and machine learning methods, promise to unite the best of both worlds to advance time series forecasting. The fundamental idea of hybrid methods is that the combination compensates for the limitations of one approach with the strengths of the other.
The breakthrough was Slawek Smyl's winning submission to a forecasting contest known as the M4 Competition. In his proposed hybrid method, the author combined statistical methods with machine learning elements to outperform every other submission (Smyl, 2020). With this achievement, the method caused a great stir among forecasters and proved the potential of hybrid methods for time series forecasting successfully.

The thesis is divided into eight chapters. This chapter introduces the subject and the motivation for the thesis. Furthermore, it presents the research question and applied research design. Chapter 2 presents the related work and investigates the relevant literature for this subject. As there are essential definitions and theoretical foundations that have to be introduced, Chapter 3 starts by giving an introduction to time series before presenting the fundamental concepts of time series forecasting. A selection of statistical and machine learning methods is presented in Chapter 3.2

and Chapter 3.3. As their strengths and limitations are a core interest of this paper, two respective subsections discuss them in more substantial detail.
From the background of the strengths and limitations of statistical and machine learning methods, follow hybrid methods to overcome these shortcomings. Chapter 4 introduces the essentials of these hybrid methods and gives an overview of relevant research conducted in this field. Slawek Smyl's outstanding example of a hybrid method is comprehensively discussed in Chapter 4.3.
Chapter 5 presents the experimental design that underlies this thesis's course of the investigation. For the experiment, a range of time series is identified and clustered regarding their characteristics. These time series are used to conduct forecasts with different forecasting methods from statistical, machine learning, and hybrid approaches. Their forecasts are evaluated using different accuracy measures to gain a better understanding of their performance. Furthermore, the chapter presents the experiment's implementation in Python.
The results of the experiment are used to find implications for the research question. In that endeavor, Chapter 6 describes the observations made in the experiment and elaborates on the implications. These implications are critically discussed in Chapter 7. Finally, this thesis closes with Chapter 8, which concludes the paper and outlines future research in that domain.

1.2 Research Question

The results of the M4 Competition prove that hybrid methods can be a functional and valuable tool for time series forecasting. The research question seeks to generalize the findings of the M4 Competition in terms of the characteristics of a time series and compares the hybrid methods against the two established cultures in time series forecasting. Therefore, the *research question* of this thesis is as follows:

> *What are the characteristics of a time series that determine the superiority of either statistical, machine learning, or hybrid forecasting methods?*

This formulation gives rise to a broad view of time series forecasting. To find appropriate answers to this research question, a research design is necessary. Therefore, the next section elaborates on the utilized research design for this thesis.

1.3 Research Design

The research design is based on the *Design Science Research (DSR)* described by Hevner (2007). This section presents its three cycles and their realization for this thesis.

The three cycles are the relevance cycle, rigor cycle, and design cycle (Hevner, 2007). The *relevance cycle* raises the question of whether the design of the artifact improves the environment and how this improvement can be measured. The *rigor cycle* connects the design science activities with the knowledge base and investigates the state of the art. Consequently, it seeks to answer the question if the artifact is suitable to answer the research question. Finally, the *design cycle* iterates between the core activities of building and evaluating the design artifacts.

The artifact of this thesis is the experimental design that is introduced in Chapter 5. It identifies a range of time series with distinguishing characteristics and applies appropriate forecasting methods from different approaches. These forecasts are evaluated with meaningful accuracy measures. This setting complies with the relevance cycle and allows for statements to answer the research question. The procedures and techniques applied in the experiment are backed by state-of-art theory from the knowledge base, just as required by the rigor cycle. Consequently, this experiment's design ensures that the thesis fulfills its purpose and finds an adequate answer to the research question.

2 Related Work

This chapter examines the related work and presents the existing research on the topic. Furthermore, it describes the research gap this thesis is closing.

When reviewing the literature, it becomes evident that many authors describe the difficulties in choosing the right forecasting method (Brockwell and Davis, 2016). Apart from this, the no free lunch theorem applied to time series forecasting states that no forecasting method is universally superior, and there does not exist a shortcut to the choice of the most suitable forecasting method (Wolpert and Macready, 1997).
As early as in the 1970s, Reid (1972) pointed out that the performance of a forecasting method changes according to the nature of the data. Concerning the research question, the nature of the data is reflected in the characteristics of the time series. The resulting matter is the algorithm selection problem, which was first described by Rice (1976). Since then, several researchers have introduced rules for forecasting based on these characteristics (Collopy and Armstrong, 1992; Adya et al., 2001; Wang et al., 2009).
With the emergence of machine learning methods, the algorithm selection problem was relabelled and the term meta-learning was coined by Prudêncio and Ludermir (2004). Over time, state-of-the-art expert systems were supplemented by data-driven approaches (Wang et al., 2009). These data-driven approaches describe frameworks that provide recommendations on which forecast method should be used to generate forecasts. Thereby, some rely on techniques based on neural networks (Talagala et al., 2018).

This thesis analyzes the characteristics of different time series to find implications for the superiority of either statistical, machine learning, or hybrid forecasting methods. Thereby, this thesis does not rely on a meta-learning approach but provides an explainable procedure to the assessment of forecasting methods. The identified *research gap* manifests in the fact that none of the existing literature considers hybrid methods in their analysis. Hybrid methods promise a new approach to time series forecasting and should be considered in every decision concerning algorithm selection. Therefore, this thesis will conduct an experiment that allows for the assessment of a range of time series and involves statistical, machine learning, as well as hybrid methods in the investigation.

The following chapter introduces the theoretical foundations of time series and time series forecasting. These foundations are necessary to gain a comprehensive understanding of the research question and the research gap.

3 Theoretical Foundations

This chapter intends to give an understanding of the theoretical foundations which this thesis is based upon. Starting with an introduction to time series, it elaborates on statistical, machine learning, and hybrid approaches for time series forecasting in the course of this chapter. Furthermore, it covers the topic of evaluation metrics.

3.1 Time Series

The following section gives an introduction to time series. It presents basic terminology and covers time series forecasting.

3.1.1 Terminology

A *time series* is a sequence of historical measurements y_t of an observable variable y at equal time intervals (Bontempi et al., 2012). These intervals can be reflected, for instance, in hours, days, weeks, or years. Examples of time series are diverse and include the hours of sunshine in a day or a company's daily stock prices. However, irregularly spaced time series do exist but are not considered in this thesis.
A fundamental property of time series is that there is a relationship between its observations. For example, the measured temperature yesterday influences the temperature today. This relationship between lagged values of a time series is called *autocorrelation*. It follows from this context, that past values can have predictive potential. Time series which do not show any autocorrelation are called white noise (Hyndman and Athanasopoulos, 2019).
Time series can be subdivided into *linear* and *nonlinear* time series. If each observation of a time series can be described as a linear combination of past values, the time series is regarded as linear. Vice versa, if the combination is nonlinear, the time series is considered nonlinear (Priestley, 1988).
Apart from linearity, time series can be classified in *univariate* or *multivariate* time series (Harvey, 1993): a univariate time series is a sequence of one single observable variable, for example, daily commuters on a particular road, whereas multivariate time series relate to multiple observable variables. Accordingly, a multivariate time series can describe the number of daily commuters on said road and further include other observations such as weather conditions, temperature, or the day of the week.

A time series often contains various patterns. Therefore it is helpful to split a time series into its components, each representing an underlying pattern category (Hyndman and Athanasopoulos, 2019). The three main components are seasonality and cyclicity, trend, and residuals.

A seasonal component or *seasonality* is present when a time series is affected by seasonal factors such as the month of a year or the day of the week. This phenomenon is always of a fixed and known period and corresponds to calendar dates (Hyndman and Athanasopoulos, 2019). There are many examples of seasonal behavior, one being that the demand for ice cream is higher in the summer than in the winter. Seasonality is to be distinguished from cyclicity. A *cycle* exists when the data shows periodical changes that are typical of a longer duration than one year. Exemplary fluctuations can be due to economic circumstances such as the business cycle (Hyndman and Athanasopoulos, 2019).
The *trend* is defined by the existence of a long-term increase or decrease in the data, whereby it does not necessarily have to be linear. It can take many forms, such as an exponential trend. Neither does a trend have to be constant over time. It can change its gradient and evolve from a positive to a negative trend (Hyndman and Athanasopoulos, 2019). A practical example of a positive trend is the increased demand for organic products, which was observed in the past years (Willer et al., 2019). In practice, it is common to combine trend and cycle into a single trend-cycle component, which will be regarded as trend for simplicity (Hyndman and Athanasopoulos, 2019).
When trend and seasonality are removed from a time series, there is a remaining component, which will be referred to as *residuals*. This last component contains random, irregular influences and is also known as *noise* (Brockwell and Davis, 2016). The process of removing trend and seasonality is called *detrending* and *deseasonalization*, respectively (Hyndman and Athanasopoulos, 2019).

The process of splitting a time series into its constituent parts is called *time series decomposition*. An *additive decomposition* is represented by the formula

$$y_t = S_t + T_t + R_t$$

with y_t being the data, S_t the seasonal component, T_t the trend component, and R_t the residuals all observed at period t (Hyndman and Athanasopoulos, 2019). Alternatively, time series decomposition can be conducted as *multiplicative decomposition*, which is described by the following formula (Hyndman and Athanasopoulos, 2019):

$$y_t = S_t \times T_t \times R_t$$

When choosing the appropriate method of decomposition, the additive decomposition is the method of choice if the magnitude of the seasonal fluctuations, or the variation around the trend, does not vary with the level of the time series. On the other hand, when the seasonal patterns, or the variation around the trend, appears to be proportional to the level of the time series, a multiplicative approach is more promising. The literature suggests that multiplicative decomposition is common

among economic time series (Hyndman and Athanasopoulos, 2019).
Moving average methods are the basis of many decomposition methods. The underlying principle is to average values around an observation, as it assumes that observations nearby in time are likely to be close in value. By doing so, the moving average eliminates some of the randomness in the data and smooths it (Hyndman and Athanasopoulos, 2019).
The *classical decomposition* method, which dates back to the 1920s, builds upon moving averages. While being easy to use, there are several problems with this method. Firstly, it assumes that the seasonal component is not subject to change and remains constant over time. While this might be a reasonable assumption for some shorter time series, it does not necessarily apply especially for longer time series. In addition to that, the method is not robust to unusual values, for instance when the demand for air travel is disturbed by a strike or an industrial dispute. Despite its disadvantages, the classical decomposition is still widely used (Hyndman and Athanasopoulos, 2019).
Apart from the classical decomposition, there are several more advanced decomposition methods such as X11 or SEATS decomposition with various advantages and disadvantages in use (Dagum and Bianconcini, 2016). One particular versatile and robust method is the *Seasonal and Trend Decomposition using Loess* or abbreviated STL decomposition (Hyndman and Athanasopoulos, 2019). Loess is a technique for estimating nonlinear relationships and is utilized in the method developed by Cleveland et al. (1990). The advantages of STL decomposition include the handling of any type of seasonality and allowing the seasonal component to change over time. Furthermore, it can be robust to outliers, which will affect the remainder component instead of the trend and seasonal component. On the downside, STL does not handle trading day or calendar variation automatically and only facilitates additive composition out of the box. Multiplicative decomposition can only be conducted by taking logs of the data and then back transforming the components (Hyndman and Athanasopoulos, 2019).

When analyzing observations over time it is important to take into account whether the behavior changes over time. The concept of *homoscedasticity* is defined by the variance of the disturbance term in each observation being constant. On the contrary, *heteroscedasticity* is defined by the variance not being constant (Dougherty, 2011).
There are a variety of procedures to examine whether observations are of equal variances. *Levene's Test for equal variances* (Levene, 1960) or the *Bartlett Test* (Snedecor and Cochran, 1989), for instance, are popular methods.

Another important concept is *stationarity*. Weak stationarity requires the properties of a time series to not depend on the point in time at which they are observed. This means that the mean and the variance are not time-variant and remain constant. Furthermore, this implies that time series that exhibit trend or seasonality are not stationary (Hyndman and Athanasopoulos, 2019). Homoscedasticity, therefore, is a necessary, but not sufficient criterion for stationarity. Apart from stationary time series, there are *trend stationary* time series. In this case, the process has a stationary behavior around a trend (Shumway and Stoffer, 2017). When a time series is following a strictly stationary process, or simpler is strict stationary, the observations have to be identically distributed (Brockwell and Davis, 2016). White noise is a simple example of such a stationary process (Hyndman and Athanasopoulos, 2019).

To determine whether a time series follows a stationary process, *unit root tests* can be applied. Unit root tests provide a way to examine whether a time series follows a causal process or is a random walk. Random walks are characterized by the fact the value of the time series at time t is the value of the series at time $t-1$ plus a completely random movement (Shumway and Stoffer, 2017).
The *Augmented Dickey-Fuller Test (ADF)* is suitable for the analysis of stationarity and tests the null hypothesis that there exists a unit root with the alternative that there is none. If a unit root is present the time series is not stationary (Harris, 1992).

After having ascertained that a time series is not stationary, there is a variety of procedures to make it stationary. As some forecasting methods will only process stationary time series, these procedures occur regularly. Transformations such as logarithms can stabilize the variance of a time series (Hyndman and Athanasopoulos, 2019). Apart from that, *differencing* can be applied to stabilize the mean of a time series by removing changes in the level of a time series. This is achieved by calculating the differences between consecutive observations. If the differenced data is still not stationary, it is plausible to difference the data a second time. This is referred to as *second-order differencing* and removes a quadratic trend (Yaffee and McGee, 2000). Furthermore, differences can be calculated between an observation and the previous observation from the same season, for example in the case of monthly data this year's data for April and last year's data for the same month. This procedure is called *seasonal differencing*. Therefore, differencing can be a mechanism for removing or reducing trend and seasonality (Hyndman and Athanasopoulos, 2019). If a transformation was applied to a time series and the forecast was conducted on the transformed values, the predictions have to be inverse transformed to get an appropriate forecast.

3.1.2 Time Series Forecasting

Time series analysis can be subdivided into description and exploration of time series (e.g., plots or relationships between variables), hypothesis testing (e.g., examine global warming using recorded temperature data), and forecasting (Brockwell and Davis, 2016). The latter will be the area of interest for this thesis.
The field of *time series forecasting* aims to estimate how the sequence of said observations will continue in the future. The analysis is conducted to understand the underlying phenomenon of the observed data points and considers the historical data (Bontempi et al., 2012). As Palit and Popovic (2006) state, forecasting values of time series is critical in various fields of science and engineering, for instance, economics, finance, or meteorology.

An early action in time series forecasting is the dataset split, in which the time series is split into three separate sequences. These sequences are referred to as *training set*, *validation set*, and *test set*. The literature suggests different ratios ranging from 60/20/20 for training, validation, and test set respectively to 98/1/1 depending on various factors such as the forecasting task and the size of the dataset (Michelucci, 2018). The following data in Figure 1 represents monthly airline passengers and is split in the respective datasets. The blue graph shows the training set with data from January 1949 to December 1957, the green graph displays the validation set containing observations from January 1958 to June 1959, and the test set, represented by the red graph, presents observations from July 1959 to December 1960.

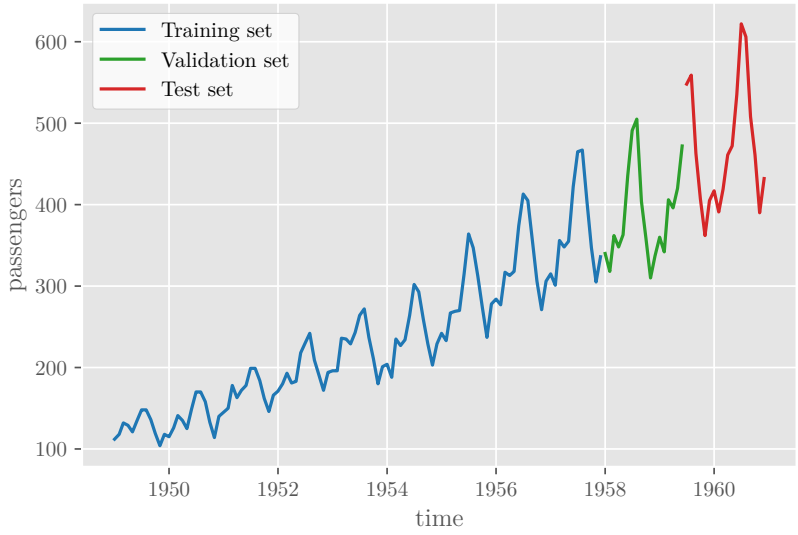


Figure 1: Exemplary dataset split

The training set is used to estimate the model, whereas the validation set is applied for hyperparameter tuning and comparing different model configurations. Finally, the test set is utilized for estimating the error in the prediction (Barrow and Crone, 2016). *Hyperparameters* are parameters whose values have to be specified before executing a model and, therefore, are not determined by the model itself. The concept *hyperparameter tuning* describes finding the best hyperparameters for maximizing a chosen optimizing metric (Michelucci, 2018).

Both statistical methods and machine learning methods can incorporate hyperparameters. Examples for methods with hyperparameters are the Autoregressive Integrated Moving Average or any neural network-based forecasting method, which will be discussed in Section 3.2.5 and Section 3.3. When a method does not incorporate hyperparameters, the training set and validation set are combined into a single training set.

The training set must consist only of observations before the observations in the validation set. The same logic applies to the validation set, which shall only contain observations after the training set and before the ones in the test set. This chronological split ensures that only information available at a particular moment in time and no future observations are used in hyperparameter tuning or constructing the forecast (Hyndman and Athanasopoulos, 2019).

Therefore, traditional cross-validation as described by Michelucci (2018) can not be applied to time series. As Hu et al. (1999) state, a *walk-forward approach* has to be followed instead. Figure 2 illustrates the series of training and test sets, in which the blue observations represent the training sets, and the red observations the test sets. The actual forecast accuracy is calculated by averaging over the test sets (Hyndman and Athanasopoulos, 2019).

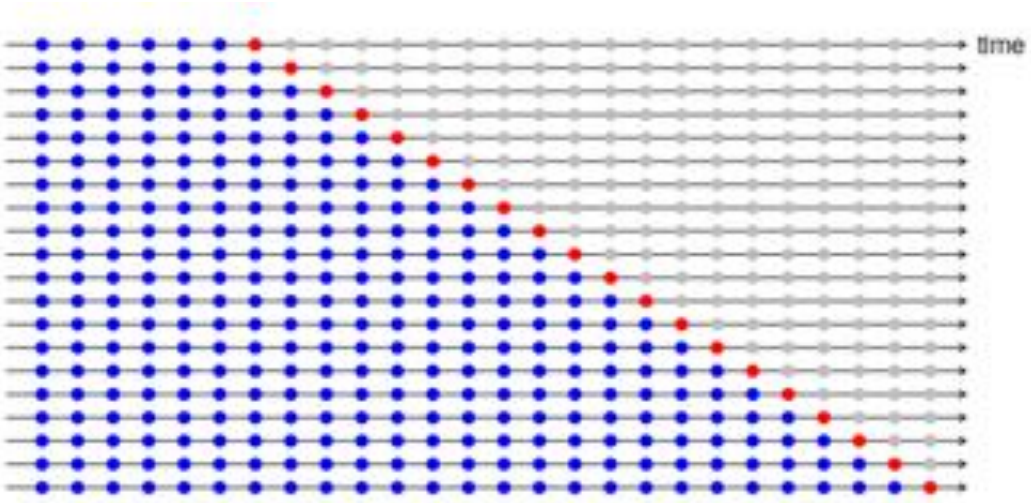


Figure 2: Walk-forward validation, reprinted from Hyndman and Athanasopoulos (2019)

As Januschowski et al. (2020) describe, forecasting is at its core an extrapolation problem and the performance of a model is evaluated by using out-of-sample accuracy measures (Gneiting and Raftery, 2007; Hyndman and Koehler, 2006; Kolassa and Schütz, 2007) or tests (Diebold and Mariano, 2002) rather than in-sample metrics. Forecast errors measuring forecast accuracy can be differentiated into *scale-dependent errors*, for instance, Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE) and *percentage errors* such as the Mean Absolute Percentage Error (MAPE) (Hyndman and Athanasopoulos, 2019). The following formulas mathematically define the MAE, MSE, and RMSE, where n is the number of observations and e_i is the difference between an observed value y_t and its forecast $\hat{y}_\mathrm{t}$ (Shcherbakov et al., 2013):

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|e_i|$$

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}e_i^2$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}e_i^2}$$

with

$$e_t = y_t - \hat{y}_t$$

In practice, the MAE is popular due to its ease of use and computability. The same popularity applies to the RMSE, despite being more difficult to interpret (Hyndman and Athanasopoulos, 2019). However, MSE and RMSE are not robust to outliers as they penalize forecasts which considerably differ from the actual value (Shcherbakov et al., 2013).

While percentage errors have the advantage of being unit-free, they have the disadvantage of being undefined if $y_\mathrm{t} = 0$ for any t in the period of interest, and having a skewed distribution when y_t is close to zero (Hyndman and Koehler, 2006). Additional caution is required as they assume the unit of measurement has a meaningful zero. As only ratio scales have meaningful zeros, measuring forecast accuracy on intervals scales, for instance, for temperature forecasts on Celsius scales, is not possible (Hyndman and Athanasopoulos, 2019). The MAPE is given by the following formula (Shcherbakov et al., 2013):

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n}100 \times |p_i|$$

with precentage errors p_i calculated based on

$$p_t = \frac{|e_t|}{y_t}$$

In practice, it is important to examine the purpose of which an error is considered. The MAPE can be applied to compare the performance of a single model on different datasets, whereas MAE, MSE, and RMSE are used when comparing competing models on the same dataset (Hyndman and Athanasopoulos, 2019; Hyndman and Koehler, 2006).

When a univariate time series Y is constituted of N observations over time, it can be formally described as $Y_{\mathrm{N}} = y_1, ..., y_{\mathrm{N}}$. Time series forecasting aims to either forecast a single value $y_{\mathrm{t+1}}$ or a sequence of H future values $y_{\mathrm{t+1}}, ..., y_{\mathrm{t+H}}$ at a chosen point in time t. Consequently, a model that forecasts only a single future value is referred to as a *one-step forecasting model*, whereas a model that performs forecasts for more than one time step is called a *multi-step forecasting model*. The length of sequence H is hereafter referred to as the forecasting horizon or short *horizon*.

The relevant literature describes a variety of H-step forecasting strategies, each of which exhibits different advantages and disadvantages. Furthermore, these strategies differ in various other aspects such as the number of models that have to be developed and the computational time required. They can be applied for both, univariate and multivariate time series (Taieb et al., 2012).

The *recursive strategy* uses a single model to predict a one-step-ahead future value and uses the predicted values as input for predicting the next step with the same model for the entire horizon (Bontempi et al., 2012). As Cheng et al. (2006) point out, this strategy utilizes previous predictions and is therefore susceptible to error accumulation, which means that errors that emerged in the past are propagated into future predictions. Despite this shortcoming, the literature suggests that this strategy can be applied successfully (Bontempi et al., 2012).

In the *direct strategy*, every step is predicted independently using a separate model, which results in H models predicting H steps in the future (Bontempi et al., 2012). Contrary to the recursive strategy, it is not exposed to error accumulation as it does not use any approximated values to conduct the forecast. On the other hand, occurring statistical dependencies between the predictions are neglected and the strategy requires more comprehensive computational time (Bontempi et al., 2012).

As the name suggests, the *DirRec strategy* (Sorjamaa and Lendasse, 2006) is a combination of the previous strategies. It conducts the forecasts with H models for H steps and adds the previous predictions to the input vector (Bontempi et al., 2012).

The strategies mentioned above share a common feature: the model maps data from a multi-input to a single-output. This neglects the existence of stochastic dependencies between future values, for instance, $y_{\mathrm{t+k}}$ and $y_{\mathrm{t+k+1}}$. Therefore, it biases the prediction accuracy. To solve this matter, multi-step forecasting models follow a multiple output strategy to conduct forecasts as a vector of future values

of the time series instead of a single value (Bontempi et al., 2012). The *multi-input multi-output strategy (MIMO)* avoids the shortcomings of the direct and recursive strategy, but reduces the model's flexibility as every horizon is forecasted using the same model structure (Taieb et al., 2009). As Taieb et al. (2012) states, the *DIRMO strategy*, as a combination of direct and MIMO strategy, can offer a beneficial trade-off between preserving a larger degree of the stochastic dependency between the future vales and provides greater flexibility of the predictor.

As Hyndman and Athanasopoulos (2019) describe, a time series exhibits different characteristics at the same time. From that follows that it is crucial to rely on a forecasting method that is capable of dealing with these characteristics in an effective manner.

3.2 Statistical Methods for Time Series Forecasting

The following chapter will elaborate on statistical methods for time series forecasting. Starting with basic methods such as naive approaches, this chapter will discuss more advanced methods like exponential smoothing and the related Holt-Winters' Method. Moreover, it introduces ARIMA models and gives an example of recent developments. An evaluation of the strengths and limitations of statistical methods closes this chapter.

3.2.1 Naive Approaches

The *naive approach* is a procedure that assumes that today's value will be the value of tomorrow. Despite the method's simple nature, it works remarkably well for many economic and financial time series (Hyndman and Athanasopoulos, 2019). It sets all forecasts $\hat{y}$ for the time period t to $t+h$ to be the value of the last observation y_t. The following formula defines the naive approach, where h denotes the forecasting horizon (Hyndman and Athanasopoulos, 2019):

$$\hat{y}_{t+h|t} = y_t$$

The *seasonal naive approach* is a variant of the naive approach and is useful for highly seasonal data. It sets each forecast equal to the last observed value from the same season of the previous year. For instance, in the case of monthly data, when forecasting this year's value for May, the value is set to the one from that month in the previous year.

Naive approaches are simple but can be surprisingly effective. Often they are considered as baseline methods or benchmarks, which are used to compare other methods

against. If a more advanced method can not obtain better results, it is not worth considering.

3.2.2 Regression Models

The basic concept of *regression models* is that there is a relationship between the forecast variable y and one or more predictor variables x. The forecast variable is often referred to as the dependent variable whereas the predictor variable is referred to as the independent variable (Hyndman and Athanasopoulos, 2019). A practical example is the forecast of daily electricity demand y using temperature x_1 and the day of the week x_2 as predictors.
In the simplest case, the *Linear Regression* model represents a linear relationship between the forecast variable y and a single predictor variable x, where the coefficients β_0 and β_1 represent the intercept and the slope of the line respectively and the random error ϵ_t denotes the deviation from the underlying straight-line model (Hyndman and Athanasopoulos, 2019):

$$y_t = \beta_0 + \beta_1 x_t + \epsilon_t$$

Apart from the linear case, a regression can take more advanced forms such as the *Multiple Linear Regression* taking into consideration more than one predictor variable, or the *Nonlinear Regression* modeling nonlinear relationships utilizing transformations like logarithms (Hyndman and Athanasopoulos, 2019).

The Linear Regression can be adjusted to forecast a univariate time series. In the univariate case, the Linear Regression describes the relationship between the observations of a forecast variable y_t and their respective point in time x_t. The model is obtained by estimating the coefficients $\hat{\beta}_0$ and $\hat{\beta}_1$, and ignoring the error in the regression equation. Finally, the prediction for a value $\hat{y}_t$ is calculated by considering the corresponding future point in time x_t in the forecasting formula, which is defined by:

$$\hat{y}_t = \hat{\beta}_0 + \hat{\beta}_1 x_t$$

As previously described in Section 3.1.2, to produce reasonable predictions only information available at the moment of conducting the forecast is permitted to be used to estimate the model's coefficients. This ensures reliable ex-ante forecasts (Hyndman and Athanasopoulos, 2019).

3.2.3 Exponential Smoothing

Exponential smoothing was introduced in the middle of the 20th century (Brown, 1959; Winters, 1960; Holt, 2004) and laid the foundation of many forecasting methods. Forecasts conducted using exponential smoothing methods are weighted averages of past observations, with the weights declining exponentially as the observations are further in the past.

The simplest form of exponential smoothing methods is called *Simple Exponential Smoothing (SES)* and is suitable for data with no visible trend or seasonality (Hyndman and Athanasopoulos, 2019). A one-step-ahead forecast for $t+1$ is a weighted average of all the observations from y_1 to y_T. The forecasting formula is therefore defined by (Hyndman and Athanasopoulos, 2019)

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha\left(1-\alpha\right) y_{T-1} + \alpha\left(1-\alpha\right)^2 y_{T-2} + ...,$$

where α is the smoothing parameter with $0 < \alpha \leq 1$. This smoothing parameter controls the rate at which the weights decrease. If α is close to 0, more weight is given to the observations from the more distant past. Contrary, if α is close to 1, more weight is given to the most recent observations. If $\alpha = 1$, the forecast gives all the weight to the very last observation and the value is equal to the one obtained by a naive forecast.

As the classification of Pegels (1969) shows, there are methods apart from the simple variant that additionally include trend components, or trend and seasonal components. A method incorporating trend is known as *Holt's Linear Method*. The forecasting equation and the two involved smoothing equations l_t for level and b_t for trend are defined as follows (Hyndman and Athanasopoulos, 2019):

$$\hat{y}_{t+h|t} = l_t + hb_t$$

$$l_t = \alpha y_t + \left(1-\alpha\right)\left(l_{t-1} + b_{t-1}\right)$$

$$b_t = \beta\left(l_t - l_{t-1}\right) + \left(1-\beta\right) b_{t-1}$$

In this definition, l_t is the estimate of the series's level at time t while b_t denotes an estimate of the trend of the series at time t. The smoothing parameter α for the level and the smoothing parameter β for the trend fall into an interval between zero and one: $0 \leq \alpha, \beta \leq 1$.

Another method based on exponential smoothing and incorporating trend as well as seasonality components, which meets with much approval in practice, is presented in the next section.

3.2.4 Holt-Winters' Method

The origin of the *Holt-Winters' Method (HW)* dates back to the 1950s when researchers were working on a high accuracy and low-cost forecasting model for the Office of Naval Research in the United States of America (Da Veiga et al., 2014). The researchers showed that the method of the exponentially weighted moving average, which was very popular during that time, could be utilized not only to smooth the level of a time series but furthermore smooth trend and seasonality (Holt, 2004). As the model extends Holt's Linear Method by incorporating seasonal components and therefore having three smoothing equations, it is sometimes referred to as Triple Exponential Smoothing (TES) (Siregar et al., 2017).

There exist two variations to this method, that differ regarding their seasonal component. The additive method is used when the seasonal variations are approximately constant through the series, whereas the multiplicative method is utilized when the seasonal variations are changing proportionally to the level of the series (Hyndman and Athanasopoulos, 2019). The *Holt-Winters' Additive Method* is defined as follows (Hyndman and Athanasopoulos, 2019)

$$\hat{y}_{t+h|t} = l_t + hb_t + s_{t+h-m(k+1)}$$

$$l_t = \alpha\left(y_t - s_{t-m}\right) + (1-\alpha)\left(l_{t-1} + b_{t-1}\right)$$

$$b_t = \beta\left(l_t - l_{t-1}\right) + (1-\beta)\, b_{t-1}$$

$$s_t = \gamma\left(y_t - l_{t-1} - b_{t-1}\right) + (1-\gamma)\, s_{t-m},$$

where k is the integer part of $\frac{(h-1)}{m}$. Similar to Holt's Linear Model this method has smoothing equations for level l_t and trend b_t but additionally includes a third smoothing equation for seasonality s_t. For the corresponding smoothing parameters α, β, γ applies $0 \leq \alpha, \beta, \gamma \leq 1$. The variable m denotes the periodicity of the time series, for instance, $m = 4$ for quarterly data, or $m = 12$ for monthly data. Having to specify a value for the periodicity of the time series can be problematic if there is no reliable estimate for it. On the other hand, it allows the practitioner to use a priori knowledge in the model. The implications of this circumstance will be further discussed in Chapter 3.2.7.

The second variant, *Holt-Winters' Multiplicative Method*, is defined by the following formula and uses the corresponding variables (Hyndman and Athanasopoulos, 2019):

$$\hat{y}_{t+h|t} = \left(l_t + hb_t\right) s_{t+h-m(k+1)}$$

$$l_t = \alpha\frac{y_t}{s_{t-m}} + (1-\alpha)\left(l_{t-1} + b_{t-1}\right)$$

$$b_t = \beta\left(l_t - l_{t-1}\right) + (1-\beta)\, b_{t-1}$$

$$s_t = \gamma \frac{y_t}{(l_{t-1} + b_{t-1})} + (1 - \gamma)\, s_{t-m}$$

Exponential smoothing methods and the related Holt-Winters' Method are widely used approaches for time series forecasting. Another complementary approach is discussed in the following section.

3.2.5 ARIMA Models

As described in two previous chapters, exponential smoothing methods focus on trend and seasonality, whereas the class of *ARIMA* models describes autocorrelations in the data (Hyndman and Athanasopoulos, 2019). The acronym ARIMA stands for *Autoregressive Integrated Moving Average*. To fully understand this model, this section will first explain the essential components.

The basic idea of *autoregressive models* is that the current value y_t of the series can be explained as a function of p past values, x_{t-1}, x_{t-2}, ... , x_{t-p}, where p determines the number of steps into the past (Shumway and Stoffer, 2017). An autoregressive model of order p, abbreviated as $AR(p)$, can be defined as (Hyndman and Athanasopoulos, 2019; Shumway and Stoffer, 2017):

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t + c$$

The term ϵ_t describes an error term, whose variance will only change the scale of the series but not the patterns, and ϕ_1, ϕ_2, ... , ϕ_p are parameters with $\phi_p \neq 0$. The variable c denotes a constant. This class of models has proven to be flexible at handling a wide range of different time series patterns (Hyndman and Athanasopoulos, 2019).

The second component are *moving average models*, whose core principle is to use past forecast errors ϵ_t in a model similar to regression (Hyndman and Athanasopoulos, 2019). A moving average model of order q, which is referred to as $MA(q)$ model, can be defined as following (Hyndman and Athanasopoulos, 2019; Shumway and Stoffer, 2017):

$$y_t = \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q} + \epsilon_t + c$$

Similar to autoregressive models, c denotes a constant, and θ_1, θ_2, ..., θ_p are parameters with $\theta_p \neq 0$. To avoid any misunderstanding, moving average models are not to be confused with the moving average smoothing introduced in Section 3.1.1.

These two main components can be combined into an *Autoregressive Moving Average (ARMA)* model. An $ARMA(p, q)$ model, which depends on p past values and q of its past values of error, can be defined as (Shumway and Stoffer, 2017; Gupta, 2018):

$$y_t = \phi_1 y_{t-1} + ... + \phi_p y_{t-p} + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q} + \epsilon_t + c,$$

with ϵ_t describing an error term and ϕ_p, $\theta_q \neq 0$. As Gupta (2018) states, ARMA models can not be applied to non-stationary time series. To conduct forecasts on non-stationary ARMA models have to be generalized.

This particular generalization is achieved by combining differencing to the ARMA model and results in an ARIMA model. As presented in Section 3.1.1, differencing is a method to make a time series stationary, and therefore making ARIMA models applicable for non-stationary time series (Gupta, 2018). The $ARIMA\,(p, d, q)$ model can be defined as (Hyndman and Athanasopoulos, 2019):

$$y'_t = \phi_1 y'_{t-1} + ... + \phi_p y'_{t-p} + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q} + \epsilon_t + c$$

The differenced series is denoted by y'_t and can be differenced more than once, for instance, in a second-order differencing. Like the concepts introduced before, p is the order of the autoregressive part, and q is the order of the moving average part. The differencing is manifested in d, which denotes the degree of differencing involved. The parameters p, d, and q are specified beforehand and, consequently, are the ARIMA model's hyperparameters.

Table 1 summarizes the connection of ARIMA models to the underlying concepts. The number zero can be understood in a way, that this part of the model is neglected, for instance, an $ARIMA\,(p, 0, q)$ model is an $ARMA\,(p, q)$ model as there is no differencing involved.

ARIMA model	**equivalent**
ARIMA(p,0,0)	AR(p)
ARIMA(0,0,q)	MA(q)
ARIMA(p,0,q)	ARMA(p,q)

Table 1: Cases of ARIMA models

ARIMA models are valued for their forecasting accuracy and flexibility in representing different types of time series (Khandelwal et al., 2015). Apart from the presented models, there are other variants and extensions described in the literature. Seasonal ARIMA models, abbreviated *SARIMA*, can model seasonal data (Hyndman and Athanasopoulos, 2019), whereas *ARIMAX* models can incorporate explanatory variables (Cools et al., 2009). Finally, *SARIMAX* models represent a combination of the aforementioned models (Cools et al., 2009).

3.2.6 Recent Developments

As mentioned previously in the introduction to this chapter, the foundation of statistical methods for time series forecasting was laid in the past century. This section aims to give an example of recent developments in the area and shows that statistical methods are far from being old news.

Prophet is open-source software released by the American social networking service Facebook's core data science team (Facebook Open Source, nd). Taylor and Letham (2018) introduced its implementation, of which the authors state that it allows many different users to forecast a large number and a variety of time series. It is used within Facebook and is supposed to work best with time series that have strong seasonal effects as it specifically handles public holidays (Facebook Open Source, nd). The implementation of Prophet is based on the Stan platform, a platform for statistical modeling and high-performance statistical computation (Stan, nd). In scientific research, Prophet was already utilized (Borowik et al., 2018).

This confirms the opinion of many authors, that statistical methods are still highly relevant. After introducing several methods in the previous sections, the next paragraph will give an evaluation of statistical methods.

3.2.7 Strengths and Limitations of Statistical Methods

Statistical methods look back on a long history, as the forecasting domain has been dominated by them from the 1960s on (Bontempi et al., 2012). This manifests in their wide usage and approval of many forecasters.

The main reasons for that are that statistical methods are simple, flexible, and can be used to model several phenomena (Domingos et al., 2019). Apart from that, they work well when few data is available (Makridakis et al., 2018b). Furthermore, some statistical methods can consider a priori knowledge. That circumstance gives the forecaster the possibility to simplify the expected task for the model. In the Holt-Winters' Method, the periodicity of the time series is provided to the model. Thus, the model does not have to estimate the periodicity itself. On the other hand, this can lead to inaccurate forecasts if the information is unknown to the forecaster or falsely specified.

Despite the well-known strengths of statistical methods, there have been some limitations found in the literature, the main one being missing nonlinearity. As the presented statistical methods mostly assume linear relationships between past values, they can not capture nonlinear patterns. From that follows that the approximation of linear models to complex real-world problems might not always be adequate (Zhang, 2003). This inadequate approximation is confirmed by Domingos et al.

(2019), who observe that real-world time series commonly present linear and non-linear patterns at the same time.
Beyond this, these presented forecasting methods are mostly univariate and can only handle one time series at a time. Consequently, they treat every time series in an isolated manner. Therefore, potential similarities between related time series are neglected, and no information is shared between the forecasts (Bandara et al., 2019).

As this paragraph presented, statistical methods are powerful tools for time series forecasting. In the recent past, another approach to forecasting methods has been extensively studied and developed. These methods will be described in the following chapter.

3.3 Machine Learning Methods for Time Series Forecasting

This chapter is dedicated to machine learning methods for time series forecasting. It starts by outlining their history and foundations before introducing the principles of Artificial Neural Networks. In the following, this chapter presents Feedforward Neural Networks, Recurrent Neural Networks, and briefly examines further kinds of networks. Finally, this chapter gives an evaluation of the strengths and limitations of machine learning methods.

3.3.1 History and Foundations

Today, *Artificial Intelligence (AI)* is a growing field with various research topics and practical applications. AI has gained remarkable prominence over the past decade, driven by numerous applications such as in autonomous vehicles, speech and image recognition, machine translation, or spectacularly beating a world chess champion (Makridakis, 2017; Makridakis et al., 2018b).

Some AI systems can acquire their knowledge by extracting patterns from raw data without being explicitly programmed (Goodfellow et al., 2016). This capability is known as *Machine Learning (ML)*. Contrary to traditional programming, which relies on step-by-step coding instructions based on logic, if-then rules and decision trees, AI-based systems are learning by trial and error and improving their performance over time (Makridakis et al., 2018b). A subdiscipline of ML is known as *Deep Learning (DL)* and deploys *Artificial Neural Networks (ANN)* that have several hidden layers (Goodfellow et al., 2016). Section 3.3.2 will discuss ANNs in detail.
Machine learning algorithms classify into *supervised* and *unsupervised* learning algorithms, whereby supervised learning distinguishes further into classification and regression (Friedman et al., 2001). A supervised learning task is characterized by

learning a function that maps an input X, consisting of one or multiple so-called features, to an output Y based on known input-output pairs. Supervised learning requires labeled data, which means a set of examples illustrating the desired behavior. For example, in an image recognition task that aims to recognize cats, labeled images are used. These labeled images indicate whether the image depicts a cat. In an unsupervised task, the model learns solely from data and does not rely on labeled data (Goodfellow et al., 2016).

In the domain of time series forecasting, AI and particularly ML are extensively researched as a considerable amount of published papers shows (Makridakis et al., 2018b). The objective of ML-based methods is thereby the same as that of statistical ones. Both aim to improve forecasting accuracy by minimizing a particular loss function. The two approaches differ in how this minimization is performed: the statistical methods described in Chapter 3.2 mostly apply linear algorithms, while the ML methods presented in this chapter can use nonlinear ones.

Machine learning methods can not forecast time series without preprocessing the data. The data has to be framed into a supervised learning setting (Brownlee, 2017). Bontempi et al. (2012) propose an approach to model an input-output setting for one-step forecasting, where the training set consists of a scalar output and a vectorial input. The input is modeled as an $[(N-n-1)\times n]$ input data matrix X and a $[(N-n-1)\times 1]$ output vector Y, representing a time series with N observations and n previous values to be considered (Bontempi et al., 2012):

$$X = \begin{pmatrix} y_1 & y_2 & \cdots & y_n \\ y_2 & y_3 & \cdots & y_{n+1} \\ \vdots & \vdots & \vdots & \vdots \\ y_{N-n} & y_{N-n+1} & \cdots & y_{N-1} \end{pmatrix}$$

$$Y = \begin{pmatrix} y_{n+1} \\ y_{n+2} \\ \vdots \\ y_N \end{pmatrix}$$

In order to perform an H-step forecast, the input data matrix is adapted to a $[(N-n-h)\times n]$ matrix and the output vector becomes a $[(N-n-h)\times h]$ matrix:

$$X = \begin{pmatrix} y_1 & y_2 & \cdots & y_n \\ y_2 & y_3 & \cdots & y_{n+1} \\ \vdots & \vdots & \vdots & \vdots \\ y_{N-n-h} & y_{N-n-h+1} & \cdots & y_{N-h} \end{pmatrix}$$

$$Y = \begin{pmatrix} y_{n+1} & y_{n+2} & \cdots & y_{n+h} \\ y_{n+2} & y_{n+3} & \cdots & y_{n+h+1} \\ \vdots & \vdots & \vdots & \vdots \\ y_{N-h+1} & y_{N-h+2} & \cdots & y_N \end{pmatrix}$$

With this supervised learning setting, machine learning algorithms can be used as a one- or multi-step forecasting model. The following section will now introduce the essentials of Artificial Neural Networks.

3.3.2 Artificial Neural Networks

Artificial Neural Networks are inspired by the human brain, which learns based on experience (Nielsen, 2015). Research on a formal description of this learning process dates back to the mid of the last century. McCulloch and Pitts (1943) first described simplified *neurons*, which became the central element of today's ANNs. Later Rosenblatt (1958) developed another type of artificial neuron referred to as *perceptron*, which was the first one to be implemented. As Figure 3 exhibits, perceptrons take several binary inputs x_1, x_2, ..., x_N, and produce a single binary output. Each input is assigned a weight w_1, w_2, ..., w_N, expressing the importance of respective input to the output (Nielsen, 2015).

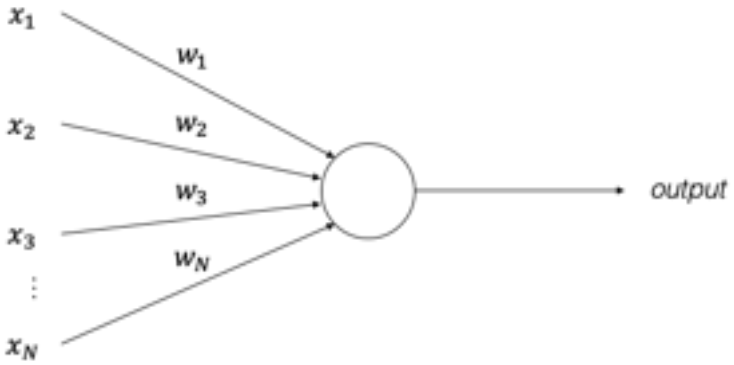


Figure 3: Model of a perceptron, adapted from Nielsen (2015)

As the output of a perceptron is binary, it can either take 0 or 1. A situation where the output becomes 1 is referred to as *activation* (Goyal et al., 2018). The output is determined if the weighted sum of the inputs $\sum_{i=1}^{n} w_i x_i$ is greater than some threshold value. Instead of using a threshold value, it is common to use *bias* denoted by b instead. The output of a perceptron is therefore defined as follows, with x denoting a vector of inputs and w denoting a vector of weights (Nielsen, 2015; Goyal et al., 2018):

$$\text{output} = \begin{cases} 0, & \text{if } w \cdot x + b \leq 0 \\ 1, & \text{if } w \cdot x + b > 0 \end{cases}$$

The *learning process* is realized during the *training* of the network, which adjusts the weights of the inputs (Nielsen, 2015). As the *activation function* significantly influences the learning process, it becomes of crucial importance to the model's learning performance (Ramachandran et al., 2017).

In the introduced example, the activation function takes the form of a step function, as Figure 4 shows. Due to its abrupt nature, deploying such an activation function results in a behavior that small changes in weight or bias can significantly affect the output (Nielsen, 2015).

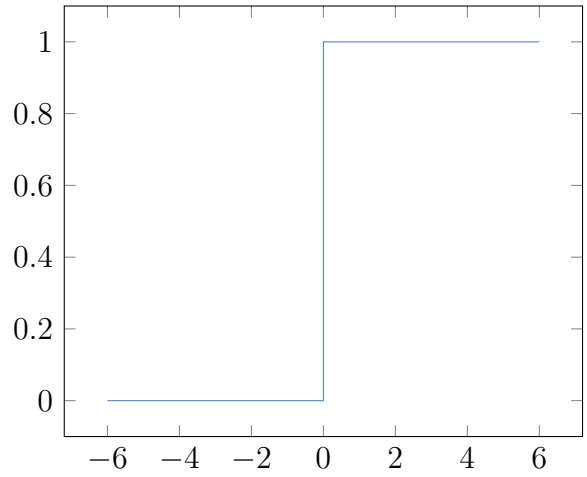


Figure 4: Step function

This step function is a nonlinear activation function. A variety of further nonlinear activation functions exist in the literature, and the following paragraphs present two of them.

Sigmoid functions overcome the step function's shortcoming of the binary output and achieve the desired behavior, in that small changes in weights and bias cause only a small change in output (Goyal et al., 2018). Thus, the sigmoid function allows the output to take any value in an interval from 0 to 1. A perceptron utilizing a sigmoid activation function is sometimes referred to as a sigmoid neuron (Nielsen, 2015). Sigmoid functions are defined by the following formula and have a smoother shape, as Figure 5 depicts (Goyal et al., 2018):

$$\sigma(x) = \frac{1}{1 + e^{-(\mathrm{w} \cdot \mathrm{x} + \mathrm{b})}}$$

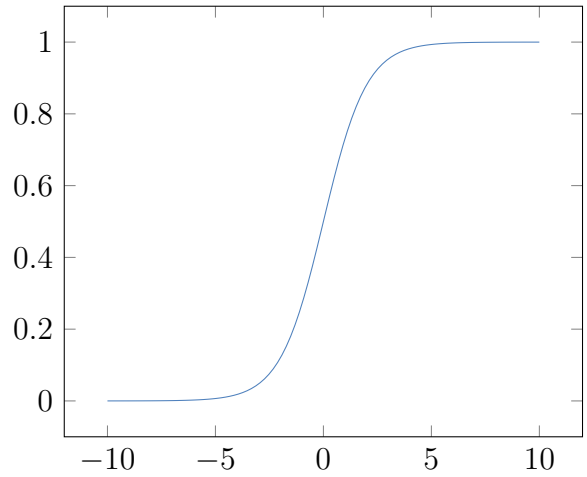


Figure 5: Sigmoid function

Thanks to its simplicity and effectiveness, the *Rectified Linear Unit (ReLU)* has become the most widely used activation function (Ramachandran et al., 2017). The ReLU is defined by the following formula, and Figure 6 illustrates its shape (Goyal et al., 2018):

$$f(x) = max(0, x)$$

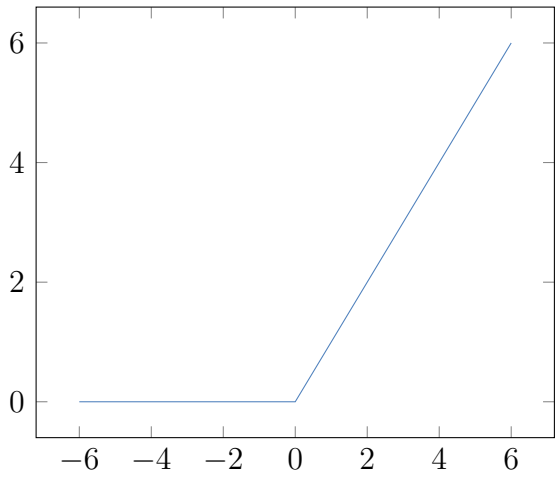


Figure 6: Rectified Linear Unit

This section generally discussed the essentials of ANNs. However, different types of neural networks (NNs) vary in architecture and usage (Goyal et al., 2018). The following section will now introduce a particular type of neural network.

3.3.3 Feed Forward Neural Networks

The previous section introduced the basic principles of neurons and perceptrons. Unfortunately, a single neuron is not adequate to model complex, nonlinear relationships. Therefore, networks of neurons have to be used. These networks consist of vertical stacks of neurons, which build up a *layer*, and connections between them. Networks with several layers are referred to as *Multilayer Perceptrons (MLP)* or *Feedforward Neural Networks (FFNN)*. Figure 7 gives an overview of the MLP architecture.

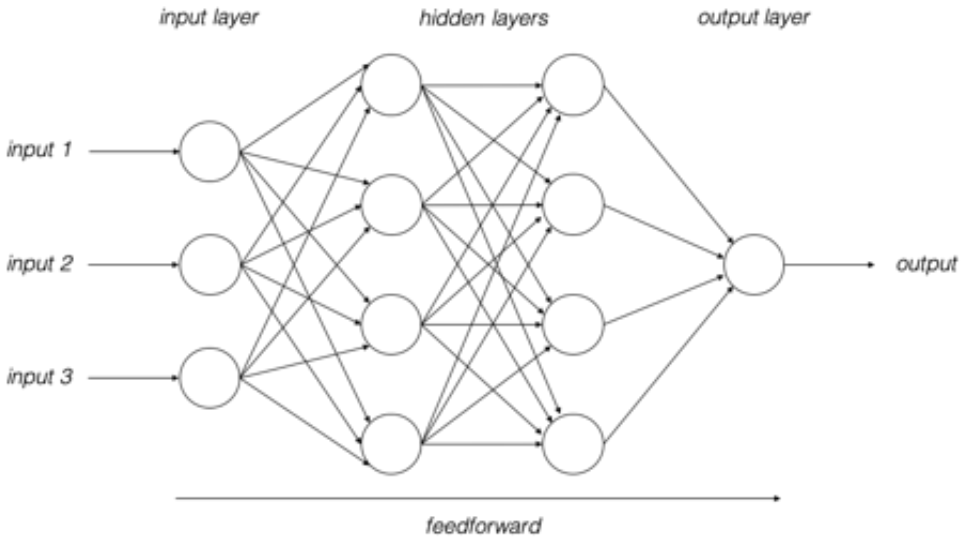


Figure 7: Multilayer Perceptron architecture, adapted from Do et al. (2019)

There are three basic types of layers. The first layer of the network is termed the *input layer* and consists of input neurons, whereas the last layer is called the *output layer* and consists of output neurons, respectively. The network depicted in Figure 7 has only one output neuron. The layers between the input and output layer are referred to as *hidden layers*, as the neurons in these layers are neither input nor output (Nielsen, 2015).
The density and type of connections between the layers of the network are referred to as *configuration* (Goyal et al., 2018). As previously mentioned in Section 3.3.1, MLPs belong to Deep Learning, as they can include several layers.
FFNNs carry their name as information flows through the network only in one direction from the input to the output layer. There are no feedback connections in which outputs of the model are fed backward. Consequently, this leads to the fact that the network does not have any loops (Goodfellow et al., 2016; Nielsen, 2015).

In general, the learning process of neural networks consists of three steps: forward propagation, backward propagation, and gradient descent (Goodfellow et al., 2016). The consecutive execution of these steps is a training step, also referred to as iter-

ation. An iteration over all samples in the training set is called an epoch (Nielsen, 2015). *Forward propagation* describes that the initial information provided by the input x propagates through the hidden units at each layer and finally produces the initial output y (Goodfellow et al., 2016).
As stated before, training drives the learning process forward. The goal of training is to minimize a *cost function* or sometimes called a *loss function* $C(w, b)$, which is a function of weights w and biases b (Nielsen, 2015). A typical example of a loss function is given by the MSE, which measures the loss from the target. In other words, the deviation of the forecast from the actual values (Michelucci, 2018). The *backpropagation* allows the loss information to flow backward through the network to compute the gradient (Rumelhart et al., 1986).
In the third step of learning, *gradient descent* tries to obtain a minimum of the specified loss function. In practice, the initial weights and biases are randomly initialized (Michelucci, 2018). A gradient ∇C is a vector, which components are the partial derivatives of the cost function concerning the weight vector w. The change in the cost function is used as a stopping criterion (Michelucci, 2018). The hyperparameter *learning rate* η controls the step sizes during the gradient computations (Nielsen, 2015). Choosing the wrong learning rate can result in failing to find a minimum (Michelucci, 2018).
As gradient descent is computationally slow, *Stochastic Gradient Descent (SGD)* is often used to speed up the learning process. The underlying principle is that ∇C is computed only for a small sample of randomly chosen training inputs, which is referred to as batch. Averaging over this sample gives a reasonable estimate of the true ∇C while significantly reducing the time required (Nielsen, 2015). Apart from the SGD, there are several more advanced modifications to the gradient descent described in the literature, such as Momentum, RMSProp, or Adam. Mainly Adam finds extensive use as it is considered faster and better than other methods (Michelucci, 2018).

There is one particular analogy between machine learning and statistical methods, specifically between NNs and AR models. As presented in Section 3.2.5, autoregressive models assume a linear function of past values to conduct a forecast. The same lagged values of a time series are the input to a NN, which can approximate a nonlinear function. Therefore, some authors describe this as a *Neural Network Autoregression* or *NNAR model* (Hyndman and Athanasopoulos, 2018). It becomes clear that an NNAR model taking p lagged values as input is equivalent to an $ARIMA\,(p, 0, 0)$ and an $AR\,(p)$ model, respectively. However, the NN-based approach has the advantage of not having restrictions on the parameters to ensure stationarity (Hyndman and Athanasopoulos, 2018).

Feedforward Neural Networks are applied to time series forecasting in various do-

mains. Do et al. (2019) describe several successful implementations to traffic data, even outperforming well-established methods such as ARIMA. This clearly shows the enormous potential of FFNNs.

3.3.4 Recurrent Neural Networks

As described in the previous section, ANNs do not have any feedback connections. This leads to the fact that each input is processed in an isolated manner. Consequently, no information is exchanged between the input windows even though they come from the same time series (Brownlee, 2017). *Recurrent Neural Networks (RNNs)* are a promising approach to overcome this shortcoming.

RNNs are designed to process a sequence of values x_1, ..., x_N one element at a time (Goodfellow et al., 2016). Thereby, the hidden units maintain a state vector that implicitly contains historical information of past elements of the sequence resulting in the fact that the current prediction $\hat{y}_t$ not only depends on the current input x_t but also on previous inputs x_{t-1}, ..., x_{t-N} (LeCun et al., 2015).
Figure 8 visualizes the two elementary representations of RNNs. The variable x denotes the input, h a hidden state, and $\hat{y}$ the output in both models. At each time step, the model processes the hidden state in addition to the input to an output. W_h stands for the weight matrix, which connects the hidden state from two consecutive time steps. The weight matrices W_x and W_y connect the input to the hidden state and the hidden state to the output, respectively. The representation on the left-hand part of Figure 8 is referred to as *folded model* (Goodfellow et al., 2016).

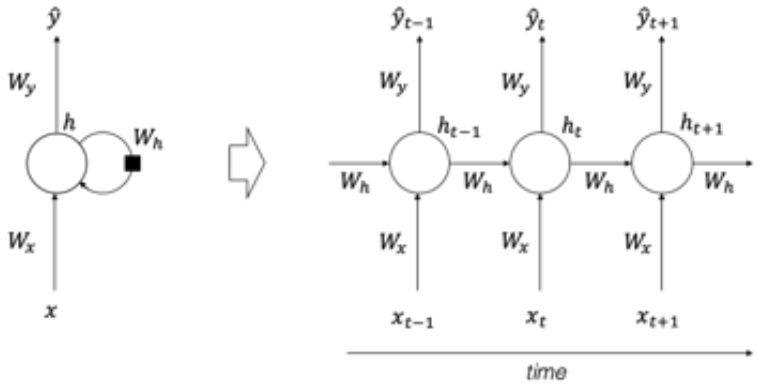


Figure 8: Recurrent Neural Network folded and unfolded into time, reprinted from LeCun et al. (2015)

As a time series is a sequential input, the RNN can be *unfolded into time*, which the right-hand part of Figure 8 depicts. This representation illustrates that the output $\hat{y}_t$ is dependent on the input of the current period x_t and the previous hidden state

h_{t-1}. Furthermore, it becomes apparent that the weight matrices are shared over time (Goodfellow et al., 2016).

Although the training procedure of RNNs builds on the same three steps as FFNNs, there is a difference in the backward propagation. As RNNs share their weights over time, backpropagation is characterized as *backpropagation through time* (Werbos, 1990). However, the backpropagation through time can be problematic as it leads to the *unstable gradient problem* (Nielsen, 2015). It causes the gradients to grow exponentially or become very small in value, which is referred to as *exploding gradient* and *vanishing gradient*, respectively. The consequences are unstable learning and difficulties in learning long-term dependencies (LeCun et al., 2015; Bengio et al., 1994).

Introduced by Hochreiter and Schmidhuber (1997), *Long Short-Term Memory Networks (LSTMs)* are an advancement to standard RNNs and are designed to overcome the unstable gradient problem. The key innovation of LSTMs is the introduction of a *cell state*, which easily lets unaltered information flow through the network. Therefore, the cell state represents the long-term memory, while the hidden state is the short-term memory. The LSTM can remove or add information to the cell state, a process that is regulated by gates (Olah, 2015). This structure enables the LSTM to potentially remember information for an extended time (Salehinejad et al., 2018). Figure 9 gives an overview of an LSTM cell.

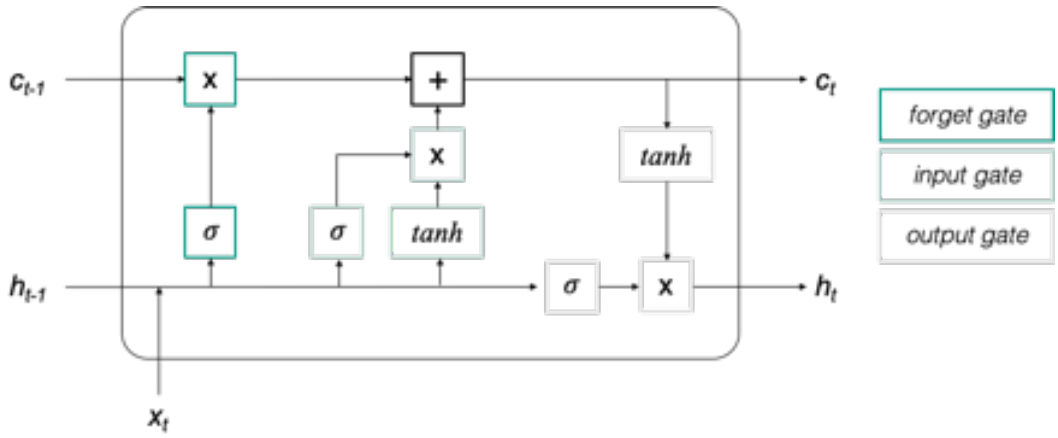


Figure 9: Long Short-Term Memory cell, adapted from Olah (2015)

LSTMs consist of repeating cells, which form a chain-like structure. The input at time t is denoted by x_t, the hidden state and cell state at time t with h_t and c_t, respectively. The *gates* optionally let information through. This is achieved by the sigmoid neural net layer σ and a pointwise multiplication operation. As described in Section 3.3.2, the sigmoid function takes values from 0 to 1 and describes the extent of information that is let through a gate. The closer the value to 0, the less

information passes through the gate and vice versa (Olah, 2015).
The LSTM unit has three different gates: the forget gate, the input gate, and the output gate. At first, the *forget gate* updates the previous cell state $c_{t\text{-}1}$. This is achieved by concatenating the current input x_t to the previous hidden state $h_{t\text{-}1}$ and feeding it in the sigmoid neural net layer (Olah, 2015). Thereby, both inputs contain short-term information. If there is a positive relationship with the cell state, the LSTM will keep the information in the cell state (Gers et al., 2000).
The *input gate* decides what new information is going to be stored in the cell state. Like the forget gate, it takes the current input x_t and the previous hidden state $h_{t\text{-}1}$ and feeds them into a sigmoid layer and a tanh layer. Thereby, the tanh layer can take values from -1 to 1. The sigmoid layer decides which values are going to be updated while the tanh layer creates a vector of new candidate values that potentially are added to the new cell state. These two components are concatenated and added to the forget gate's output to obtain the new cell state c_t (Olah, 2015).
Finally, the *output gate* determines what information shall be added to the hidden state and therefore stored in the long-term memory. The output gate takes the updated cell state c_t, the current input x_t, and the previous hidden state $h_{t\text{-}1}$. It utilizes a sigmoid layer to decide which parts of x_t and $h_{t\text{-}1}$ are combined with c_t that was previously filtered by the tanh layer. The outcome of the output gate is the updated hidden state h_t, which is passed to the next time step along with the updated cell state c_t (Olah, 2015).

The immense potential of LSTMs in capturing long-term dependencies comes with an increase in computational cost. The *Gated Recurrent Unit (GRU)* described by Cho et al. (2014) simplifies the LSTM architecture by reducing the number of gates and using only one memory representation instead of two. Consequently, this makes the GRU computational less expensive (Cho et al., 2014).
Further modifications to traditional LSTMs are *dilated LSTMs* that use the hidden state from previous but not necessarily last steps (Vezhnevets et al., 2017). This allows the network to remember information from earlier time instances and results in an improved long-term memory performance (Redd et al., 2019; Smyl, 2020).
Another modification are *LSTMs with peephole connections* which are described in detail by Gers et al. (2000).
These modifications to the traditional LSTM have been extensively studied in the literature. Unfortunately, as a detailed analysis of Greff et al. (2017) shows, these modifications do not always necessarily result in a better performance.

The LSTM is not immune to a critical challenge all machine learning methods face: the ability of a model to perform well on new, unseen inputs known as *generalization*. There are several techniques to support machine learning methods in generalizing well. They are called *regularization methods* (Goodfellow et al., 2016).

One particular regularization method is known as *dropout*. Dropout was proposed by Srivastava et al. (2014) as an inexpensive method to support a better generalization. It mutes non-output neurons with a specified probability for the duration of an epoch and reactivates them afterward. On the downside, this technique reduces the model's effective capacity and therefore requires an increase in model size (Goodfellow et al., 2016).

3.3.5 Further Neural Network Based Models

Apart from the previously discussed machine learning models, there has been extensive research in the past on further models that are based on neural networks. Do et al. (2019) describe in their work several different types of models. Time-delay Neural Networks (TDNNs) and Convolutional Neural Networks (CNNs) are among those and are presented in the following section.

Time-delay Neural Networks are defined as FFNNs in which delayed inputs or states are utilized through the time-shifting approach. This allows the models to capture the temporal dynamics of the time series. Figure 10 illustrates this time-shifting approach. The neurons in the hidden layer receive not only the current input x_t but furthermore previous inputs such as x_{t-1} and x_{t-2}. Apart from the depicted architecture, other layers of the model also allow for these delays (Do et al., 2019).

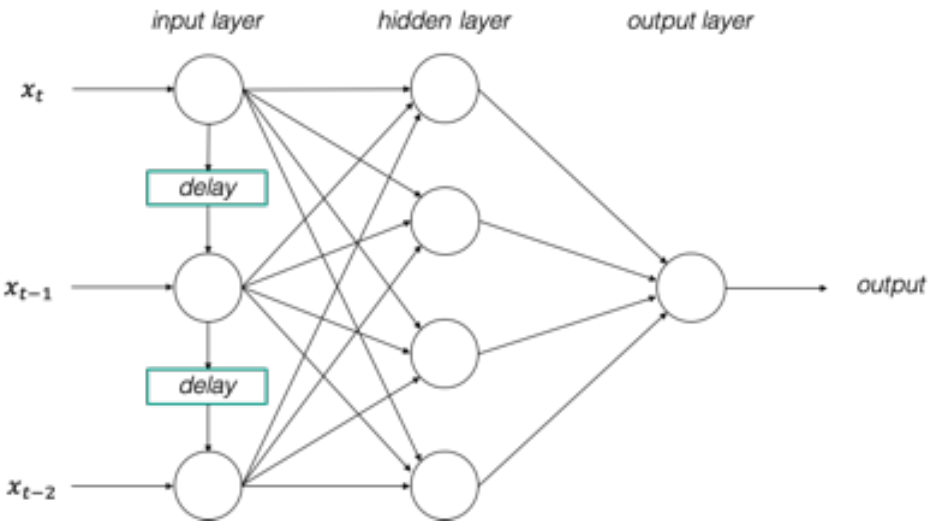


Figure 10: Time-delay Neural Network architecture, reprinted from Do et al. (2019)

TDNNs promise a simple way to represent correlations between past and present values in a feedforward model. They are easier to train compared to the RNNs described in the previous section. However, the fixed delays may not be suitable to capture temporal dynamics that change their behavior over time (Do et al., 2019).

Traditionally *Convolutional Neural Networks* have been successfully applied in the field of computer vision and image classification tasks. In the recent past, there have been promising applications to time series, for instance, in short-term traffic forecasting (Do et al., 2019). CNNs are another type of FFNNs and are designed to use minimal amounts of pre-processing compared to other deep architectures (Qiu et al., 2014).
Apart from input and output layers, CNNs include hidden layers such as convolution layers, pooling layers, or fully connected layers. The convolution layer convolves the input features with different kernels or filters to produce the output features. A pooling layer then subsamples the input either for further convolutional layers or eventually via a fully connected layer to the output layer (Do et al., 2019). The use of convolutions and filters suggests that this principle can have a positive influence on the forecasting performance for time series.

3.3.6 Strengths and Limitations of Machine Learning Methods

After introducing the fundamental principles and concepts of machine learning methods in the previous sections, this section will discuss their strengths and limitations in time series forecasting. ML-based methods have proven successful on several occasions in the recent past. Bandara et al. (2019) describe the successful application of Long Short-Term Memory Networks to sales demand forecasting in e-commerce. In their work, LSTMs deliver competitive results even compared to state-of-the-art statistical methods. Apart from demand forecasting, RNNs and CNNs have shown to be promising in supply chain planning (Bandara et al., 2019). As shown by Do et al. (2019), NN-based models furthermore are capable of forecasting highly dynamic traffic data. Hewamalage et al. (2019) conclude that RNNs can directly model seasonality, which makes deseasonalization redundant.

The major strength of machine learning methods is their flexible nonlinear modeling capability as they are capable of universally approximating almost any function. The model is adaptively formed based on the features presented from the data (Zhang, 2003). Thereby, ML algorithms identify complex nonlinear patterns and explore unstructured relationships without hypothesizing them forehand. Hence, machine learning methods are not limited by assumptions and allow the data to speak for itself (Smyl, 2020).
Statistical methods come with the disadvantage of treating each time series separately and forecast them in an isolated manner. For instance, in demand forecasting, the effects of related products that show similar sales demand patterns are neglected by the presented methods (Bandara et al., 2019). Exploiting this cross-series information is referred to as *cross-learning* and got increased attention in the recent past.

The basic principle is similar to the concept of transfer learning, where the transfer of knowledge from an already successfully learned task transferred to a new task achieves an improvement (Torrey and Shavlik, 2010). Cross-learning's underlying concept is that instead of exclusively developing one model for each time series in the setting, a global model is developed by exploiting information from many time series simultaneously (Hewamalage et al., 2019). Consequently, many series are utilized for training a single model. However, more effort in the preprocessing is required to learn properly across many time series (Smyl, 2020).

Data is a fundamental challenge for machine learning methods. As presented in Section 3.3.1, time series have to be framed into a supervised learning setting. This increases the data preprocessing requirements and reflects that none of the popular ML algorithms were created for time series forecasting (Smyl, 2020). Furthermore, the evidence that machine learning models struggle to generalize well from small datasets is regarded as a limitation compared to statistical methods (Cerqueira et al., 2019). While machine learning methods have the unique strength to universally approximate almost any function, they have to learn each relationship in the time series tediously from the presented data. This effort is a shortcoming over the statistical methods, where the incorporation of a priori knowledge simplifies the forecasting task. Thus, NN-based models need large and appropriate data sets, especially for deep architectures with numerous layers to ensure the accuracy of a forecast (Do et al., 2019). If the characteristics of a time series have changed over time, even a long time series may not contain enough relevant data to fit a complex model (Hewamalage et al., 2019). As data availability is often limited and regressors are not available in typical time series forecasting problems, the performance of machine learning methods, therefore, tends to be below expectations (Makridakis et al., 2018b).

As elaborated in the previous sections, NNs have many design parameters that can be adapted, such as the configuration or the type of activation function. All of them must be tuned to achieve the best performance for a particular dataset, which can result in high tuning effort (Do et al., 2019). As an NN's performance depends on various factors, it can not be generalized that deep NNs are better than ones with fewer layers, mainly because the amount of relevant data is of such crucial importance.

Furthermore, machine learning methods are criticized for their black-box nature, which makes the explainability of the forecast results more difficult (Makridakis et al., 2018b). Statistical methods provide a more straightforward and comprehensible procedure (Hewamalage et al., 2019).

Finally, machine learning methods are computationally demanding, especially compared to statistical methods (Makridakis et al., 2018b). Do et al. (2019) report that

the training of deep NNs for large-scale traffic network can require several weeks. As Makridakis et al. (2018b) state, in order to advance time series forecasting, machine learning methods need to become more accurate, require less computational time, and be less of a black-box.

The key message of this chapter is that machine learning methods can be useful instruments for time series forecasting. Unfortunately, they bring several limitations to the table that inhibit excellent performance. The following chapter seeks to find a solution to these limitations and an improvement in forecasting accuracy.

4 Hybrid Methods for Time Series Forecasting

So far, this thesis discussed two distinct approaches for time series forecasting. The following chapter is devoted to a core topic of this work, which is the hybrid approach. First, an introduction to the topic is presented that defines terminology and points out potential benefits. After a look at relevant research, it elaborates on a successful example that attracted considerable attention.

4.1 Essentials of Hybrid Methods

Hybrid methods have to be distinguished from ensemble methods. *Ensemble methods* are simplistic combinations of forecasts. The forecast combination is defined by

$$u = \sum_{m=1}^{N} w_m u\,(m),$$

where $u\,(m)$, $m = 1, ..., N$, are the N forecasts to be combined (Atiya, 2020). The variable w_{m} denotes a combination weight. The underlying principle is similar to risk diversification, as the combination of forecasts conducted by several models hedges against the resulting inaccuracy (Atiya, 2020). Furthermore, Krogh and Vedelsby (1995) show that ensembling several regression systems significantly improves the generalization of any regression system. The same concept finds use in other areas such as Random Forests. Random Forests take the output of multiple decision trees and consolidate their outputs into a single one (Breiman, 2001a).

In the forecasting context, *hybrid methods* are understood as a sophisticated combination of statistical and machine learning methods that interact with each other. This combination is a different approach than consolidating isolated forecasts and separates hybrid methods from the ensemble methods.

As Chapters 3.2.7 and 3.3.6 demonstrated, statistical and machine learning methods have distinctive strengths in time series forecasting. Unfortunately, neither of these methods is flawless. Table 2 summarizes the strengths and limitations of both approaches.

	Strengths	Limitations
Statistical methods	+ Simplicity, comprehensibility + A priori knowledge + Effectiveness with limited data availability	- Assumption of linearity - Missing cross-learning
Machine learning methods	+ No assumption of linearity + Universal approximation + Cross-learning	- Data requirements - Computational effort

Table 2: Strengths and limitations of statistical and machine learning methods

The fundamental idea of hybrid methods is that the combination of statistical and machine learning methods compensates for the limitations of one approach with the strengths of the other. For instance, the effectiveness with limited data availability of statistical methods can counteract the extensive data requirements of machine learning methods. Apart from that, the consideration of a priori knowledge can simplify the expected forecasting task and decreases the computational effort. Furthermore, hybrid methods can incorporate cross-learning, a capability that the presented statistical methods lack. Finally, the hybrid methods provide a solution for the dilemma of the assumption of linearity. Real-world time series may be purely linear, purely nonlinear, or often contain a combination of those two patterns (Panigrahi and Behera, 2017). Although ANNs successfully overcome the drawback of ARIMA models in nonlinear relationships, they have produced mixed results for purely linear time series. Thus, neither ARIMA nor ANNs are solely sufficient to model a real-world time series (Khandelwal et al., 2015). Consequently, hybrid methods promise to combine the best of both worlds to advance time series forecasting.

Hybrid methods were studied intensively in past years, and a substantial number of papers were published. The following section presents relevant research by discussing selected models and demonstrates how the models realize the hybridization.

4.2 Relevant Research

Research on hybrid methods is not a novelty, as the work began already almost twenty years ago. The basic idea was developed by Zhang (2003), who proposed a hybridization of ARIMA and MLP. The underlying principle is that the MLP learns the deviation of the ARIMA prediction from the actual value and seeks to adjust it to obtain a more accurate result. This methodology is plausible because a time series is composed of a linear autocorrelation structure and a nonlinear component, which can be described as follows (Zhang, 2003):

$$y_t = L_t + N_t$$

L_t thereby denotes the linear component, and N_t denotes the nonlinear component.

The ARIMA is fitted to capture the linear component, so consequently, the residuals from the linear model account for the nonlinear relationship (Zhang, 2003). The residuals e_t are computed by subtracting the ARIMA predictions $\hat{L}_t$ from the actual values y_t (Zhang, 2003):

$$e_t = y_t - \hat{L}_t$$

In the next step, the MLP takes the past residuals as input to learn a function $f(e_{t-1}, e_{t-2}, ..., e_{t-n})$ that can be used to forecast the residuals. $\hat{N}_t$ denotes this prediction. Finally, the hybrid forecast $\hat{y}_t$ is obtained by adding the predictions of both models (Zhang, 2003):

$$\hat{y}_t = \hat{L}_t + \hat{N}_t$$

Zhang (2003) shows that the hybrid forecast yields promising results. The application to three real-world time series delivered considerably better forecasting accuracies compared to the isolated models. However, as this model does not deploy general diagnostic statistics for nonlinear relationships, it might still not be adequate in modeling nonlinear relationships appropriately (Zhang, 2003).

Based on these foundations, several authors have proposed modifications and extensions to this model. Khandelwal et al. (2015) seek to tackle the lack of diagnostic statistics for nonlinear relationships by deploying a Discrete Wavelet Transform (DWT) to the ARIMA-MLP hybridization. The DWT separates the time series into linear and nonlinear components. Then, the ARIMA is fitted to the linear part, and forecasts are computed. After that, the MLP is fitted to the corresponding residuals together with the nonlinear part provided by the DWT. Finally, the hybrid forecast is obtained by adding these two forecasts (Khandelwal et al., 2015). The authors show that their approach outperforms the standalone models as well as the former model by Zhang (2003).
Domingos et al. (2019) also pick up the basic principle but focus on the way the linear and nonlinear predictions are joined. Therefore, they use a data-driven approach and a machine learning model to generate the final output. The fundamental principle is to find the most suitable function, which describes the relationship between the forecast of the time series and the forecast of the residual series. Support Vector Regression (SVR), another supervised method for nonlinear mappings, delivered the most promising results (Domingos et al., 2019).

Another modification is to change the components that form the model. Panigrahi and Behera (2017) already proposed a hybridization of exponential smoothing and MLP, but the breakthrough was a model introduced by Smyl (2020). His combination of exponential smoothing and Long Short-Term Memory Networks caused a great stir among forecasters. The next section is dedicated to his model and discusses it in detail.

4.3 Slawek Smyl's Hybrid Method

The presented model was the winning submission to the M4 Competition (Makridakis et al., 2018a). Before discussing the model in detail, this section briefly describes the contest and its terms and conditions.

The *M4 Competition* is a forecasting contest and follows on from the three previous M competitions. Their purpose is to learn from empirical evidence how forecasting accuracy can be improved, and how these learnings advance the theory and practice of forecasting (Makridakis et al., 2020).
The competitors developed and submitted models to forecast the M4 dataset. This dataset consists of 100,000 individual time series from different domains, such as microeconomic, financial, or demographic backgrounds. Apart from that, the time series in the dataset show different time intervals between the successive observations. Hence, the data has both high-frequency data, like weekly, daily, hourly, and low-frequency data, such as yearly, quarterly, monthly (Makridakis et al., 2020). Table 3 shows the domains and respective frequencies and gives an overview of the distribution of the M4 dataset.

Frequency	Micro	Industry	Macro	Finance	Demographic	Other	Total
Yearly	6,538	3,716	3,903	6,519	1,088	1,236	23,000
Quarterly	6,020	4,637	5,315	5,305	1,858	865	24,000
Monthly	10,975	10,017	10,016	10,987	5,728	277	48,000
Weekly	112	6	41	164	24	12	359
Daily	1,476	422	127	1,559	10	633	4,227
Hourly	0	0	0	0	0	414	414
Total	**25,121**	**18,798**	**19,402**	**24,534**	**8,708**	**3,437**	**100,000**

Table 3: Number of M4 time series per data frequency and domain, reprinted from Makridakis et al. (2020)

The performance of the forecasts is evaluated according to the Overall Weighted Average (OWA) of two accuracy measures, Mean Absolute Scaled Error (MASE) and the symmetric Mean Absolute Percentage Error (sMAPE). The respective calculation is described in Appendix A.1.
The results of the M4 Competition showed some remarkable results. First, it confirmed that ensemble methods outperformed approaches relying solely on statistical or machine learning methods. Especially ensembles of statistical methods were among the top-performing solutions (Makridakis et al., 2020). Furthermore, the results supported the hypothesis that state-of-the-art machine learning methods do not outperform statistical methods. The submitted pure ML-based methods delivered a poor performance, with only one beating the seasonal naive baseline (Makridakis et al., 2020). The surprising winner was Slawek Smyl, who provided a hybrid so-

lution that outperformed all other methods (Makridakis et al., 2020). His success demonstrates the enormous potential of hybrid methods for time series forecasting.

In his method, Slawek Smyl combines exponential smoothing and Recurrent Neural Networks to a *hybrid HW-LSTM model*. An LSTM produces a trend forecast for the exponential smoothing, which is a sightly modified Holt-Winters' Method (Smyl, 2020).

In general, the method is a hybrid and hierarchical method. It is *hybrid*, as it joins statistical with machine learning methods. The Holt-Winters' Method handles the seasonality of the time series, while the LSTM cares for the non-linearity and makes use of cross-learning (Smyl, 2020).
Hierarchical describes the fact that the model has local and global parameters. Thereby, local parameters reflect the behavior of a particular time series, whereas global parameters relate to a broad set of time series (Smyl, 2020). Local constants are, for instance, the smoothing coefficients of the HW or the initial seasonal components. Apart from local constants, there are local states such as the level or seasonal components. These local states reflect the behavior of a single time series, but contrary to local constants, change over time. On the other hand, an example of a global constant is the weights of the RNN, which are learned across a broad set of time series (Smyl, 2020).

Apart from the exponential smoothing and Recurrent Neural Network, further elements complete the hybrid method. Figure 11 visualizes the architecture and contextualizes the method's main four components.

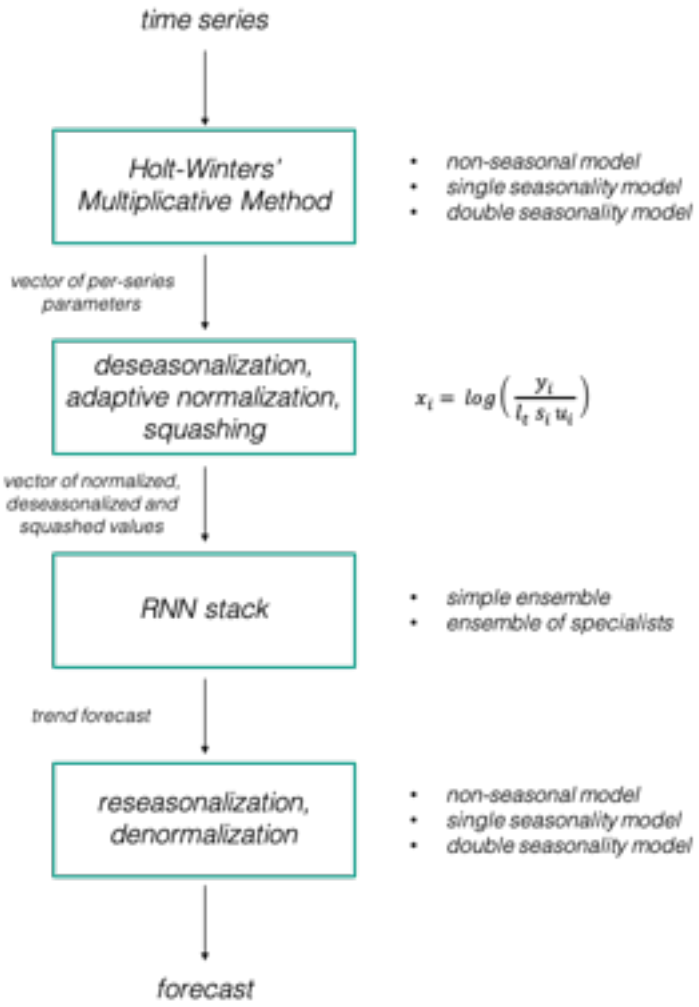


Figure 11: Architecture of Slawek Smyl's hybrid HW-LSTM model

First, the data points of the time series are fed into the component that deploys the Holt-Winters' Multiplicative Method, where each time series is handled differently depending on its frequency. There are three types of models: non-seasonal models for yearly and daily data, single seasonality models for monthly, quarterly, and weekly data, and double seasonality models for hourly data (Smyl, 2020). The corresponding formulas for each case are defined by (Smyl, 2020):

Non-seasonal model:

$$l_t = \alpha y_t + (1 - \alpha)\, l_{t-1}$$

Single seasonality model:

$$l_t = \alpha \frac{y_t}{s_t} + (1 - \alpha)\, l_{t-1}$$

$$s_{t+K} = \beta \frac{y_t}{l_t} + (1 - \beta)\, s_t$$

Double seasonality model:

$$l_t = \alpha \frac{y_t}{s_t u_t} + (1 - \alpha)\, l_{t-1}$$

$$s_{t+K} = \beta \frac{y_t}{l_t u_t} + (1 - \beta)\, s_t$$

$$u_{t+L} = \gamma \frac{y_t}{l_t s_t} + (1 - \gamma)\, u_t$$

y_t is the value of the time series at time t, whereas l_t, s_t, and u_t are the level, seasonality, and second seasonality component, respectively. K denotes the number of observations per seasonal period, for instance, 12 for monthly or 52 for weekly data. Finally, L is the number of observations per second seasonal component in hourly data, 168. For all models applies $s_t,\ u_t \geq 0$ and $\alpha, \beta, \gamma \in [0, 1]$. The smoothing coefficients are fitted by SGD. These formulas allow for the estimation of the level and seasonality components for all points of the series and form a vector of per-series parameters. This vector is processed in the second component (Smyl, 2020).

The second component is responsible for deseasonalization, adaptive normalization, and squashing. It processes the input window and output window values, which are fed to the RNN stack. To do so, it takes the vector of the per-series parameters as input. The deseasonalization and adaptive normalization of the vector's scalar components are realized by the following formulas (Smyl, 2020):

Non-seasonal model:

$$x_i = \frac{y_i}{l_t}$$

Single seasonality model:

$$x_i = \frac{y_i}{l_t s_i}$$

Double seasonality model:

$$x_i = \frac{y_i}{l_t s_i u_i}$$

Thereby y_i is the value of the time series in the input window at the point i, l_t the last value of the level of the input window, and s_i and u_i are the seasonality components at the time i.

The division of y_i by the respective seasonality components deseasonalizes the data. Therefore, only the trend component and the error remains in x_i. By further dividing y_i by the last value of the input window, the data is normalized. Smyl (2020) points out that this adaptive normalization yields two significant advantages. First, it overcomes the shortcoming that when a series is globally normalized, some values are ignored. For example, the parts of the series with small values will be neglected for time series that change massively in value over their lifespan. Furthermore, information on the trend's strength is lost. For instance, two time series of the same

length and similar shapes, but one developing from 100 to 110 and another from 100 to 200 look very similar after the normalization to an $[0, 1]$ interval (Smyl, 2020).
The following rule determines the size of the input window: for seasonal series, it should cover at least on seasonal period. For example, in the case of monthly data, at least 12 past values are part of the input window. For non-seasonal series, the size of the input window is based on the forecasting horizon. This rule results in the input and output values being close to 1, regardless of the original amplitude of the series or its history (Smyl, 2020).
The application of a squashing function completes this second component. The function of choice is a logarithm. The logarithm prevents potential outliers from having an overly disturbing effect on learning (Smyl, 2020). Furthermore, the hybrid model will not forecast an actual value but the difference between consecutive values. Thus, the overall formulas are defined as follows (Smyl, 2020):

Non-seasonal model:

$$x_i = \log\left(\frac{y_i}{l_t}\right)$$

Single seasonality model:

$$x_i = \log\left(\frac{y_i}{l_t s_i}\right)$$

Double seasonality model:

$$x_i = \log\left(\frac{y_i}{l_t s_i u_i}\right)$$

In conclusion, the values of x_i form X_i, which is a vector of normalized, deseasonalized, and squashed values. This vector is the input for the RNN stack.

As previously mentioned, the RNN's task is to forecast a trend that is most likely non-linear (Smyl, 2020). The RNN architecture is built up of dilated LSTM-based stacks. As stated in Section 3.3.4, dilated LSTMs endeavor to improve long-term memory performance. The stacks are followed by a linear adapter layer, which adapts the size of the state of the last layer to the size of the forecasting horizon (Smyl, 2020).
The stacks are composed of several blocks. These blocks are a sequence of one to four layers that belong to one of three types of dilated LSTMs. The three types are a standard dilated LSTM (Chang et al., 2017), dilated LSTM with an attention mechanism (Qin et al., 2017), and a special residual dilated LSTM (Kim et al., 2017).
Table 4 gives an overview of the deployed RNN architecture and shows that a different stack is used for each frequency. The number of bracket pairs indicates the number of blocks, whereas the quantity of numbers inside the bracket denotes the number of individual layers. The value of the respective numbers shows the dilation of the layer (Smyl, 2020). In the case of quarterly data, the architecture (1,2)-(4,8)

Standard is used. Standard stands for the standard dilated LSTM while (1,2)-(4,8) reflects two blocks. The two blocks each consist of two layers with dilation of 1 and 2, and 4 and 8, respectively. The last part of the entry indicates whether the RNN stack applies the ensemble of specialists. This technique is presented in detail later in this section.

Frequency	**Architecture**
Yearly	(1,6) Attentive, ensemble of specialists
Quarterly	(1,2)-(4,8) Standard, simple ensemble
Monthly	(1)-(3)-(6)-(12) Residual, simple ensemble
Weekly	(1,52) Attentive, ensemble of specialists
Daily	(1,3)-(7,14) Standard, ensemble of specialists
Hourly	(1,4)-(24,168) Standard, ensemble of specialists

Table 4: Overview of the RNN architecture, adapted from Smyl (2020)

The dilation indicates the point in time whose hidden state is considered. In the case of dilation of 2, the model takes the hidden state from $t-2$ as input and passes its updated hidden state to $t+2$. Smyl (2020) chooses the dilations consistent with the seasonal periods of the frequency of the time series. For instance, for monthly data, the dilations are 3, 6, and 12. These dilations reflect three possible seasonality periods: one of three months, one of half a year, and a full-year seasonal period. This procedure emphasizes the relevance of the lagged values and intends to better support the RNN in learning the seasonal effects. Consequently, it creates an artificial weighted average state that allows the RNN to focus on a particular group of past states dynamically (Smyl, 2020).
Figure 12 shows exemplarily the structure of the RNN stack, which is used for the quarterly time series. The solid grey box encloses one stack, while the dashed grey box represents the respective blocks. The shortcut between the two blocks is a Resnet-style shortcut and adds one block's output to the next (Smyl, 2020). Additionally, an alternative representation of this RNN stack is given in Appendix A.2.

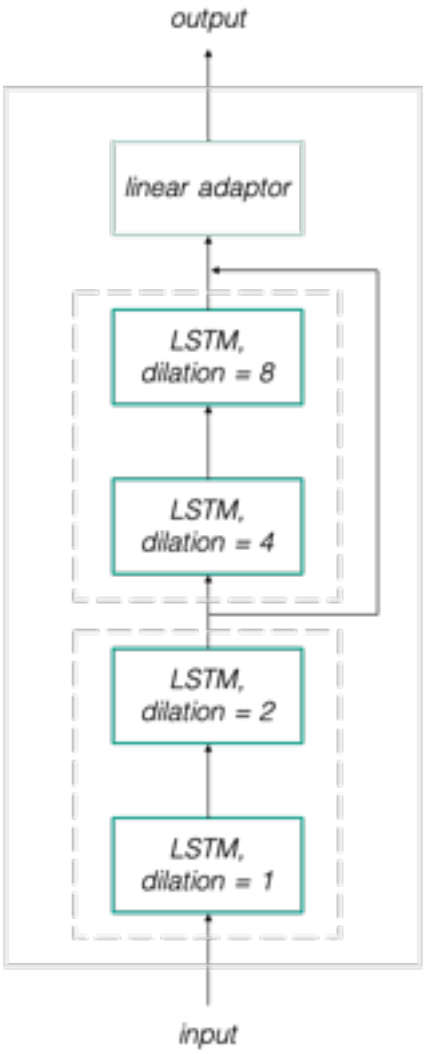


Figure 12: Dilated LSTM stack: (1,2)-(4,8) Standard for quarterly time series, adapted from Smyl (2020)

There is one last technique Smyl (2020) applies in the RNN stack: an *ensemble of specialists*. This technique was presented a few years earlier (Smyl, 2017) and is part of his winning submission to the M4 Competition.
The author states that when a dataset contains a large number of time series from unknown sources, which is the case in the M4 Competition, the series can be grouped into subsets. Consequently, individual models can be applied to these subsets, instead of using a single model for the whole dataset, which leads to improved overall forecasting accuracy (Smyl, 2020). With this context in mind, the fundamental idea of the ensemble of specialists is to train several models concurrently and force them to specialize in a subset of the series (Smyl, 2020).
To deploy an ensemble of specialists, M models are created first, and a subset of the dataset (e.g., half of it) is randomly allocated to each model. A single training is then executed on the allocated dataset, and the in-sample performance of each model is recorded. In the next step, the models are ranked in terms of their

performance for each series in the dataset. Finally, each time series is allocated to the top best N models. This procedure continues until the error on the validation set is increasing. The final forecast for a particular time series is calculated by the average of the forecasts conducted by the top N models (Smyl, 2020). Therefore, this technique of clustering the time series in the dataset and mapping them to the best performing models is an unsupervised learning task.
As Table 4 shows, the method does not apply the ensemble of specialists for every frequency due to computational reasons. For monthly and quarterly data, the method applies a simple ensemble (Smyl, 2020). Its underlying principle is bootstrap aggregating or shortly termed *bagging*. Bagging might not be as sophisticated as the ensemble of specialists but is still adequate to improve forecasting accuracy (Breiman, 1996).

Finally, the method's last component performs the reverse transformation, which includes reseasonalization and denormalization and obtains the actual forecast. The following formulas reverse transform the trend forecast from the RNN stack respective the frequency of time series (Smyl, 2020):

Non-seasonal model:

$$\hat{y}_{t+1..t+h} = exp\left(NN\left(x\right) \times l_t\right)$$

Single seasonality model:

$$\hat{y}_{t+1..t+h} = exp\left(NN\left(x\right) \times l_t \times s_{t+1..t+h}\right)$$

Double seasonality model:

$$\hat{y}_{t+1..t+h} = exp\left(NN\left(x\right) \times l_t \times s_{t+1..t+h} \times u_{t+1..t+h}\right)$$

$NN\left(x\right)$ denotes the output vector of the RNN stack, whereas h is the forecasting horizon and l_t the last value of the input window. The exponential function reverse transforms the squashing.

Slawek Smyl's hybrid HW-LSTM method caused a sensation by winning the M4 Competition, and some researchers are confident that hybrid methods have a great future (Makridakis et al., 2018a). After presenting the method in detail, the core question that remains unanswered is why it works so well. The method applies sophisticated deep learning techniques, but the Holt-Winters' Method accounts for a significant part of the work. A possible hypothesis is that the HW supports the LSTM to play its strengths more effectively. The presented strengths of machine learning methods, especially cross-learning and nonlinearity, result in an excellent modeling power. However, this modeling power comes at the cost of requiring appropriate data. The HW could reduce the training effort by taking out the seasonality and leaving the LSTM a more straightforward task, where the data requirements are

mitigated. In addition to the research question, the following experimental design seeks to find evidence to support this hypothesis and aims to evaluate the model's superior performance in the M4 Competition in another setting.

5 Experimental Setup

The following chapter describes the design of the experimental setup. It starts by exhibiting the overall process and then presents the considered time series with their distinguishing characteristics. Furthermore, it elaborates on the utilized methods and accuracy measures. This chapter closes with details on the experiment's implementation.

5.1 Course of Investigation

The experiment's objective is to consider various time series with different characteristics in order to investigate the influence of these characteristics on the performance of the forecasting method. The investigation is divided into three steps, which Figure 13 displays. The first step is to identify several time series with distinguishing characteristics and find meaningful clusters. Then, individual forecasts are conducted applying different methods. Apart from statistical and machine learning approaches, the hybrid HW-LSTM method is used. Finally, the forecasts are evaluated with relevant error metrics to assess their performance on the different time series.

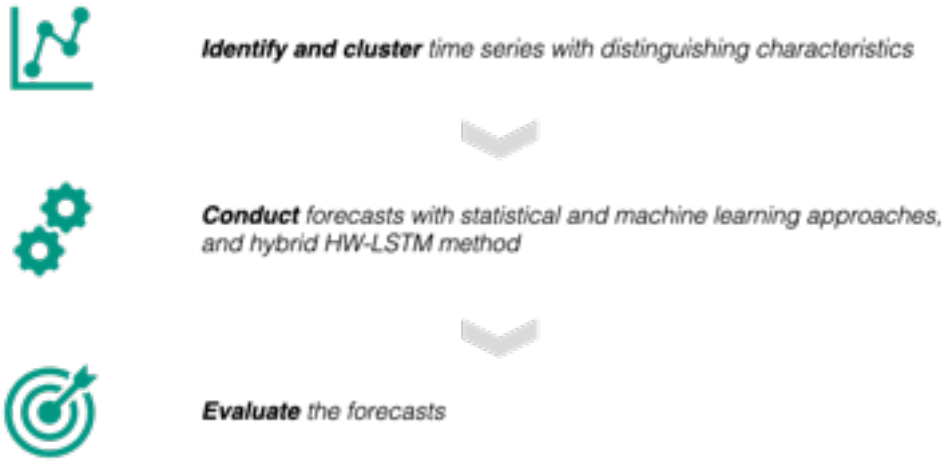


Figure 13: Course of investigation

The goal of this course of the investigation is to evaluate the presented strengths and limitations of both statistical and machine learning methods. Furthermore, it enables the validation of the outstanding performance of the hybrid HW-LSTM method in another setting and allows for the conclusion of the research question.

5.2 Identification and Clustering of Time Series

This section describes the identification of relevant datasets and the clustering. As the time series exhibit different characteristics, a detailed analysis of each is necessary to understand the experiment's input thoroughly.
The considered univariate time series are recognized in the literature and obtained from freely available sources. This section analyzes the time series regarding the following characteristics: stationarity, homogeneity of variance, trend, seasonality and cycle, and the number of observations.

To analyze the stationarity of the eligible time series, the *Augmented Dickey-Fuller Test (ADF)* is used. As set out in Section 3.1.1, it tests against the null hypothesis of the presence of a unit root in the data. In the first run, the ADF investigates whether the time series is stationary. When the result shows nonstationarity, the time series is differenced, and the ADF is applied anew to discover if the time series is trend stationary. If the second ADF reveals the presence of a unit root, the time series is considered not stationary. Using differencing instead of detrending has the advantage that no parameters need to be estimated in that operation and is reasonable to coerce the data to be stationary (Shumway and Stoffer, 2017).
The *Levene's Test for equal variances* reveals if a time series is homoscedastic or heteroscedastic. As the test compares subsets of a time series in terms of their disturbance term's variance, the breakdown into the respective subsets influences the test result. Therefore, the time series is split into two, three, four, five, and ten subsets, and the results of each test are consolidated.
The experiment performs a decomposition and examines the time series in terms of trend, seasonality and cycle, and residuals. As the applicability of the decomposition methods depends on whether a multiplicative relationship underlies a time series, two approaches are practiced. Whenever the *Seasonal and Trend Decomposition using Loess (STL decomposition)* is applicable, it is the method of choice. However, when a time series clearly shows multiplicative behavior, the STL decomposition can not be applied, and a *classical decomposition* is executed. Apart from the decomposition, the autocorrelation of a time series is analyzed via the *Autocorrelation Function (ACF)*. The ACF visualizes the autocorrelation and provides information on the correlation between particular lagged values.

In total, nine real-world time series are identified as suitable and considered in the experiment. The following paragraphs introduce these time series in detail and show the results of the conducted tests. For the better readability of this section, the illustrations of the decomposition are located in Appendix A.3 and the ACF plots in Appendix A.4. Further details on the results of the ADF and the Levene-Test, as well as the first-order difference, are included in Appendix A.5, Appendix

A.6, and Appendix A.7, respectively.

The *Airline Passengers* dataset describes monthly passengers on an airline and consists of 144 observations. The ADF indicates clearly that the time series does not follow a stationary process. Furthermore, the Levene-Test shows heteroscedasticity. As Figure 14 depicts, the amplitudes increase over time, which suggests a multiplicative relationship. Therefore, a multiplicative classical decomposition is performed. The results show a strong yearly seasonality and an almost linear, positive trend. The ACF supports this, as the values show a strong positive correlation to the respective value 12 months ago. The Airline Passengers dataset was acquired from Kaggle (Kaggle, nda).

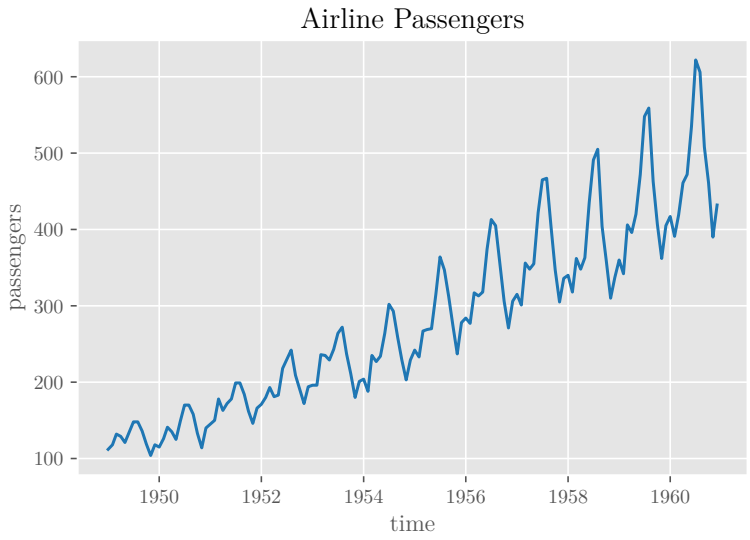


Figure 14: Airline Passengers

The number of lynx trapped per year in the Mackenzie River District of North-West Canada is the subject of the *Canadian Lynx* time series. The time series includes data from 1821 to 1932 and therefore has 112 observations. Figure 15 depicts the data and suggests a periodicity of approximately ten years. The ACF confirms this hypothesis, as apart from the previous value, the value ten years ago shows a significant positive correlation. The Canadian Lynx has been extensively studied in the literature with a focus on nonlinear modeling (Zhang, 2003). In terms of trend and residuals, the STL decomposition indicates no clear trend but little noise and single outliers. Although the Levene-Test indicates homoscedasticity, the ADF result is that the Canadian Lynx is a nonstationary time series. This is not a contradiction, as homoscedasticity is a necessary, but not sufficient criterion for stationarity. The data was obtained from GitHub (GitHub, nda).

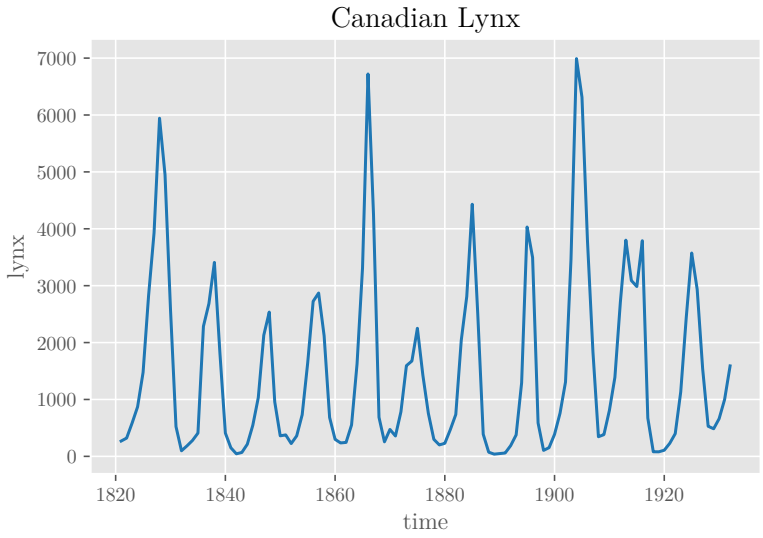


Figure 15: Canadian Lynx

The next time series is *Daily Minimum Temperatures Melbourne*. As the name suggests, it describes the daily minimum temperature recorded in Melbourne, Australia. It contains daily observations of ten years, ranging from January 1981 to December 1990. The time series is both heteroscedastic and nonstationary, as the pertinent tests confirm. The ACF shows that Melbourne experiences the change of the seasons, although the summer and winter months are reversed to the ones in Europe due to the city's location in the southern hemisphere. Hence, the time series has a yearly seasonality. Apart from that, the ACF reveals a strong correlation to the previous value. This strong correlation is explained by the fact that temperature is inert and obeys the laws of physics. As extreme temperature drops are seldom, today's temperature has a significant influence on tomorrow's temperature. The analysis of the STL decomposition exhibits no clear trend but a moderate range of residual values. Figure 16 presents the dataset that was acquired from Kaggle (Kaggle, ndb).

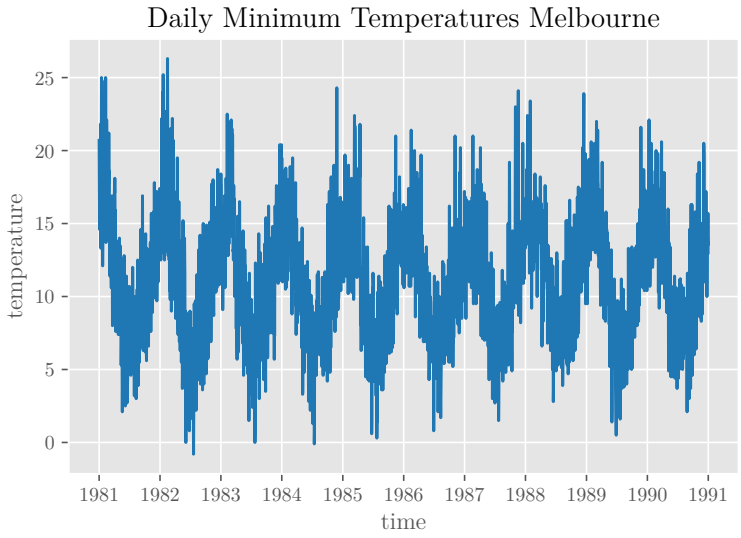


Figure 16: Daily Minimum Temperatures Melbourne

The *Daily Total Female Births California* dataset has 365 observations and represents the daily number of female births of 1959 in California, USA. The ADF and Levene-Test indicate that the time series is nonstationary and homoscedastic. The time series reveals a slight significant positive correlation to values from 7 and 21 days ago, suggesting a weak weekly seasonality. In terms of trend and residuals, there is no distinct trend observable, and the STL decomposition shows a moderate noise in the data as the residuals have a moderate range. Figure 17 depicts the dataset which originates from Kaggle (Kaggle, ndc).

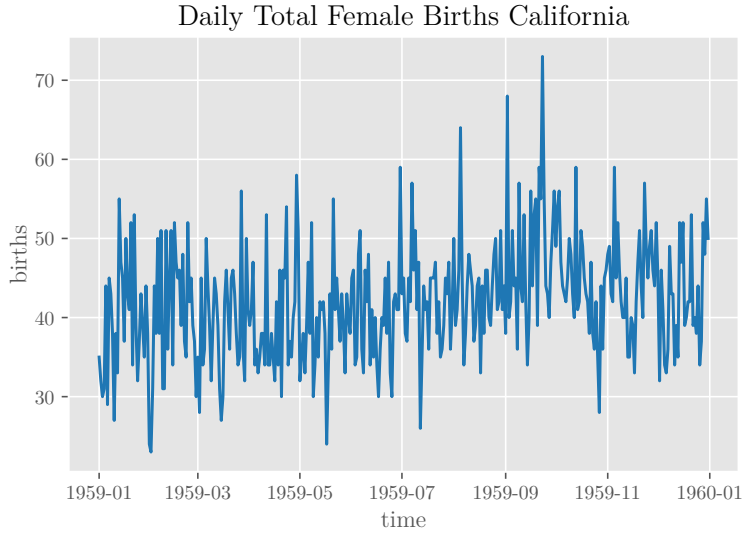


Figure 17: Daily Total Female Births California

A challenging task is the forecast of exchange rates as the observations rely on various influencing factors. The *GBP USD Daily Exchange Rate* incorporates this demanding task in the experiment with its numerous turning points. As Figure 18 shows, the dataset contains 5977 observations between January 1993 and May 2018, reflecting the daily exchange rate of the pound sterling (GBP) in currency units per U.S. dollar (USD). While the dataset is heteroscedastic, the ADF on the first-order difference in the time series reveals that the GBP USD Daily Exchange Rate is trend stationary. The ACF shows a strong short-term correlation, and the STL decomposition exhibits neither a clear trend nor seasonality. The dataset was collected from Kaggle (Kaggle, ndd).

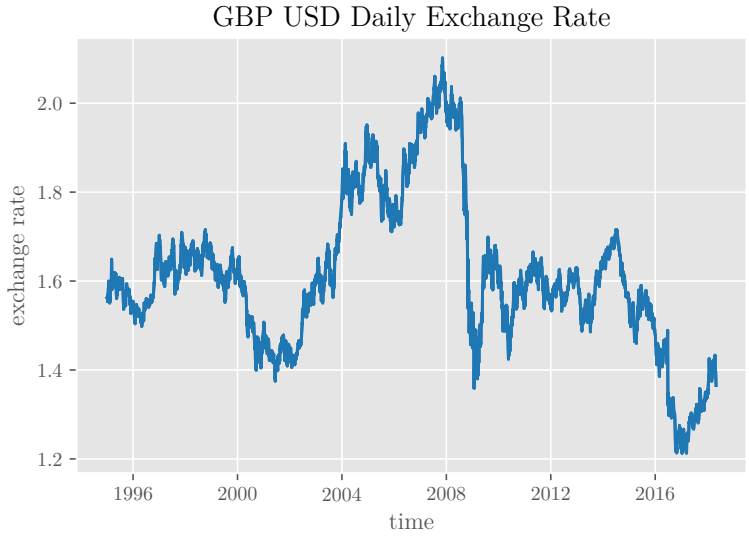


Figure 18: GBP USD Daily Exchange Rate

The next time series measures the real output of all relevant electric and gas establishments in the United States and originates from the Federal Reserve Bank of St. Louis (Board of Governors of the Federal Reserve System (US), nd). For better readability, it is shortly referred to as *Industrial Production*, and it describes the monthly industrial production index in which values are relative to a base unit and are not seasonally adjusted. Figure 19 shows its 84 observations between January 2007 and December 2013. As the ADF result indicates stationarity, it is not surprising that the STL decomposition reveals neither trend nor seasonality. However, the ACF suggests a strong negative correlation and a strong positive correlation between lagged values, which indicates a cyclic behavior. As Hyndman and Athanasopoulos (2019) state, time series with cyclic behavior without trend or seasonality can be stationary. The authors argue that when the mean and the variance remain constant over time respective to a significance level, the time series is stationary. Constant mean and variance appear to be the case for the Industrial Production dataset, as

the STL decomposition does not indicate an inconstant trend, and the Levene-Test confirms constant variance by indicating homoscedasticity. Therefore, the result of the ADF seems credible, and this time series is considered stationary.

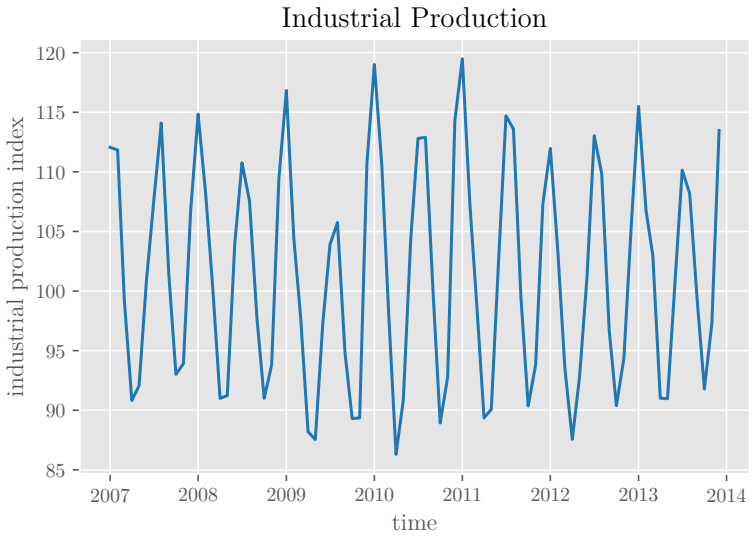


Figure 19: Industrial Production

The *Rossmann Store Sales* dataset originates from a Kaggle forecasting competition (Kaggle, nde). The dataset includes the daily store turnover for 1,115 stores of the German retail chain Rossmann. In the experiment, a single store is chosen, and its sales from January 2013 until July 2015 are considered. Thereby the data is adjusted by the days the store was closed, which is the case for Sundays and state holidays. The Sundays were removed from the time series to avoid that the forecasting models have to predict a value of zero. The values on weekday state holidays are imputed by the mean of the same weekday one week before and after the holiday. After this careful preprocessing, the time series has 808 observations.
The ADF indicates that the time series is not stationary, while the Levene-Test delivers mixed results. For the smaller sample sizes, the result indicates homoscedasticity, but for larger sample sizes, the result is the opposite. As a higher number of subsamples leads to the fact that the variance has to be equal for more subsets concurrently, its result tips the scales and, therefore, Rossmann Store Sales is regarded as heteroscedastic. Figure 20 reveals two peaks in the data that reflect the Christmas business. As the time series is likely to underlie several influencing factors, the STL decomposition has difficulties giving a precise picture of trend and seasonality. In terms of residuals, however, it indicates a noisy behavior. For this time series, it is worthwhile to analyze two ACF visualizations to consider short-term and long-term relationships. Although, these analyses have to been seen in the light of the removed values. The removed values cause the actual periods to appear shorter in

the graphic than in reality. Thus for the long-term ACF, the positive correlation of a particular value with values around three hundred days ago indicates a yearly seasonality. The ACF for short-term relationships reveals further significant positive correlations. All in all, this suggests multiple overlaying seasonalities and presents a difficult task for the forecasting methods.

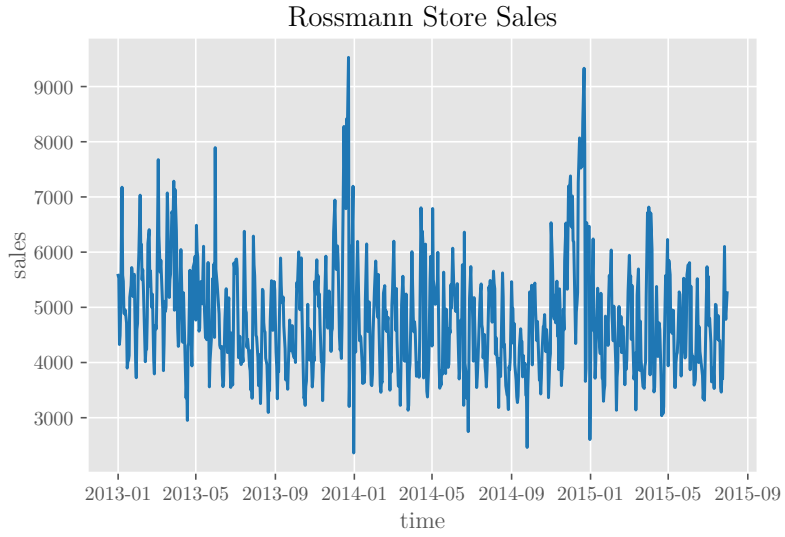


Figure 20: Rossmann Store Sales

The *Sunspot* dataset describes the annual activity of spots visible on the sun and is of practical importance to geophysicists, environmental scientists, and climatologists (Hipel and McLeod, 1994). Figure 21 displays the 289 observations that range from 1700 to 1988. The data is regarded as nonlinear in the literature and used to evaluate the effectiveness of nonlinear models (Zhang, 2003). While the time series is heteroscedastic, the ADF on the first-order difference exhibits trend stationarity. However, ACF indicates a cycle of approximately 11 years. Similar to Industrial Production, this does not contradict the trend stationary process of the time series. In terms of noise, the STL decomposition reveals a moderate level of noise in the data. The time series is broadly used in the literature and was obtained from GitHub (GitHub, ndb).

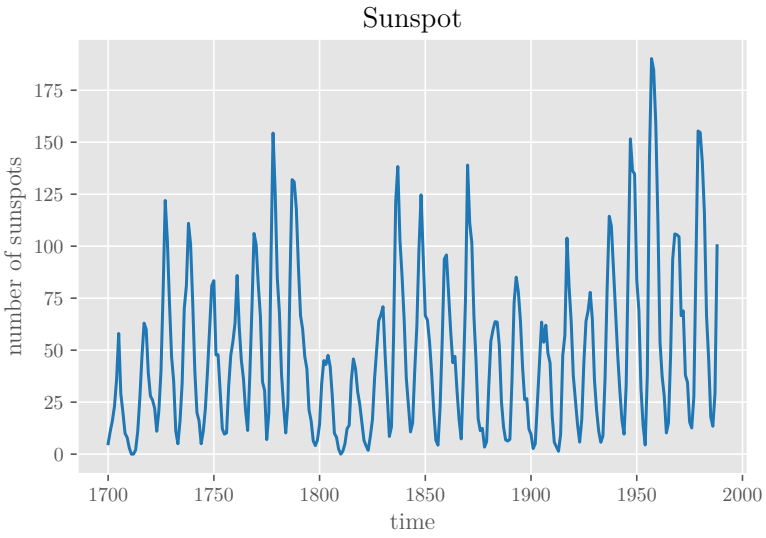


Figure 21: Sunspot

Finally, the last time series considered in the experiment is a *Random Walk* with 1000 daily steps. Its objective is to confront the forecasting method with randomness that can not be reasonably predicted. The Random Walk is constructed using a random variable that determines whether 1 is added or subtracted to the current value of the time series. Figure 22 shows the resulting course. Per definition, this Random Walk is nonstationary (Hyndman and Athanasopoulos, 2019).

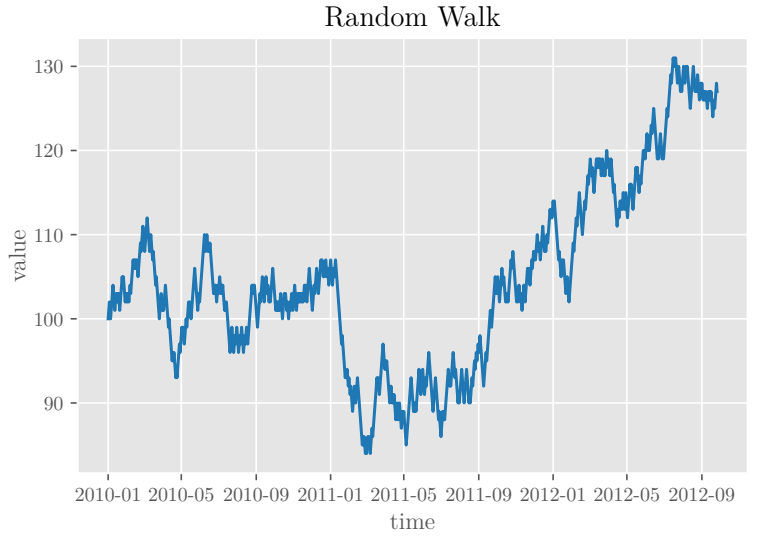


Figure 22: Random Walk

The presented analysis of the nine time series suggests the division into three clusters: *stationary*, *trend stationary*, and *nonstationary time series*. Table 5 shows the classification and reviews the characteristics of the time series elaborated in this section.

Cluster	Time series	Frequency	Length	Homogeneity of variance	Trend	Seasonality/ Cyclicity	Residuals
Stationary	Industrial Production	Monthly	84	Homoscedastic	None	Cycles of approx. 6 months	Minimal noise
Trend stationary	GBP USD Daily Exchange Rate	Daily	5977	Heteroscedastic	None	None	Minimal noise
	Sunspot	Yearly	289	Heteroscedastic	None	Cycles of approx. 11 years	Moderate noise
Nonstationary	Airline Passengers	Monthly	144	Heteroscedastic	Positive, almost linear	Yearly seasonality	Minimal noise
	Canadian Lynx	Yearly	112	Homoscedastic	None	Cycles of approx. 10 years	Single outliers
	Daily Minimum Temperatures Melbourne	Daily	3650	Heteroscedastic	None	Yearly seasonality	Moderate noise
	Daily Total Female Births California	Daily	365	Homoscedastic	None	Weekly seasonality	Strong noise
	Rossmann Store Sales	Daily	808	Heteroscedastic	None	Multiple seasonalities	Moderate noise
	Random Walk	Daily	1000				

Table 5: Identified and clustered time series

5.3 Forecasting Methods

This section describes the forecasting methods, which are the core of the experiment. The experiment thereby includes statistical, machine learning, and hybrid approaches as well as baseline methods. Table 6 gives an overview of the particular forecasting methods.

Category	Method
Baseline	Naive approach
	Linear regression
Statistical methods	Holt-Winters' Method
	ARIMA
Machine learning methods	Multilayer Perceptron
	Long Short-Term Memory Network
Hybrid methods	Slawek Smyl's HW-LSTM method

Table 6: Overview of forecasting methods

First, two simple forecasting methods are used as a baseline to compare the other methods against. As Section 3.2.1 describes, naive approaches are a useful tool in that endeavor. For that reason, the first baseline is the *naive approach* that sets all values to the last observation. As a second baseline method, a *linear regression* model is used. It is characterized by its straightforward handling and provides another reasonable baseline.
In terms of statistical methods, the experiment considers two well-established methods. It applies the *Holt-Winters' Method* from the class of exponential smoothing methods and the *ARIMA* class of autocorrelation models. Consequently, the experiment uses two successful methods with different characteristics that have proven their capabilities successfully in various contexts.
From the presented machine learning methods, the experiment includes a *Multilayer Perceptron* and a *Long Short-Term Memory Network*. The MLP represents an entry-level method, while the LSTM has a more sophisticated architecture. Thus, two methods of different complexity are considered and compared in the experiment. Finally, *Slawek Smyl's hybrid HW-LSTM method* represents the hybrid methods in the experiment. The forecasting results of the hybrid approach allow for the examination of whether it can outperform its constituent parts or the isolated methods can provide better results than their hybridization.

With this selection, the experiment covers a broad range of time series forecasting methods. Nevertheless, a remark must be made on the comparability of the methods. These methods are per se different in their approach to time series forecasting. For instance, some methods apply a recursive forecasting strategy while others deploy a multi-input multi-output strategy. However, this experiment focuses not on forecasting strategies but the comparison of forecasting methods. Hence, the forecasting methods are used according to their respective nature. To make them more comparable and the comparison fair for all methods involved, every method is tuned to achieve the best forecasting performance. With this effort, each forecasting method produces the best result it is capable of and shows its forecasting potential. Therefore, it seems fair that this best potential is compared for the purpose of this thesis's experiment.

5.4 Evaluation of Forecasts

The experiment's final step is to evaluate the predictions conducted by the forecasting methods. Section 3.1.2 presented several accuracy measures with their respective advantages and disadvantages.

In the experiment, two scale-dependent error metrics evaluate the performance of the forecasting methods. The MAE assesses the forecasting method's performance on a

particular dataset. Its advantage is its interpretability, as the MAE allows for direct statements about the accuracy of a method. However, as the experiment intends to punish outliers and serious forecasting errors, the RMSE is the decisive error metric. Despite being more challenging to interpret, the RMSE is widespread in practice and is commonly used to compare the different methods on the same dataset. Therefore, the methods are ranked according to the RMSE, and the relative improvement over the best baseline is calculated.
Using the percentage error MAPE would be appreciated, as it is easy to interpret the result. However, as stated in Section 3.1.2, the MAPE is not defined for $y_t = 0$. This missing definition is problematic as the values of some time series such as the Daily Minimum Temperatures Melbourne can take the value 0 and exclude the use of the MAPE. To use accuracy measures that apply to each time series, the MAE and RMSE are the error metrics of choice for the experiment.

The evaluation of forecasts in the experiment follows the conditions of the M4 Competition. As the M4 Competition setting does not intend a walk-forward validation, it is not applied in the experiment. Thus, the error is calculated once for the entire forecast.

5.5 Implementation

This chapter closes with an elaboration on the experiment's implementation. First, this section presents the requested forecasting horizons for each frequency. Then it gives a general overview of the deployed technologies and finally elaborates on the particular implementation of each forecasting method.

The experiment uses the same forecasting horizons as the M4 Competition. They depend on the frequency of the time series, as Table 7 shows. These forecasting horizons are determined based on the nature of the decision that each data frequency is most likely to support within a company or organization. Yearly data typically supports long-term decision making on a strategic level between one and five years ahead. Monthly forecasts are frequent for budgeting purposes varying from a few months to two years, and daily forecasts usually back operations at short-term levels up to a few weeks ahead (Makridakis et al., 2020).

Frequency	Yearly	Monthly	Daily
Forecasting Horizon	6	18	14

Table 7: Overview of forecasting horizons

The entire project is implemented in Python 3.7 using well-established libraries such as SciPy, statsmodels, and scikit-learn (SciPy 1.0 Contributors et al., 2020; Seabold and Perktold, 2010; Pedregosa et al., 2011). The neural networks for the machine learning methods are developed with Keras running on a Tensorflow backend (Chollet, 2015; Abadi et al., 2015). Finally, the implementation of the hybrid method uses the open-source machine learning framework PyTorch (Paszke et al., 2019).

The naive approach's implementation is straightforward. The forecast is set to be the last known value, which is the last value of the validation set. For the linear regression, the corresponding sci-kit learn module fits the regression parameters on the training and validation set and then predicts the values for the test period.

The implementation of the Holt-Winters' Method is accomplished with the designated statsmodels module. The Holt-Winters' Additive Method is used for each time series except for the Airline Passengers, which requires the Holt-Winters' Multiplicative Method. As the HW incorporates per se a recursive one-step-ahead forecasting strategy, it is susceptible to error accumulation.
The implementation expects the specification of a seasonal component. This seasonal component was obtained from the ACF, which considered the training and validation set and indicated the lag to the value with the most significant correlation. Thereby the last observation is omitted. In the case of the Airline Passengers, the ACF would suggest a seasonal component of 2 because of the strong correlation. As this contradicts the strong yearly seasonality, the seasonality overrules the ACF, and the seasonal component is set to 12. This manual intervention provides a practical example of a priori knowledge incorporated in the forecasting method.

Two libraries are in consideration for the implementation of the ARIMA. As PyFlux requires an older version of Python (Taylor, nd), statsmodels is the library of choice. The ARIMA can deploy both direct or recursive forecasting strategy. As the direct strategy requires more preprocessing of the data and statsmodels does not natively include it, the recursive forecasting strategy is implemented. As the ARIMA models in the M competitions use a recursive strategy, this particular procedure is considered reasonable (Makridakis et al., 2018b).
ARIMA requires the specification of the parameters p, d, and q. In that endeavor, a grid search hyperparameter tuning is performed on the training set to find the optimal parameter combination. As the computation time increases with the more values included, a trade-off between optimization and computing time is found. Hence, the grid search considers only promising combinations of the hyperparameters. These promising values reflect the behavior of the time series in terms of frequency, seasonality, and cycles. In case of the strong yearly seasonality in the monthly Airline Passengers dataset, the eligible values for the autoregressive and moving average component are the interval from 0 to 12 and additionally 24. Thus, 24 months or

in other words, two seasonal periods are used to fit the ARIMA hyperparameters. In terms of differencing, the grid search considers the original time series and, additionally, the first-order and second-order difference. The parameters are validated for the MSE on the validation set, which has the same length as the test set. The MSE is broadly used in practice and provides a reasonable measure. On the whole, this approach ensures a reliable forecasting performance of the ARIMA.

The Multilayer Perceptron consists of three layers: an input layer, one hidden layer, and one output layer. It uses a ReLu activation function, an Adam optimizer, and an MSE loss function. This architecture of the MLP implies a multi-input multi-output forecasting strategy.
The samples are framed according to the supervised learning setting described in Section 3.3.1, and split into the respective sets. Thereby, the test set contains as many observations as the input window and the forecasting horizon combined. For instance, if the input window has six values and the requested forecasting horizon is 18 values, the test set holds 24 values. To ensure adequate data for the validation of the NN training, the validation set contains twice as many observations as the test set. Finally, the remainder of observations is used for training and is consequently in the training set.
A grid search determines the size of the input window. In that endeavor, the NN is trained on the training set and validated on the validation set. The performance of different input windows is compared for the MSE, and the best input window size is used to conduct the actual forecast. The eligible values tested in the grid search draw from the characteristics of the time series, similar to the process described for the hyperparameter tuning of the ARIMA. As the Industrial Production dataset has too few observations, this process can not be applied to that time series. Instead, the input window size is set to 3 concerning the ACF.

The Long Short-Term Memory Network's underlying architecture consists of an input layer, several hidden layers, and an output layer. It uses a ReLu activation function, and the loss function is the MSE. The Keras Tuner performs the hyperparameter tuning and optimizes the number of hidden layers and neurons per layer. Furthermore, it analyzes whether dropout layers are inserted. The optimizer associated is Adam with learning rates of 0.01, 0.001, and 0.0001. These learning rates are broadly used and ensure that the SGD finds a minimum. The estimation of the input window size and sample split is carried out analogously to the MLP. Per definition, the LSTM follows a recursive forecasting strategy.

Initially, Slawek Smyl's hybrid HW-LSTM method is implemented in C++, making it unavailable to the experiment's Python pipeline. Redd et al. (2019) generalize the method and provide a PyTorch implementation. The underlying concept of the hybrid HW-LSTM method is the same in both implementations and follows a re-

cursive forecasting strategy. The model incorporates cross-learning, as it uses the M4 Competition dataset for the training of the LSTMs.
However, there are some modifications to the respective implementations. The main one is the architecture of the LSTM. While Smyl (2020) deploys the stacked architecture described in Section 4.3, Redd et al. (2019) utilize a dilated LSTM architecture provided by Zalando Research (Zalando Research, nd). It uses similar dilations and is suitable for use in the hybrid architecture, although it lacks the ensemble of specialists. This modification influences the computation time for the better, as Redd et al. (2019) describe a tremendous decrease in required computation time.
The PyTorch implementation only provides accuracy measures out of the box. Therefore, the code is extended for use in the experiment. An adjusted data loader enables the consideration of further time series, and the training process was adapted accordingly. Furthermore, the forecasts were extracted from the model and brought to a comprehensible output format.

While the applied baseline and statistical methods provide a deterministic result, the result of the NN-based methods is stochastic. This fluctuation is because the NNs deploy training methods such as Stochastic Gradient Descent, Dropout Layers, and Random Search that lead to unsteady results for the same input. The MLP, LSTM, and HW-LSTM perform ten runs to overcome this shortcoming, and the average of the runs determines the final forecast. The fluctuation of each forecasting method is listed in Appendix A.9.

6 Results

This chapter presents the result of the experiment. It is divided into two sections and presents the observations and their implications. These implications reveal whether the experiment confirms the potential of hybrid methods promised by the literature, and evaluate the performance of the hybrid HW-LSTM method in another setting. Finally, this chapter answers the research question.

6.1 Observations

The experiment's outcome is the forecast of the applied methods on the identified and clustered time series. This section analyzes the forecasts for each time series individually and is structured according to the identified clusters. It will solely include selected plots to ensure a pleasant flow of reading. For completeness, Appendix A.8 includes every plot for each time series and forecasting method.

The first cluster of stationary time series included the Industrial Production dataset. The performance of the forecasting methods is presented in Table 8. Apart from the MAE and RMSE, the table exhibits the *improvement over baseline (IOB)*. The IOB relates the accuracy measured by the RMSE of a particular forecast to the best baseline and is denoted in percent. The rank is based on the IOB and indicates the performance compared to the other forecasting methods.

Time series	Error	Naive	Linear Regression	HW	ARIMA	MLP	LSTM	HW-LSTM
	MAE	7.373	7.450	2.021	2.791	3.901	10.001	3.093
Industrial	RMSE	8.339	8.575	2.607	3.406	4.637	12.227	3.577
Production	IOB			+ 68,75%	+ 59.16%	+ 44.39%	- 46.62%	+ 57.11%
	Rank	**5**	**6**	**1**	**2**	**4**	**7**	**3**

Table 8: Results cluster stationary time series

The accuracy measures show that HW, ARIMA, MLP, and HW-LSTM achieved excellent performance on the stationary time series. Solely the LSTM is well behind the others and is not able to surpass any baseline method.
The Industrial Production dataset provided a challenging task for the forecasting methods as it only contains 84 observations. This small number of observations certainly complicated the learning process for the NNs. This concern is reflected in the results, as both statistical methods obtained better results than the machine learning methods. HW and ARIMA achieved superior results and surpassed the best baseline by 68.75% and 59.16%. The HW-LSTM method was close behind the

ARIMA and delivered a reliable forecast that is depicted in Figure 23.

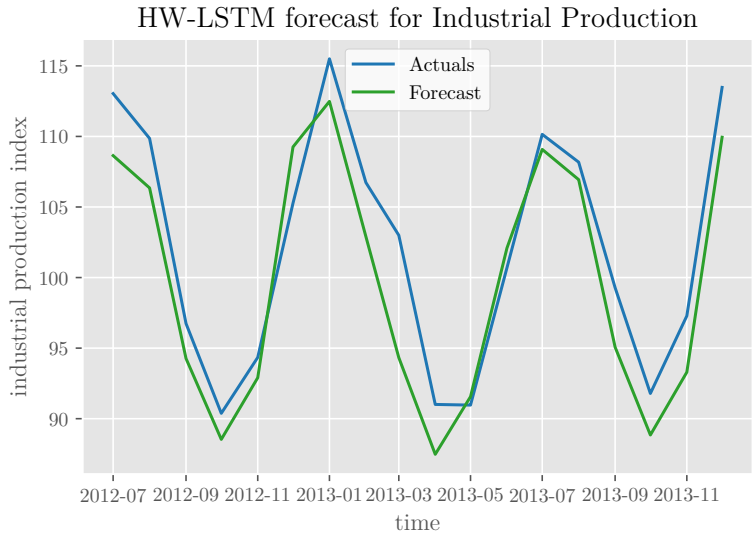


Figure 23: HW-LSTM forecast for Industrial Production

The second cluster included the trend stationary time series GBP USD Daily Exchange Rate and Sunspot. Table 9 presents the results of the forecasting methods for this cluster.

Time series	Error	Naive	Linear Regression	HW	ARIMA	MLP	LSTM	HW-LSTM
GBP USD Daily Exchange Rate	MAE	0.024	0.152	0.024	0.026	0.025	0.028	0.047
	RMSE	0.029	0.154	0.029	0.031	0.031	0.034	0.055
	IOB			0.00%	- 6.90%	- 6.90%	- 17.24%	- 89.66%
	Rank	**1**	**7**	**1**	**3**	**3**	**5**	**6**
Sunspot	MAE	70.367	31.340	42.962	11.222	21.310	13.204	87.856
	RMSE	76.599	35.332	45.985	15.473	29.988	23.609	94.101
	IOB			- 30.15%	+ 56.21%	+ 15.13%	+ 33.18%	- 166.33%
	Rank	**6**	**4**	**5**	**1**	**3**	**2**	**7**

Table 9: Results cluster trend stationary time series

The expectations for the GBP USD Daily Exchange Rate are low, as its prediction is an almost impossible task. The results confirm this hypothesis, as none of the forecasts was able the beat the naive approach. As stated in Section 3.2.1, naive forecasts work surprisingly well in this domain.

The results of all forecasting methods were very close together, and only the HW achieved a slightly better result. The plot shows that the HW imitated the naive approach and nearly matched its predictions because the seasonal component was set to 2. Despite the strong short-term autocorrelation that could favor ARIMA and MLP, both could not perform well.

The Sunspot dataset put HW and HW-LSTM before difficulties, as both were not able to surpass the linear regression baseline. The MLP was able to beat the baseline, but its performance stood short of the ARIMA and LSTM. The LSTM obtained a promising result and surpassed the baseline by 33.18%. It is only outperformed by the ARIMA that scored 56.21% above the baseline. Figure 24 visualizes both forecasts for the Sunspot dataset.

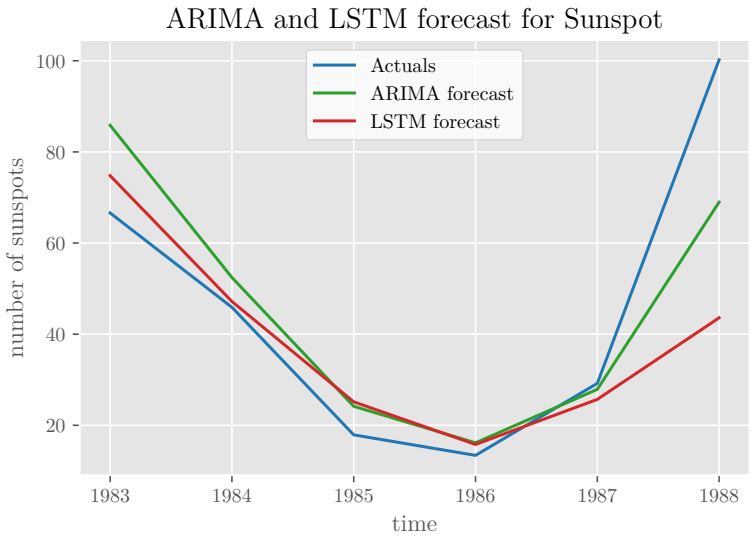


Figure 24: ARIMA and LSTM forecast for Sunspot

The time series Airline Passengers, Canadian Lynx, Daily Minimum Temperatures Melbourne, Daily Total Female Births California, Rossmann Store Sales, and Random Walk are nonstationary and consequently in the final cluster. The error metrics for these time series are presented in Table 10.

Time series	Error	Naive	Linear Regression	HW	ARIMA	MLP	LSTM	HW-LSTM
Airline Passengers	MAE	62.778	63.179	13.743	39.727	43.358	96.537	64.704
	RMSE	74.878	83.737	15.500	51.055	53.676	120.201	74.657
	IOB			+ 79.30%	+ 31.82%	+ 28.32%	- 60.53%	+ 0.30%
	Rank	**5**	**6**	**1**	**2**	**3**	**7**	**4**
Canadian Lynx	MAE	1,967.833	810.642	1,354.480	422.728	414.570	631.005	2,001.547
	RMSE	2,019.303	927.960	1,366.429	499.116	477.867	742.795	2,136.074
	IOB			- 47.25%	+ 46.21%	+ 48.50%	+ 19.95%	- 130.19%
	Rank	**6**	**4**	**5**	**2**	**1**	**3**	**7**
Daily Minimum Temperatures Melbourne	MAE	1.157	2.866	1.174	1.191	1.346	1.618	2.346
	RMSE	1.602	3.132	1.656	1.630	1.696	1.986	2.663
	IOB			- 3.37%	- 1.75%	- 5.87%	- 23.97%	- 66.23%
	Rank	**1**	**7**	**3**	**2**	**4**	**5**	**6**
Daily Total Female Births California	MAE	6.000	5.973	5.121	5.289	4.472	5.370	4.376
	RMSE	8.009	6.563	6.496	6.598	4.877	6.451	5.574
	IOB			+ 1.02%	- 0.53%	+ 25.69%	+ 1.71%	+ 15.07%
	Rank	**7**	**5**	**4**	**6**	**1**	**3**	**2**
Rossmann Store Sales	MAE	605.357	576.044	596.973	392.495	283.678	566.477	787.563
	RMSE	755.322	712.042	749.366	561.065	433.387	747.043	949.509
	IOB			- 5.24%	+ 21.20%	+ 39.13%	- 4.92%	- 33.35%
	Rank	**6**	**3**	**5**	**2**	**1**	**4**	**7**
Random Walk	MAE	1.071	11.475	1.051	2.723	0.974	0.921	2.937
	RMSE	1.389	11.521	1.349	3.031	1.195	1.157	3.317
	IOB			+ 2.88%	- 118.21%	+ 13.97%	+ 16.70%	- 138.80%
	Rank	**4**	**7**	**3**	**5**	**2**	**1**	**6**

Table 10: Results cluster nonstationary time series

The HW obtained a superior performance on the strongly seasonal Airline Passengers dataset and outperformed all other methods. The predictions surpassed the baseline by 79.30% and are depicted in Figure 25.

ARIMA and MLP achieved similar performances and improved the baseline by 31.82% and 28.32%, respectively. The HW-LSTM provided a mixed result and beat the baseline by a narrow margin. The LSTM itself failed to conduct a reliable forecast and obtained a result of 60.53% below the baseline.

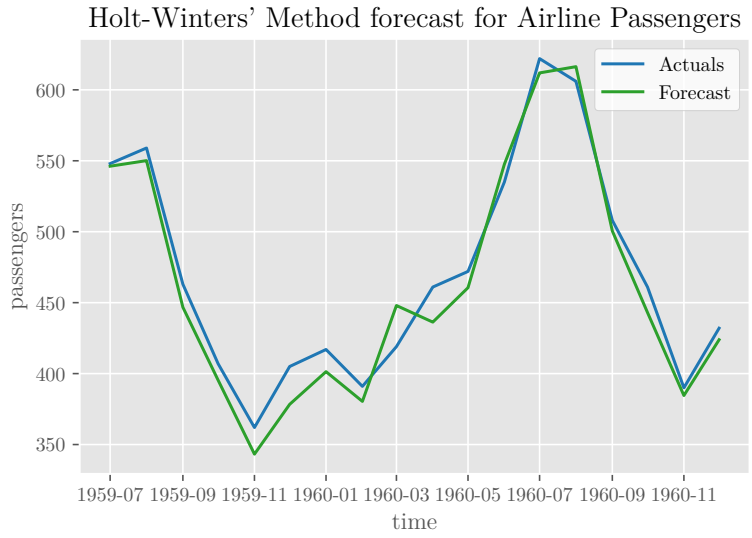


Figure 25: Holt-Winters' Method forecast for Airline Passengers

In terms of the Canadian Lynx, two methods were almost on par. The MLP performed best with 48.50% above baseline, closely followed by ARIMA with 46.21%. The LSTM delivered promising results on the time series and beat the baseline by 19.95%. The HW and HW-LSTM failed to conduct a reasonable forecast and both score below the baseline. Thereby the HW-LSTM misses the baseline by far. Figure 26 shows the MLP's forecast for the Canadian Lynx time series.

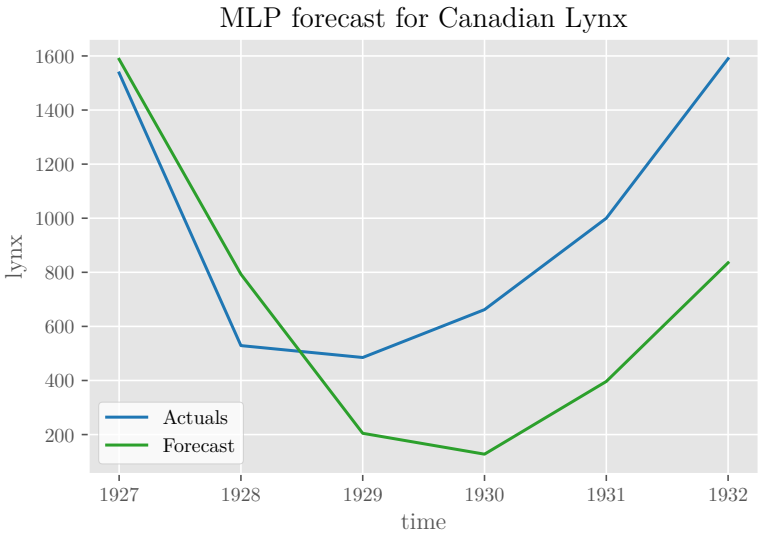


Figure 26: MLP forecast for Canadian Lynx

The Daily Minimum Temperatures Melbourne dataset was another challenging forecasting task. This challenge resulted in the fact that none of the forecasting methods conducted a reliable forecast, and consequently, all fell short of the baseline. Soley the statistical methods and the MLP seemed to make something out of the yearly seasonality and short-term autocorrelation, but the performance was still unsatisfactory. The LSTM and HW-LSTM were far behind.
For the Daily Total Female Births California, the ARIMA and HW-LSTM stood out by beating the baseline by 25.69% and 15.07%. While HW and LSTM were barely above the performance of the baseline, ARIMA did not outperform the linear regression. Figure 27 presents the forecast for the HW-LSTM method on the Daily Total Female Births California dataset.

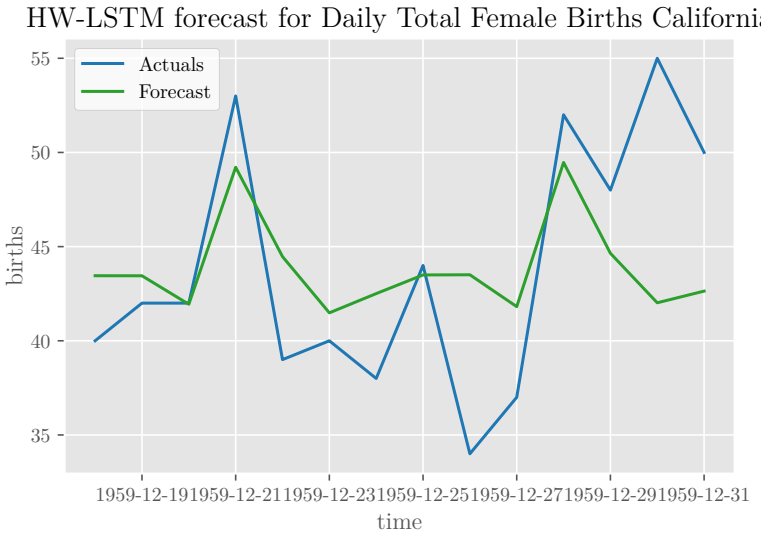


Figure 27: HW-LSTM forecast for Daily Total Female Births California

Only two methods handled the overlaying seasonalities of the Rossmann Store Sales dataset to satisfaction. Again ARIMA and MLP were close together and beat the baseline by 21.20% and 39.13%. Figure 28 presents their performance as the two most successful methods. LSTM and HW missed the baseline by a narrow margin and did not seem to cope with the overlaying seasonalities. The HW-LSTM failed to produce a reliable forecast and ranked in the last place.

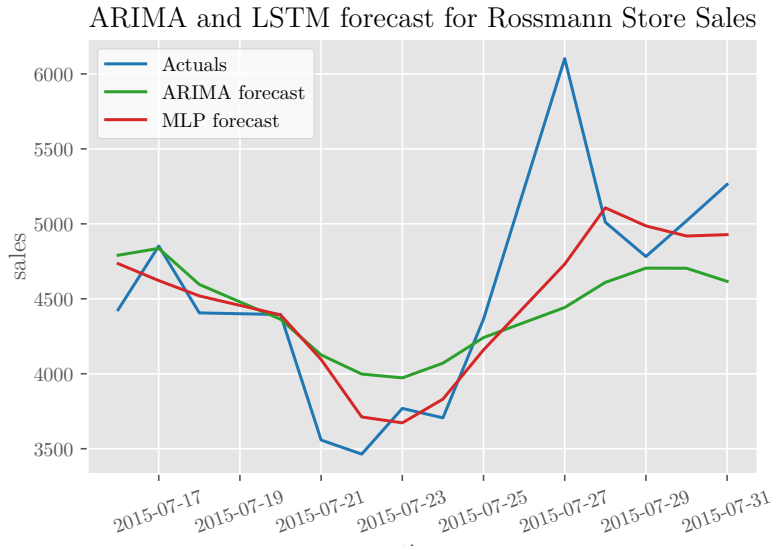


Figure 28: ARIMA and MLP forecast for Rossmann Store Sales

Finally, the Random Walk was the last time series in this analysis. It confronted the methods with randomness, which is impossible to predict. When comparing the accuracy measures, the machine learning methods achieved a better result on this time series than the statistical and hybrid approaches. The LSTM provided the best result surpassing the baseline by 16.70%, and the MLP ranked second with 13.97% above the baseline. The ARIMA and HW-LSTM fell short of the baseline for the Random Walk.

So far, this chapter presented the observations of this experiment. The next section seeks to make sense of these observations and find implications for the research question and the formulated hypotheses.

6.2 Implications

This section manifests the implications of the observations described in the previous section. It elaborates on the research question, the formulated hypotheses, and general findings.

First of all, the performance of the conducted forecast varied significantly between the time series. There was not a single time series in the experiment where all forecasting methods surpassed the best baseline. In the case of the GBP USD Daily Exchange Rate, all methods failed to conduct an accurate forecast. However, for some time series such as Airline Passengers, Industrial Production, and Canadian Lynx, most forecasting methods were able to produce reliable forecasts.
For the Random Walk, it is surprising that the machine learning methods obtained a promising result above the baseline even though the naive forecast is the optimal

per definition (Hyndman and Athanasopoulos, 2019). This achievement suggests the hypothesis that the machine learning methods learned the process described in Section 5.2 that generated the Random Walk, and approximated this particular process successfully. Since the hypothesis is based on only a single time series, it has yet to be confirmed by further investigations. However, this result has no bearing on the general predictability of random walks whose underlying processes are random and unpredictable. Much more, this example shows the variance of the forecasts and the importance of critically checking the predictions.
The literature identifies the Canadian Lynx and Sunspot as nonlinear time series. The machine learning methods performed well on both time series and can provide an improvement to the baseline methods. Therefore, the result confirms the hypothesis that machine learning methods provide reliable forecasts on nonlinear time series. However, machine learning methods were not the only methods that obtained accurate forecasts. In both cases, the ARIMA delivered a reliable forecasts as well. As time series always inhibit a combination of characteristics, it can not be assumed that nonlinearity is the only characteristic of these two time series. Therefore, no conclusion can be drawn that either the nonlinearity or autocorrelation characteristics of the time series favored the superior performance of the machine learning methods or the ARIMA alone. It is more important to conclude that several forecasting methods with different approaches can obtain accurate forecasts on the same time series due to the combination of characteristics.
If the challenging time series Daily Minimum Temperatures Melbourne and Random Walk are neglected, the forecasts of the MLP were accurate on all other time series except for the Airline Passengers. As Table 5 in Section 5.2 showed, this dataset is the only time series that exhibits a trend in the nonstationary time series cluster. Hence, two findings are evident in this cluster. First, the MLP performs well on nonstationary time series without trend. Second, as a trend in a time series requires the forecasting methods to extrapolate, this result supports the hypothesis that NN-based methods such as the MLP have difficulties extrapolating. Consequently, the forecasts on time series exhibiting trend are not reliable.

The *research question* to be answered is which characteristics of a time series determine the superiority of either statistical, machine learning, or hybrid forecasting approaches. Based on the obtained results, it is impossible to make a general assessment and answer the research question conclusively. The characteristics underlying each analyzed time series are so intertwined that no conclusion can be drawn which category of forecasting methods will be superior. Therefore, it is impossible to assess whether statistical, machine learning, or hybrid methods will have a superior performance in the first place. However, when focusing on the particular methods within the respective approaches, the experiment's results allow for several conclu-

sions which forecasting method promises a reliable forecast in a particular setting.

The results exhibit that the Holt-Winters' Method promises reliable predictions when the periodicity of a time series can be precisely specified. For the Airline Passengers dataset, this resulted in a superior performance. However, the HW faces difficulties when the periodicity can not be explicitly determined, or several overlaying seasonalities are involved, which was the case for the Rossmann Store Sales dataset. On this particular dataset, the HW was not able to provide a good forecast. On the other hand, the ARIMA and MLP handled this time series very well. This advantage could be due to their autoregressive approach to forecasting that makes them less susceptible to overlaying seasonalities.
When comparing the two machine learning methods, it is evident that the LSTM requires more input data than the MLP. Because of its sophisticated architecture, the LSTM necessitates an appropriate dataset to support the learning process. For the Industrial Production dataset, the length of the time series did not allow for an elaborate estimation of the optimum input window beforehand, and so the forecast was expectedly poor. The time series seems to be too short for the LSTM to produce an accurate forecast. On the other hand, the data seems to be enough to train the more simplistic MLP, which delivered a reliable prediction. These conclusions confirm the hypothesis that statistical methods perform better than machine learning methods when the available data is limited.

The performance of the hybrid HW-LSTM method was moderate. Although it proved its forecasting capability in the M4 Competition, the results for the time series considered in the experiment were mediocre. The results suggest the hypothesis that the HW-LSTM is a solid all-rounder that delivers on average accurate predictions when applied to many time series simultaneously but rarely produces outstanding results on a single time series. As the number of considered time series in this experiment is small compared to the M4 Competition, this hypothesis requires further investigation.
The hypothesis raised in Section 4.3 that the HW supports the LSTM to play its strengths more effectively when hybridized to the HW-LSTM method can be mostly confirmed. When neglecting the time series GBP USD Daily Exchange Rate, Daily Minimum Temperatures Melbourne, and Random Walk that barely allow for a reasonable forecast, the HW-LSTM obtained an accurate forecast, whenever the HW obtained an accurate forecast. This concordance is the case for the Industrial Production and the Airline Passengers dataset. Vice versa, when the HW was not able to conduct an accurate forecast, for instance, for Canadian Lynx and Sunspot, the forecast of the HW-LSTM was not accurate. Hence, a time series whose periodicity can be precisely determined favors the Holt-Winters' Method's performance and, consequently, the performance of the hybrid HW-LSTM method. Therefore, the ac-

curacy of the HW on a particular time series can be an indicator of the HW-LSTM method's accuracy.
However, for the Daily Total Female Births California dataset, the HW-LSTM outperforms the forecasts of its respective constituent parts. The fact that the NN-based methods rank the first three places on this particular dataset provides evidence that this time series might incorporate substantial nonlinearity. This nonlinearity furthermore suggests that a nonlinear relationship in a time series might compensate for the weak performance of the HW and still allow the HW-LSTM to conduct a reliable forecast.

Like the result of any other experiment, the result of this experiment must be scrutinized. The next chapter puts the presented observations and implications into context and provides a critical analysis.

7 Critical Review

This chapter presents a critical review of the experiment's results. Furthermore, it elaborates on the compromises made in the course of the investigation and the limitations of this thesis.

The results demonstrated that the hybrid HW-LSTM can not meet the high expectations and does not repeat the superior performance achieved at the M4 Competition. This shortcoming could be due to the modifications made by Redd et al. (2019). As the initial implementation by Smyl (2020) was not applicable in the experiment due to its implementation in C++, the modified version was the only option to investigate the HW-LSTM method.
The absence of the ensemble of specialists seems to inhibit the dilated LSTM stack's forecasting potential. Consequently, this led to a decrease in the efficiency of the hybrid method. However, the HW-LSTM still provided an accurate performance in some cases. Therefore the experiment still proves the concept of the hybrid method when time series meet the specified prerequisites. A further investigation applying additional time series should review this proof of concept in depth.

The selected time series were a core part of the experiment. The research for finding these presented a notable challenge. As time series inhibit a combination of different characteristics at the same time, only a few were identified suitable for the experiment and potentially helpful to answer the research question. The quest finally resulted in nine promising time series, which do not represent a large sample.
However, the approach of this thesis is to investigate real-world cases. Therefore, explainable time series based on real-world circumstances were preferred over large datasets such as the M4 Competition dataset. As described in Section 4.1, this dataset contains 100,000 time series from different domains. Nevertheless, as no further information is provided about the individual time series, the M4 Competition dataset presents an anonymous dataset and is not consistent with the objectives of this thesis.

The methods conducted a one-time multi-step forecast in the experiment to follow the conditions of the M4 Competition. This setting did not intend a walk-forward validation, and consequently, it was not applied in the experiment. However, the walk-forward validation promises to ensure a more accurate error estimation that could improve the assessment of the experiment's result.

Finally, the choice of considered forecasting methods has to be critically reviewed. The experiment includes the hybrid method, which is a core interest of this thesis, and its constituent parts. Apart from those methods, the ARIMA represents another statistical method following a different forecasting approach than the HW. In terms

of machine learning methods, the MLP is chosen as a more straightforward method than the LSTM. Although this is a reasonable selection with five different forecasting methods, a variety of methods were not considered in the experiment. Additional forecasting methods, such as SARIMA, TDNNs, or CNNs, could assess the research question from a different perspective.

8 Conclusion and Outlook

This master thesis examined time series forecasting from different points of view. It sought answers to the clash of the two cultures and investigated whether hybrid methods combine the best of both worlds to advance time series forecasting. In its course, it reviewed related work and laid the theoretical foundations for this thesis's core subjects. It analyzed both statistical and machine learning methods and elaborated on their respective strengths and limitations for the forecasting domain. Thereby the thesis emphasized that both approaches have properties that can complement each other. The compensation of each other's weaknesses is the foundation upon which hybrid methods are built. This thesis investigated how this sophisticated unification of statistical and machine learning is realized and provided a detailed examination of Slawek Smyl's hybrid HW-LSTM method.
The conducted experiment aimed to gain findings for the research question. However, it turned out that the research question can not be answered definitively. The results did not allow for an assessment of which characteristics of a time series determine the superiority of either statistical, machine learning, or hybrid forecasting methods. Nevertheless, the results provided useful insights on the superiority of specific methods in particular settings.

The artifact of this thesis is the experimental design implemented in Python. It allows for the analysis of time series with different forecasting methods due to its modular pipeline structure and enables the straightforward analysis of additional time series in the future. Therefore, it yields a valuable contribution to time series analysis.
Apart from further time series, many relevant related areas should be investigated in the future and connected with the thesis. While these relevant areas would improve the understanding of time series forecasting, they are well beyond the scope of this thesis and could not be included in the paper. For instance, the analysis of nonlinearity in a time series is a relevant research topic. As Zhang (2003) states, there are currently no general diagnostic statistics for nonlinear relationships available. The ability to specify the composition of a time series precisely in terms of linear and nonlinear parts would provide significant progress in understanding a time series better. An enhanced understanding of a time series to be forecast provides excellent potential for an advanced selection of the forecasting method.
Furthermore, meta-learning offers unique potential to advance time series forecasting. It provides a framework for the identification of the most suitable forecasting method. There have been promising approaches by Talagala et al. (2018), which developed a framework for algorithm selection based time series features utilizing neural networks.

Apart from the forecasting method selection, the topic of uncertainty quantification is of exceptional interest. Uncertainty quantification aims to obtain an assessment of how sure an algorithm is in its prediction. As a mere prediction is ineffective without an estimation of its reliability, this research area focuses on how especially machine learning methods can give a reasonable estimate of their uncertainty. Considering the criticized black-box nature of machine learning methods, this reasonable estimate of uncertainty is significant progress in the explainability of machine learning methods.

In the context of time series forecasting, the no free lunch theorem describes the circumstance that none of the existing forecasting methods is universally better than any other method. Each forecasting task has to be examined separately, and the most suitable method has to be selected (Wolpert and Macready, 1997).
This thesis closes with a statement made by Bontempi (2020) that takes up this theory. The author states that the top-performing method in a competition or any data science contest is not necessarily the best one since there can not be a single best method. These competitions assess the usefulness of various tools for solving a specific problem. The fact that one specific tool fulfills the task better than others reveals much more about the nature of the task than about the usefulness of the same tool in a future task. For instance, using a fish knife is very useful for eating a trout but might not be useful when pasta is on the menu (Bontempi, 2020). With this in mind, the thesis shows that hybrid methods for time series forecasting do not generally make statistical or machine learning methods obsolete, but can be a valuable tool in the forecaster's toolbox.

A Appendix

A.1 Error Calculation in the M4 Competition

The formulas for symmetric Mean Absolute Percentage Error (sMAPE) and Mean Absolute Scaled Error (MASE) are defined by (Makridakis et al., 2020):

$$\text{sMAPE} = \frac{2}{h} \sum_{t=n+1}^{n+h} \frac{\left|Y_t - \hat{Y}_t\right|}{|Y_t| + \left|\hat{Y}_t\right|}$$

$$\text{MASE} = \frac{1}{h} \frac{\sum_{t=n+1}^{n+h} \left|Y_t - \hat{Y}_t\right|}{\frac{1}{n-m} \sum_{t=m+1}^{n} \left|Y_t - \hat{Y}_{t-m}\right|},$$

with:

Y_t = value of the time series at point t
$\hat{Y}_t$ = estimated forecast
h = forecasting horizon
n = number of data points available in-sample
m = time interval between successive observations, e.g. 12 for monthly or 24 for hourly

The Overall Weighted Average (OWA) is calculated as follows (M Open Forecasting Center (MOFC), 2017):

- Divide sMAPE and MASE by the error of the seasonal naive benchmark to obtain the relative sMAPE and relative MASE, respectively.
- Compute the OWA by averaging the relative sMAPE and relative MASE.

The OWA is therefore defined by:

$$\text{OWA} = \frac{\text{relative sMAPE} + \text{relative MASE}}{2}$$

A.2 Dilated LSTM Stack Unfolded into Time

This section aims to give another perspective on the dilated LSTM stack that Smyl (2020) uses in the hybrid HW-LSTM method. The depicted architecture is the (1,2)-(4,8) Standard architecture used for time series of quarterly frequency. In analogy to Figure 8, Figure 29 is the unfolded representation of the folded model depicted in Figure 11.
The solid arrows reflect the current time step t, while the dashed arrows indicate a future time step. The black arrows allow the shortcut within a stack, and the blue arrows represent the dilation.

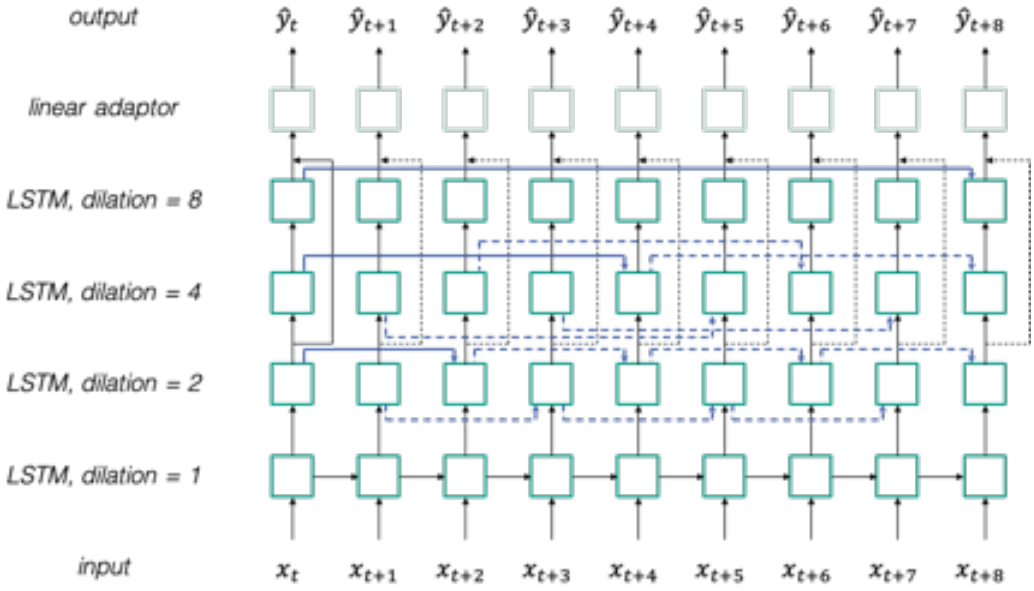


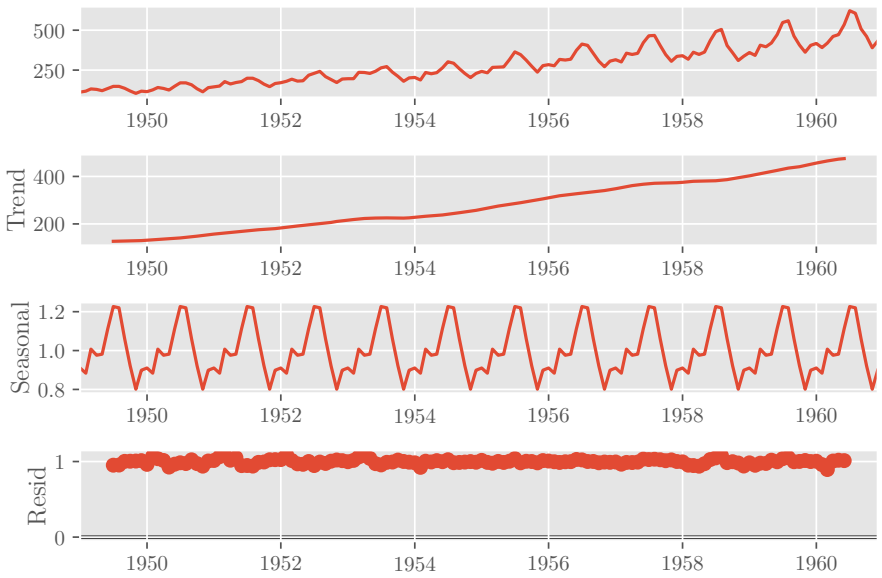
Figure 29: Dilated LSTM (1,2)-(4,8) Standard unfolded into time, adapted from Redd et al. (2019)

A.3 Experiment - Time Series Decomposition

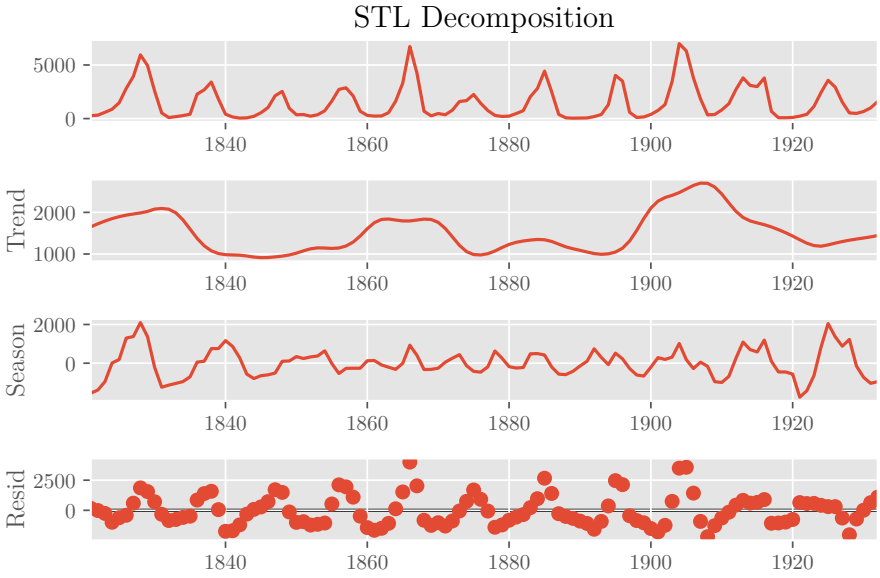
The decomposition of the time series is conducted with the statsmodels module for classical decomposition and STL decomposition (Seabold and Perktold, 2010). The output depicts four graphs ordered from top to bottom: observed time series, trend, seasonal component, and residuals. In the case of additive decomposition, trend, seasonal component and residuals are added to obtain the observed time series and for multiplicative decomposition multiplied, respectively. Table 11 displays the used decomposition method for each time series.

Time series	Decomposition method
Airline Passengers	Multiplicative Classical Decomposition
Canadian Lynx	STL Decomposition
Daily Minimum Temperatures Melbourne	STL Decomposition
Daily Total Female Births California	STL Decomposition
GBP USD Daily Exchange Rate	STL Decomposition
Industrial Production	STL Decomposition
Rossmann Store Sales	STL Decomposition
Sunspot	STL Decomposition

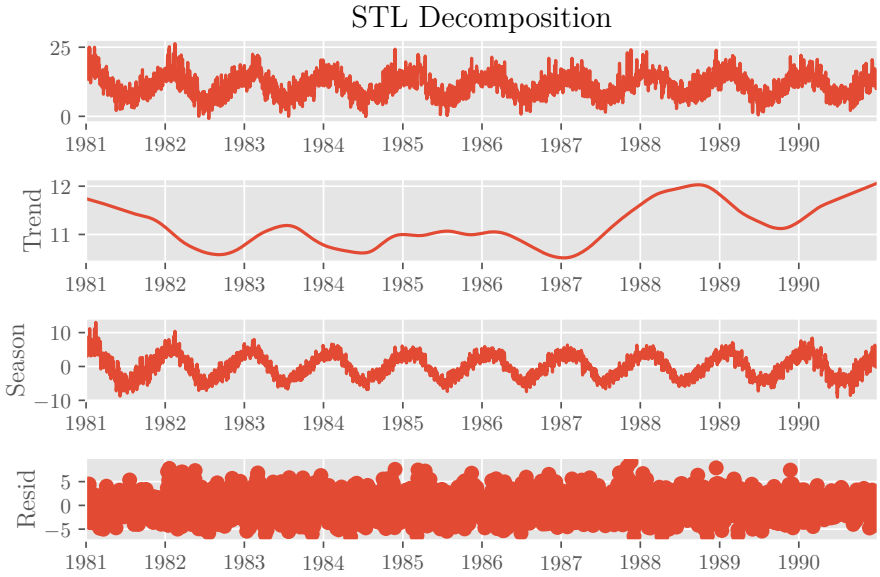
Table 11: Overview of applied methods for time series decomposition



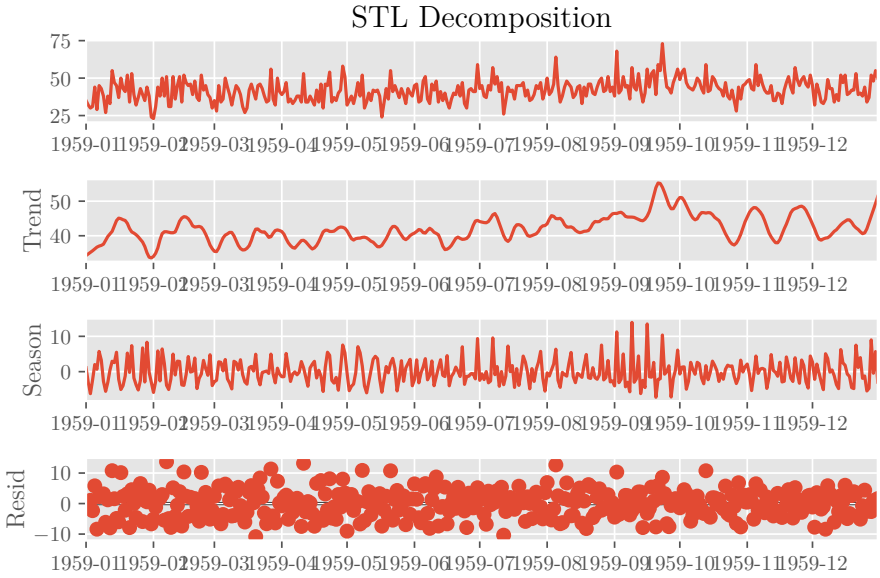
(a) Multiplicative Classical Decomposition Airline Passengers



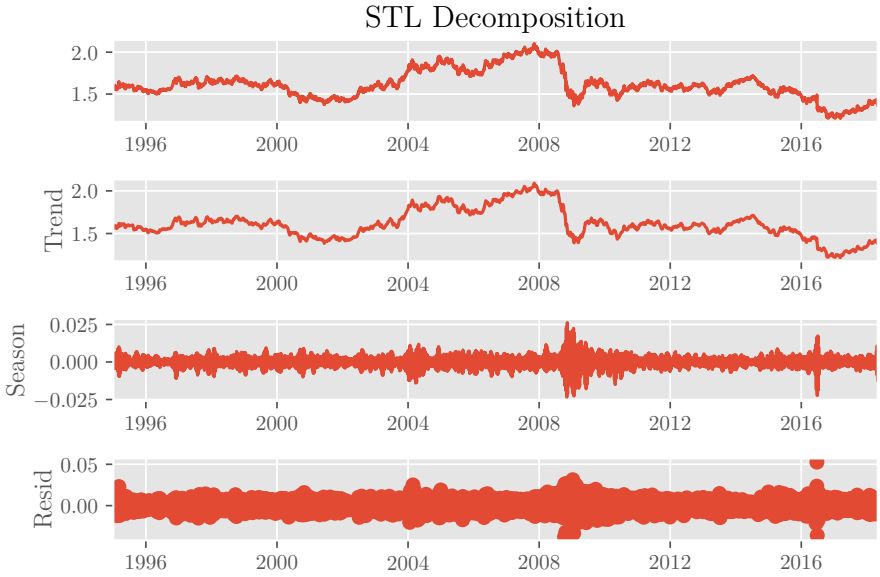
(b) STL Decomposition Canadian Lynx



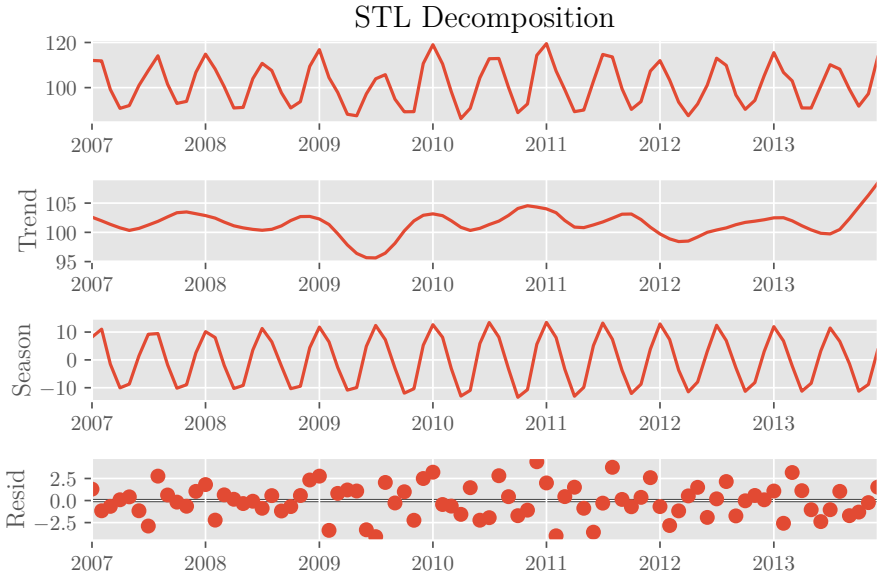
(c) STL Decomposition Daily Minimum Temperatures Melbourne



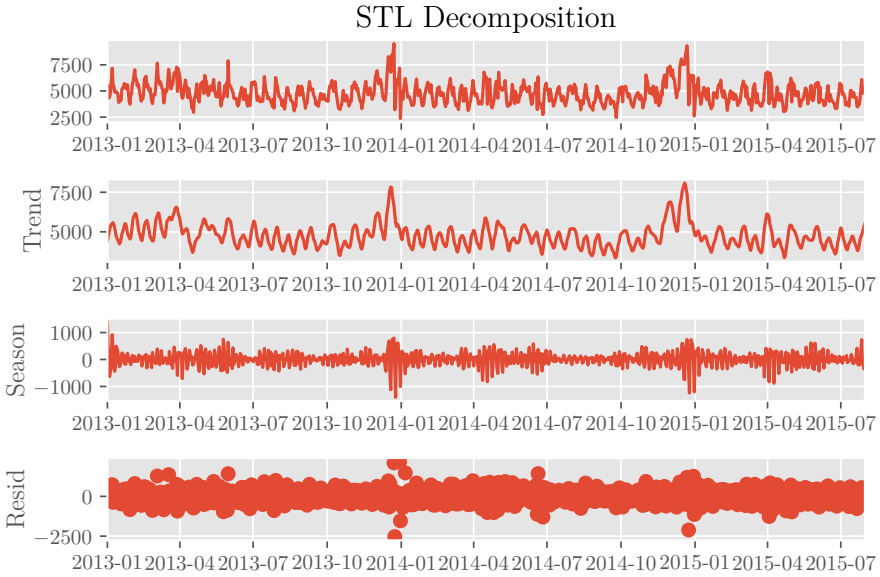
(d) STL Decomposition Daily Total Female Births California



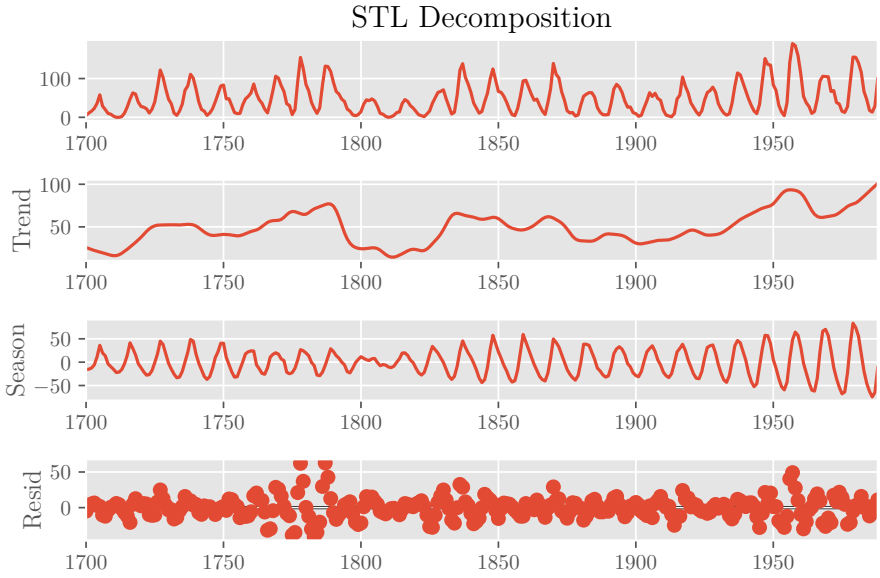
(e) STL Decomposition GBP USD Daily Exchange Rate



(f) STL Decomposition Industrial Production



(g) STL Decomposition Rossmann Store Sales

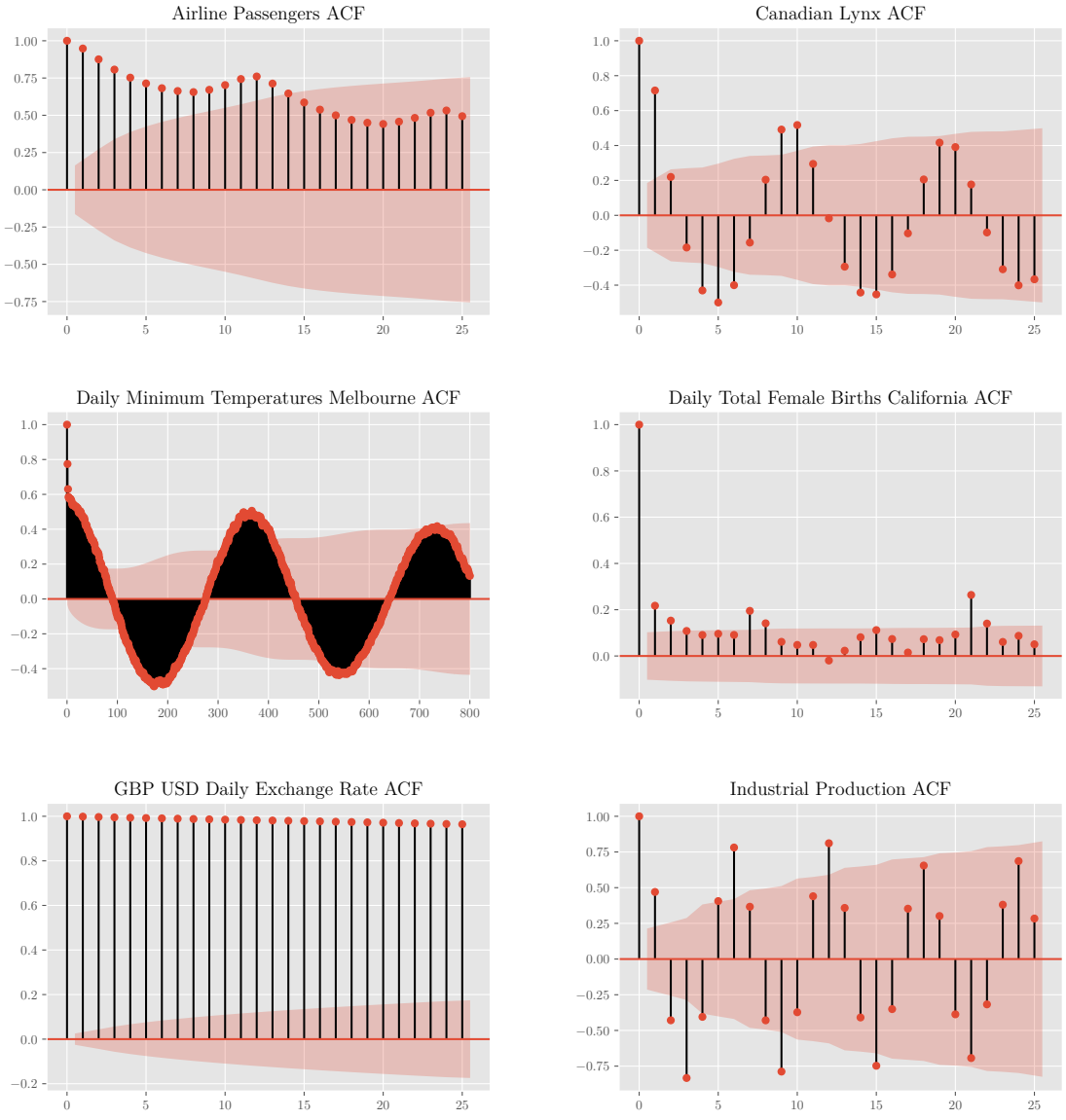


(h) STL Decomposition Sunspot

Figure 30: Decomposition for each time series in the experiment

A.4 Experiment - Autocorrelation Function

The plot of the Autocorrelation Function (ACF) is conducted with the statsmodels module (Seabold and Perktold, 2010). The output shows the lags on the horizontal axis and the correlations on the vertical axis. The values outside the red area are statistically significant, with significance level $\alpha = 0.05$.



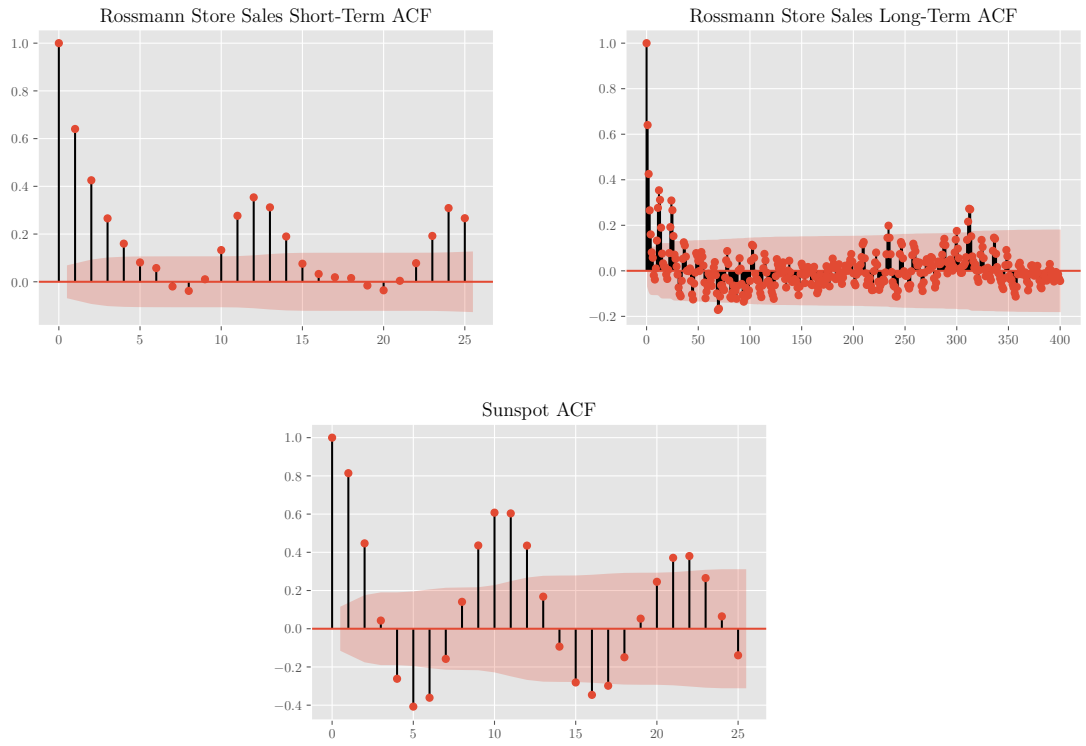


Figure 31: Autocorrelation Function for each time series in the experiment

A.5 Experiment - Results Augmented Dickey-Fuller Test

The Augmented Dickey-Fuller Test (ADF) is performed with the designated statsmodels module (Seabold and Perktold, 2010). It tests the null hypothesis that there is a unit root, with the alternative that there is none present. If the calculated p value is below the significance level $\alpha = 0.05$, the null hypothesis has to be rejected, and the result confirms stationarity. Table 12 presents the ADF results for each time series, not differenced, and in first-order difference.

Time series	**Differencing**	p **value**
Airline Passengers	none	0.386
	first-order	0.296
Canadian Lynx	none	0.974
	first-order	0.782
Daily Minimum Temperatures Melbourne	none	0.381
	first-order	0.957
Daily Total Female Births California	none	0.953
	first-order	0.999
GBP USD Daily Exchange Rate	none	0.485
	first-order	1.126×10^{-29}
Industrial Production	none	2.174×10^{-4}
Rossmann Store Sales	none	0.996
	first-order	0.695
Sunspot	none	0.146
	first-order	5.309×10^{-20}

Table 12: Results Augmented Dickey-Fuller Test

A.6 Experiment - Results Levene's Test for Equal Variances

The Levene-Test is conducted using the designated SciPy module (SciPy 1.0 Contributors et al., 2020). It tests the null hypothesis that all input samples are from populations with equal variances. If the calculated p value is below the significance level $\alpha = 0.05$, the null hypothesis has to be rejected, and the result indicates heteroscedasticity. Table 13 presents the Levene-Test results for each sample size and time series.

Time series	**Sample size**	p **value**
Airline Passengers	2	1.624×10^{-4}
	3	3.819×10^{-5}
	4	2.418×10^{-6}
	5	1.042×10^{-5}
	10	3.153×10^{-5}
Canadian Lynx	2	0.650
	3	0.502
	4	0.896
	5	0.400
	10	0.227
Daily Minimum Temperatures Melbourne	2	1.879×10^{-4}
	3	3.114×10^{-6}
	4	2.665×10^{-10}
	5	5.038×10^{-8}
	10	1.152×10^{-11}
Daily Total Female Births California	2	0.356
	3	0.086
	4	0.425
	5	0.532
	10	0.611
GBP USD Daily Exchange Rate	2	1.509×10^{-68}
	3	3.591×10^{-168}
	4	6.230×10^{-311}
	5	0.000
	10	1.788×10^{-189}

Time series	**Sample size**	p **value**
Industrial Production	2	0.895
	3	0.209
	4	0.649
	5	0.303
	10	0.868
Rossmann Store Sales	2	0.867
	3	0.317
	4	0.667
	5	0.004
	10	7.416×10^{-10}
Sunspot	2	0.036
	3	0.033
	4	0.001
	5	5.051×10^{-6}
	10	2.677×10^{-5}

Table 13: Results Levene's Test for equal variances

A.7 Experiment - Time Series First-Order Difference

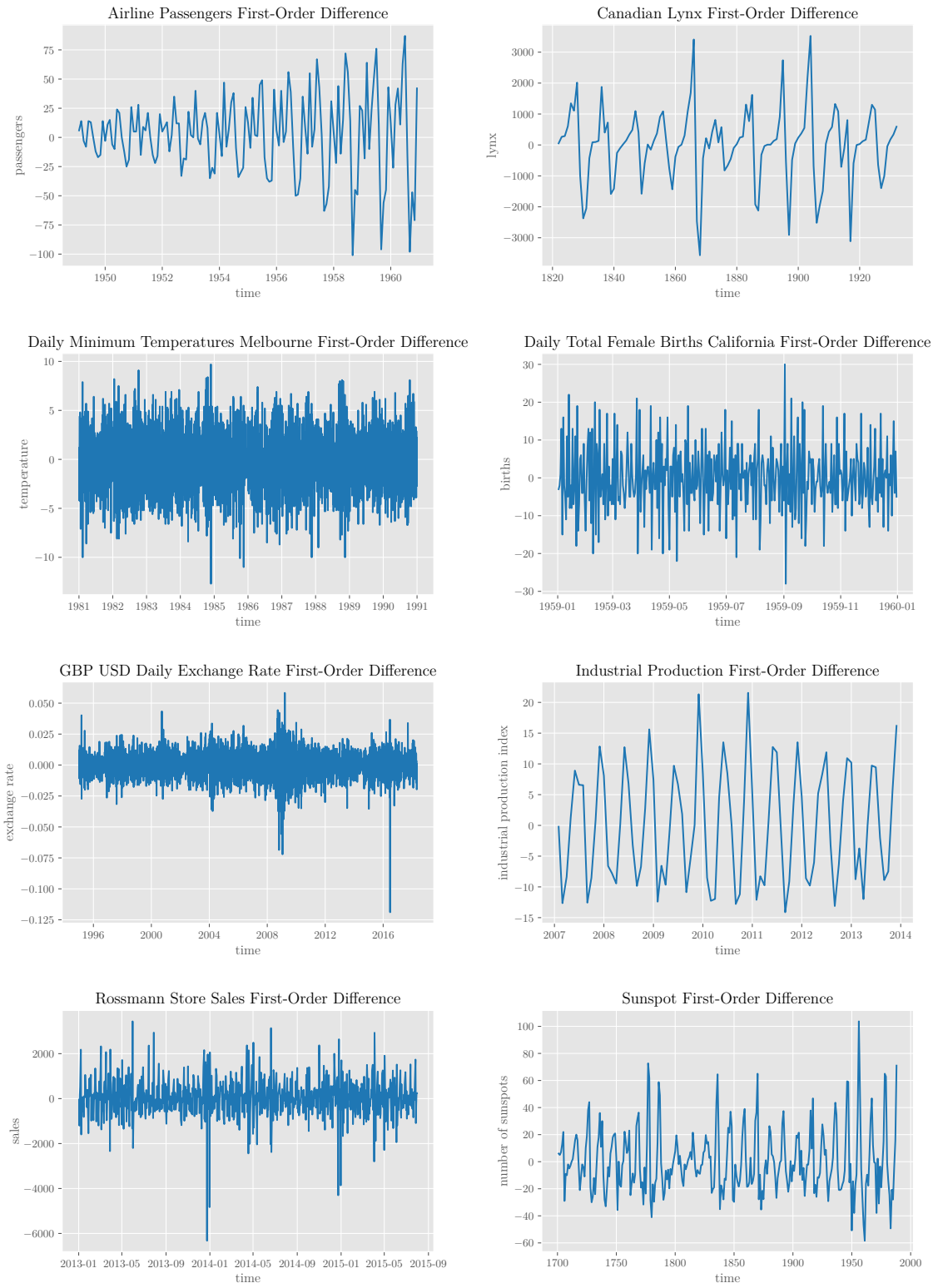


Figure 32: First-order difference for each time series in the experiment

A.8 Experiment - Forecasts

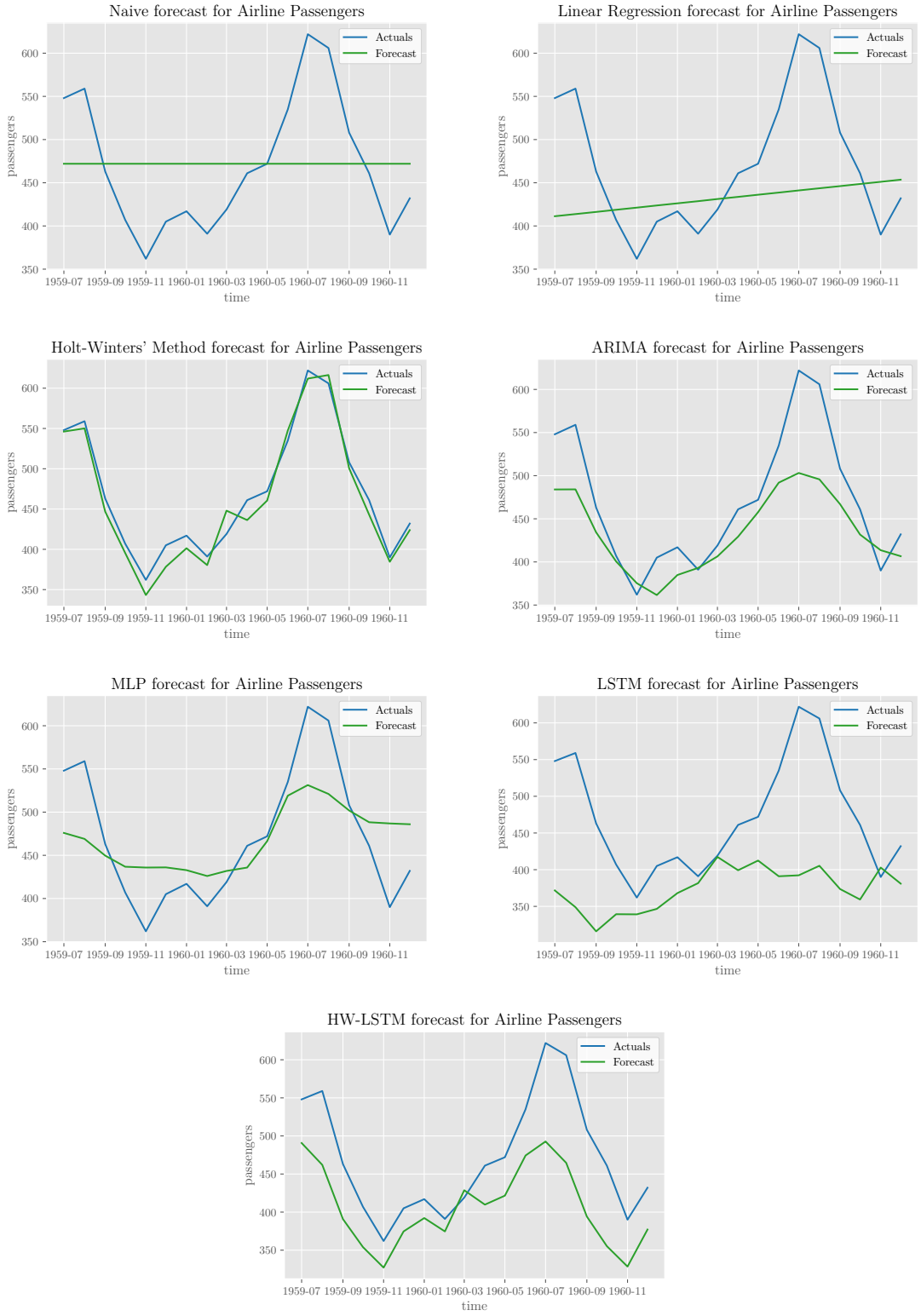


Figure 33: Forecasts for Airline Passengers

Figure 34: Forecasts for Canadian Lynx

Figure 35: Forecasts for Daily Minimum Temperatures Melbourne

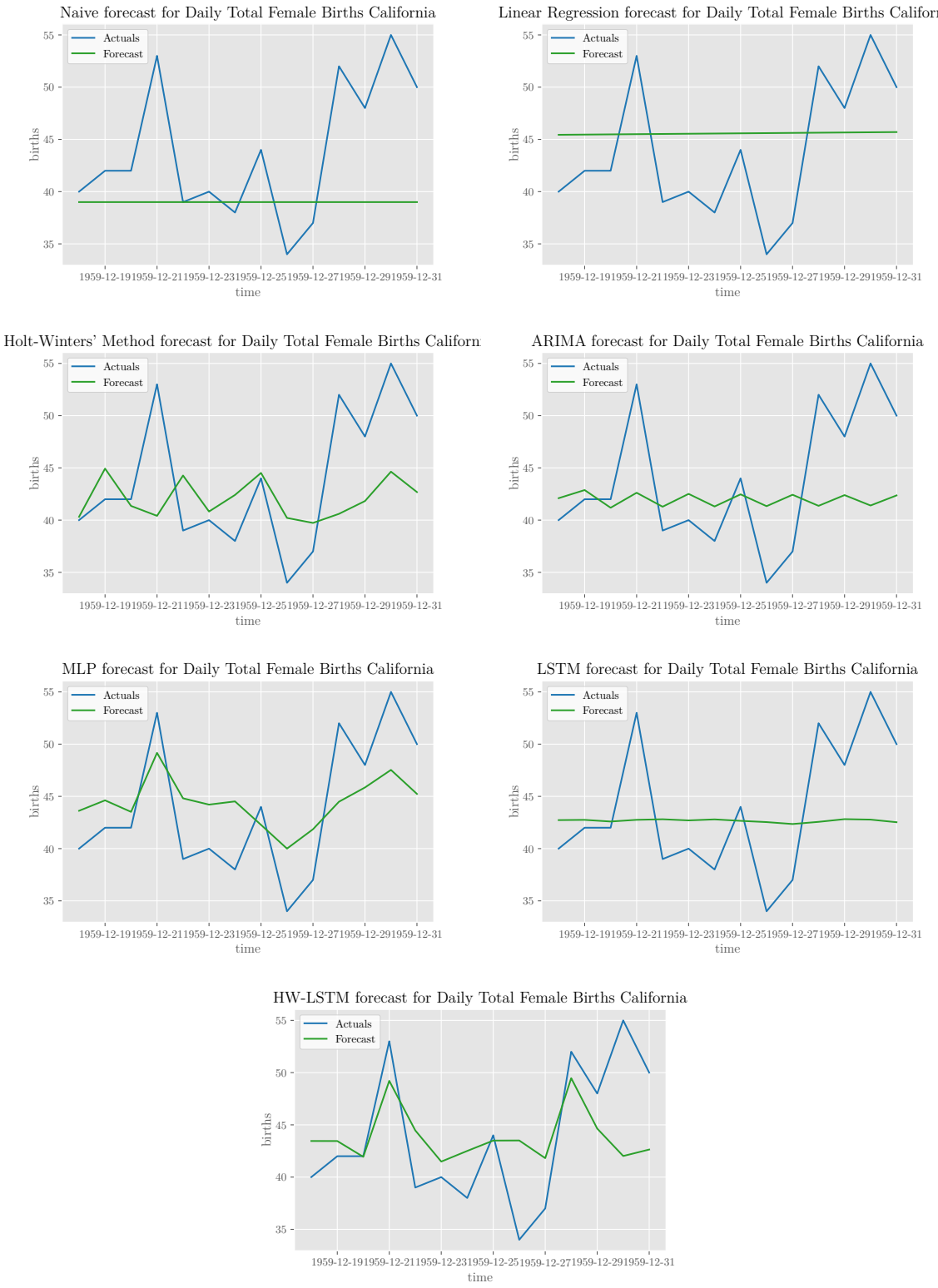


Figure 36: Forecasts for Daily Total Female Births California

Figure 37: Forecasts for GBP USD Daily Exchange Rate

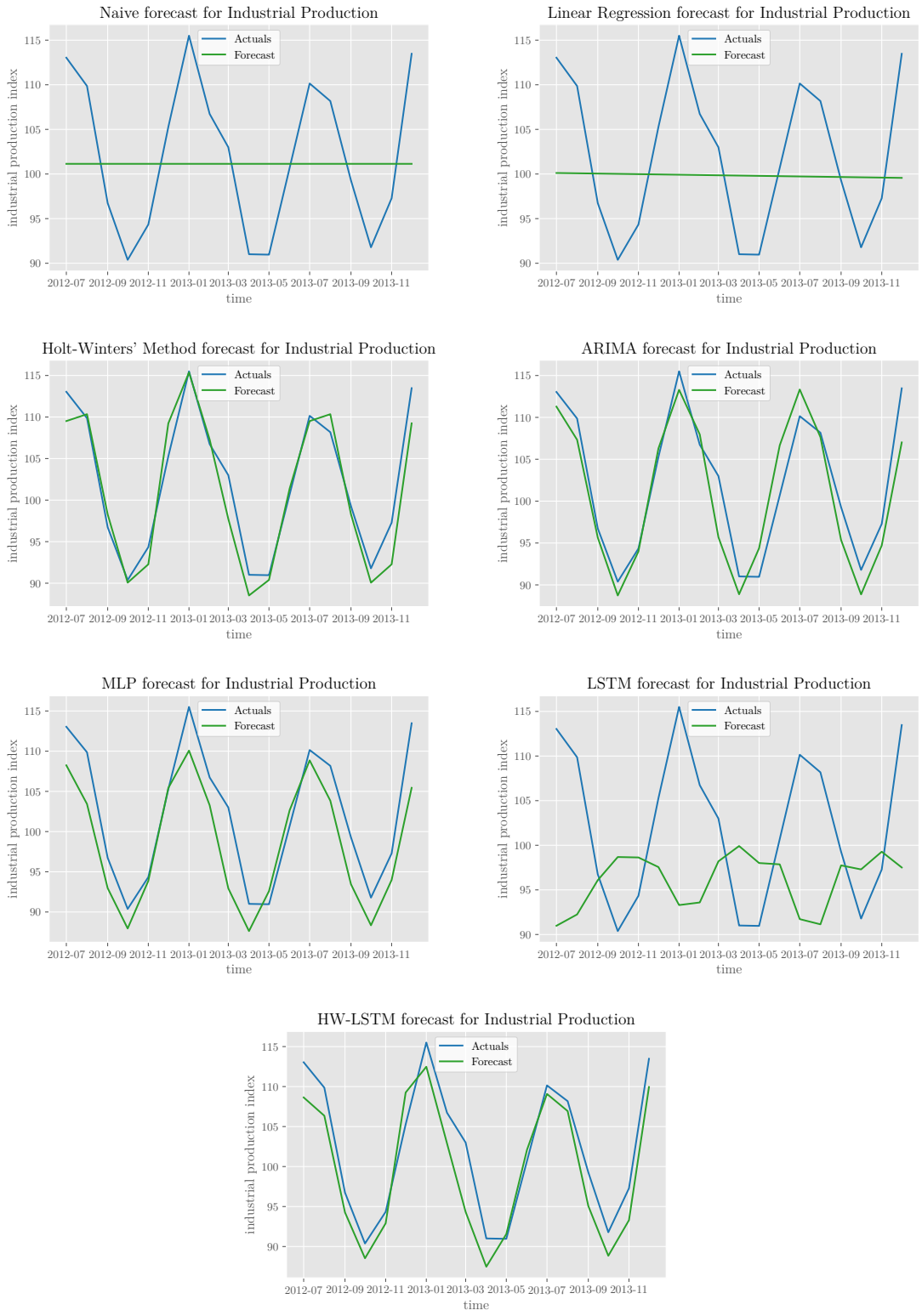


Figure 38: Forecasts for Industrial Production

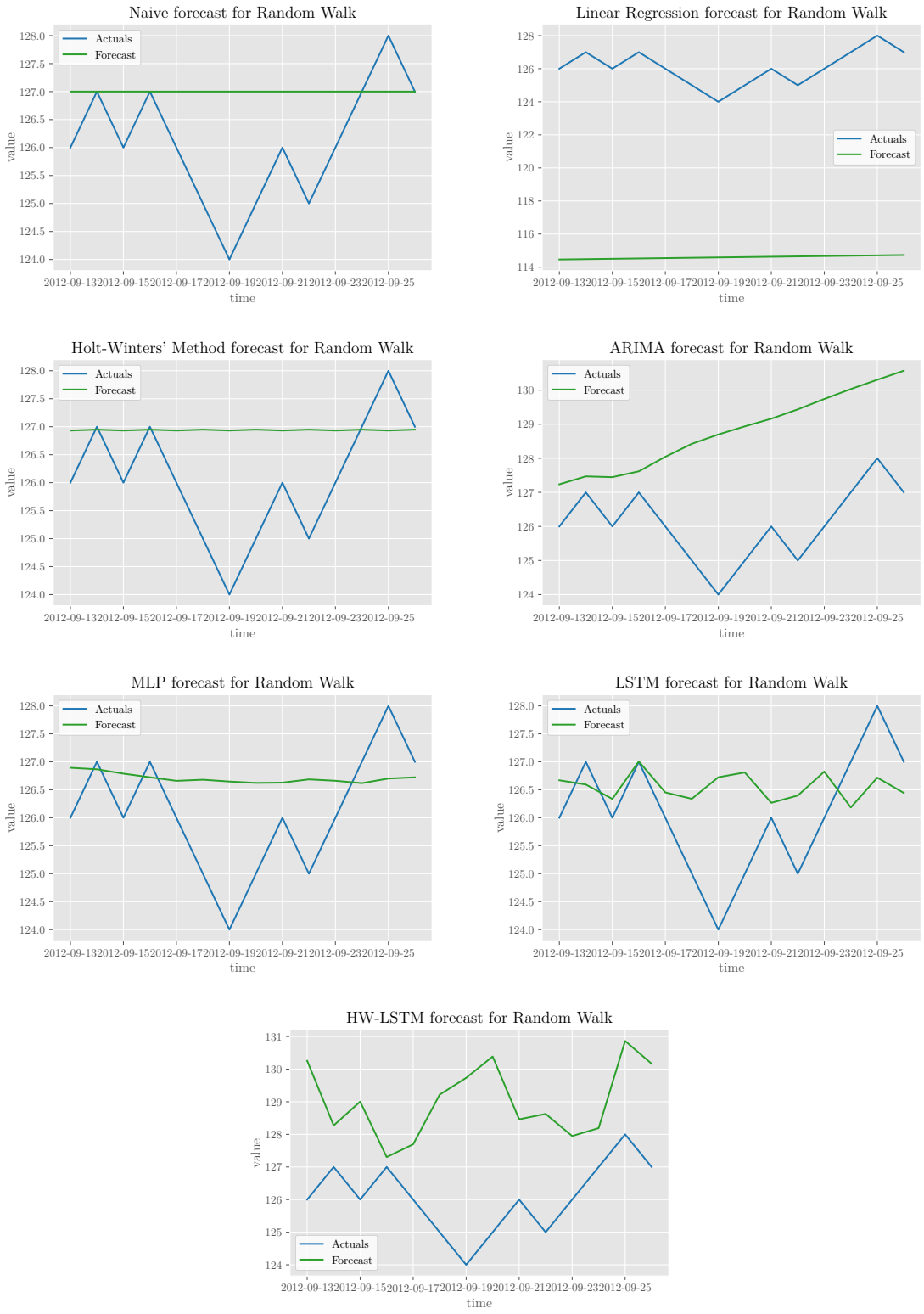


Figure 39: Forecasts for Random Walk

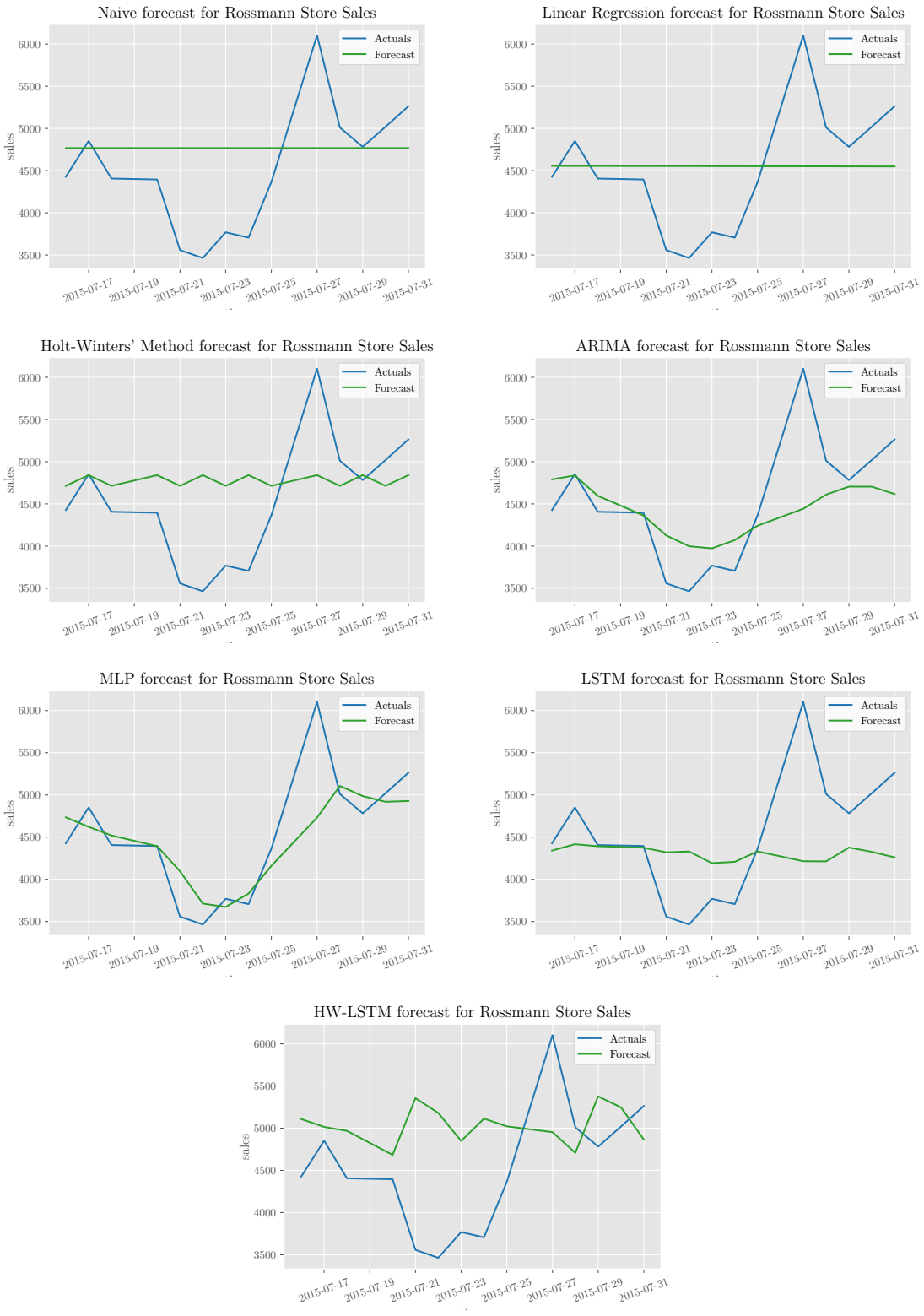


Figure 40: Forecasts for Rossmann Store Sales

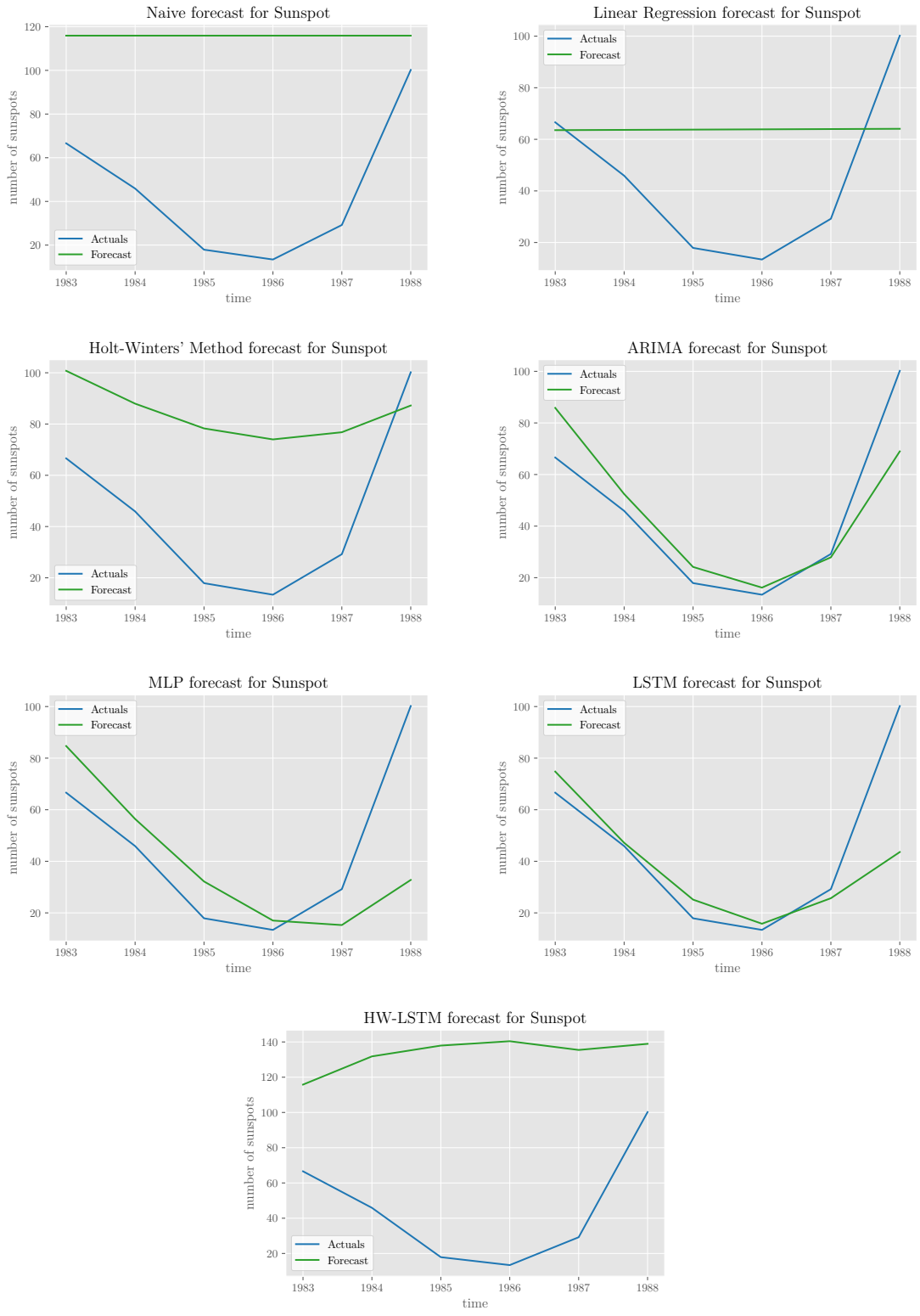


Figure 41: Forecasts for Sunspot

A.9 Experiment - Standard Deviation of NN-based methods

As described in Section 5.5, the NN-based methods obtain unsteady results for the same input. To give an overview of the fluctuation recorded in the ten runs, the MAE and RMSE are calculated for each particular run. The entries in Table 14 denote a tuple, where the first value is the mean of the respective accuracy measure for all runs, and the second value indicates the associated standard deviation.

Time series	Error	MLP	LSTM	HW-LSTM
Airline Passengers	MAE	(43.493; 1.704)	(104.354; 21.681)	(64.704; 2.099)
	RMSE	(2,915.046; 174.685)	(16,966.810; 5,899.571)	(5,582.078; 330.279)
Canadian Lynx	MAE	(426.395; 34.662)	(670.907; 191.877)	(2,001.547; 62.772)
	RMSE	(240,679.878; 35,189.392)	(667,462.245; 332,607.335)	(4,572,232.365; 407.395)
Daily Minimum Temperatures Melbourne	MAE	(1.397; 0.172)	(1.623; 0.155)	(2.350; 0.094)
	RMSE	(3.001; 0.509)	(4.000; 0.522)	(7.193; 0.544)
Daily Total Female Births California	MAE	(4.674; 0.440)	(5.426; 0.214)	(4.457; 0.186)
	RMSE	(29.102; 4.673)	(42.570; 1.926)	(31.870; 2.659)
GBP USD Daily Exchange Rate	MAE	(0.026; 0.002)	(0.028; 0.001)	(0.051; 0.010)
	RMSE	$(0.001; 2.168 \times 10^{-19})$	$(0.001; 2.168 \times 10^{-38})$	(0.004; 0.001)
Industrial Production	MAE	(3.908; 0.112)	(10.139; 0.666)	(3.115; 0.475)
	RMSE	(21.705; 1.194)	(154.432; 18.825)	(13.337; 3.619)
Rossmann Store Sales	MAE	(320.877; 36.044)	(619.891; 76.350)	(787.562; 29.709)
	RMSE	(222,278.103; 45,550.428)	(625,041.353; 154,352.583)	(912,591.675; 64,465.466)
Sunspot	MAE	(21.310; 0.993)	(15.915; 2.749)	(87.855; 2.895)
	RMSE	(902.609; 50.683)	(619.247; 131.593)	(8,873.670; 511.965)
Random Walk	MAE	(1.071; 0.279)	(1.059; 0.152)	(3.219; 0.901)
	RMSE	(1.783; 0.890)	(1.724; 0.444)	(18.190; 8.126)

Table 14: Standard deviation of the NN-based methods

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., and Devin, M. (2015). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv:1603.04467.

Adya, M., Collopy, F., Armstrong, J. S., and Kennedy, M. (2001). Automatic identification of time series features for rule-based forecasting. *International Journal of Forecasting*, 17(2):143–157.

Atiya, A. F. (2020). Why does forecast combination work so well? *International Journal of Forecasting*, 36(1):197–200.

Bandara, K., Shi, P., Bergmeir, C., Hewamalage, H., Tran, Q., and Seaman, B. (2019). Sales demand forecast in e-commerce using a long short-term memory neural network methodology. arXiv: 1901.04028.

Barrow, D. K. and Crone, S. F. (2016). Cross-validation aggregation for combining autoregressive neural network forecasts. *International Journal of Forecasting*, 32(4):1120–1137.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Board of Governors of the Federal Reserve System (US) (n.d.). Industrial Production: Electric and gas utilities [IPG2211A2N]. Retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/IPG2211A2N.

Bontempi, G. (2020). Comments on M4 competition. *International Journal of Forecasting*, 36(1):201–202.

Bontempi, G., Ben Taieb, S., and Le Borgne, Y.-A. (2012). Machine learning strategies for time series forecasting. In *European business intelligence summer school*, pages 62–77. Springer.

Borowik, G., Wawrzyniak, Z. M., and Cichosz, P. (2018). Time series analysis for crime forecasting. In *2018 26th International Conference on Systems Engineering (ICSEng)*, pages 1–10. IEEE.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Breiman, L. (2001a). Random forests. *Machine Learning*, 45(1):5–32.

Breiman, L. (2001b). Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199–231.

Brockwell, P. J. and Davis, R. A. (2016). *Introduction to time series and forecasting*. Springer International Publishing.

Brown, R. G. (1959). *Statistical forecasting for inventory control*. McGraw/Hill.

Brownlee, J. (2017). *Introduction to time series forecasting with python: how to prepare data and develop models to predict the future*. Machine Learning Mastery.

Cerqueira, V., Torgo, L., and Soares, C. (2019). Machine learning vs statistical methods for time series forecasting: size matters. arXiv: 1909.13316.

Chang, S., Zhang, Y., Han, W., Yu, M., Guo, X., Tan, W., Cui, X., Witbrock, M., Hasegawa-Johnson, M., and Huang, T. S. (2017). Dilated recurrent neural networks. arXiv: 1710.02224.

Cheng, H., Tan, P.-N., Gao, J., and Scripps, J. (2006). Multistep-ahead time series prediction. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 765–774. Springer.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv: 1406.1078.

Chollet, F. (2015). Keras. Retrieved from https://keras.io/.

Cleveland, R. B., Cleveland, W. S., McRae, J. E., and Terpenning, I. (1990). STL: A seasonal-trend decomposition. *Journal of Official Statistics*, 6(1):3–73.

Collopy, F. and Armstrong, J. S. (1992). Rule-based forecasting: Development and validation of an expert systems approach to combining time series extrapolations. *Management Science*, 38(10):1394–1414.

Cools, M., Moons, E., and Wets, G. (2009). Investigating the variability in daily traffic counts through use of ARIMAX and SARIMAX models: assessing the effect of holidays on two site locations. *Transportation Research Record*, 2136(1):57–66.

Da Veiga, C. P., Da Veiga, C. R. P., Catapan, A., Tortato, U., and Da Silva, W. V. (2014). Demand forecasting in food retail: a comparison between the Holt-Winters and ARIMA models. *WSEAS Transactions on Business and Economics*, 11(1):608–614.

Dagum, E. B. and Bianconcini, S. (2016). *Seasonal adjustment methods and real time trend-cycle estimation*. Springer International Publishing.

Diebold, F. X. and Mariano, R. S. (2002). Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 20(1):134–144.

Do, L. N. N., Taherifar, N., and Vu, H. L. (2019). Survey of neural network-based models for short-term traffic state prediction. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(1).

Domingos, S. d. O., de Oliveira, J. F., and de Mattos Neto, P. S. (2019). An intelligent hybridization of ARIMA with machine learning models for time series forecasting. *Knowledge-Based Systems*, 175:72–86.

Dougherty, C. (2011). *Introduction to econometrics*. Oxford University Press.

Facebook Open Source (n.d.). Prophet. Retrieved from http://facebook.github.io/prophet/.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*. Springer.

Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471.

GitHub (n.d.a). Canadian Lynx. Retrieved from https://vincentarelbundock.github.io/Rdatasets/csv/datasets/lynx.csv.

GitHub (n.d.b). Sunspot. Retrieved from https://vincentarelbundock.github.io/Rdatasets/csv/datasets/sunspot.year.csv.

Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT Press. Retrieved from: http://www.deeplearningbook.org.

Goyal, P., Pandey, S., and Jain, K. (2018). *Deep learning for natural language processing*. Apress.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2017). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232.

Gupta, D. (2018). *Applied analytics through case studies using SAS and R: Implementing predictive models and machine learning techniques*. Apress.

Harris, R. I. D. (1992). Testing for unit roots using the augmented Dickey-Fuller test: Some issues relating to the size, power and the lag structure of the test. *Economics Letters*, 38(4):381–386.

Harvey, A. C. (1993). *Time series models*. Harvester Wheatsheaf, 2nd edition.

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian Journal of Information Systems*, 19(2):7.

Hewamalage, H., Bergmeir, C., and Bandara, K. (2019). Recurrent neural networks for time series forecasting: Current status and future directions. arXiv: 1909.00590.

Hipel, K. W. and McLeod, A. I. (1994). *Time series modelling of water resources and environmental systems*. Elsevier.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10.

Hu, M. Y., Zhang, G., Jiang, C. X., and Patuwo, B. E. (1999). A cross-validation analysis of neural network out-of-sample performance in exchange rate forecasting. *Decision Sciences*, 30(1):197–216.

Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts, 2nd edition. Retrieved from https://otexts.com/fpp2/.

Hyndman, R. J. and Athanasopoulos, G. (2019). *Forecasting: principles and practice*. OTexts, 3rd edition. Retrieved from https://otexts.com/fpp3/.

Hyndman, R. J. and Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688.

Januschowski, T., Gasthaus, J., Wang, Y., Salinas, D., Flunkert, V., Bohlke-Schneider, M., and Callot, L. (2020). Criteria for classifying forecasting methods. *International Journal of Forecasting*, 36(1):167–177.

Kaggle (n.d.a). Airline Passengers. Retrieved from https://www.kaggle.com/rakannimer/air-passengers#AirPassengers.csv.

Kaggle (n.d.b). Daily Minimum Temperatures Melbourne. Retrieved from https://www.kaggle.com/paulbrabban/daily-minimum-temperatures-in-melbourne#daily-minimum-temperatures-in-me.csv.

Kaggle (n.d.c). Daily Total Female Births California. Retrieved from https://www.kaggle.com/dougcresswell/daily-total-female-births-in-california-1959.

Kaggle (n.d.d). GBP USD Daily Exchange Rate. Retrieved from https://www.kaggle.com/thebasss/currency-exchange-rates.

Kaggle (n.d.e). Rossmann Store Sales. Retrieved from https://www.kaggle.com/c/rossmann-store-sales/data.

Khandelwal, I., Adhikari, R., and Verma, G. (2015). Time series forecasting using hybrid ARIMA and ANN models based on DWT decomposition. *Procedia Computer Science*, 48(1):173–179.

Kim, J., El-Khamy, M., and Lee, J. (2017). Residual LSTM: Design of a deep recurrent architecture for distant speech recognition. arXiv: 1701.03360.

Kolassa, S. and Schütz, W. (2007). Advantages of the MAD/MEAN ratio over the MAPE. *Foresight: the International Journal of Applied Forecasting*, (6):40–43.

Krogh, A. and Vedelsby, J. (1995). Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems*, pages 231–238.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Levene, H. (1960). Levene test for equality of variances. *Contributions to probability and statistics*, pages 278–292.

M Open Forecasting Center (MOFC) (2017). M4 Competition - competitor's guide: prizes and rules. Retrieved from: https://www.m4.unic.ac.cy/wp-content/uploads/2018/03/M4-Competitors-Guide.pdf.

Makridakis, S. (2017). The forthcoming artificial intelligence (AI) revolution: Its impact on society and firms. *Futures*, 90:46–60.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2018a). The M4 Competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2018b). Statistical and machine learning forecasting methods: Concerns and ways forward. *PloS one*, 13(3).

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2020). The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.

Michelucci, U. (2018). *Applied deep learning: a case-based approach to understanding deep neural networks*. Apress.

Nielsen, M. A. (2015). *Neural networks and deep learning*, volume 2018. Determination press.

Olah, C. (2015). Understanding LSTM networks. Retrieved from: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Palit, A. K. and Popovic, D. (2006). *Computational intelligence in time series forecasting: theory and engineering applications*. Springer Science & Business Media.

Panigrahi, S. and Behera, H. (2017). A hybrid ETS-ANN model for time series forecasting. *Engineering Applications of Artificial Intelligence*, 66:49–59.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in neural information processing systems 32*, pages 8026–8037.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830.

Pegels, C. C. (1969). Exponential forecasting: some new variations. *Management Science*, pages 311–315.

Priestley, M. B. (1988). Non-linear and non-stationary time series analysis. *nlns*.

Prudêncio, R. B. and Ludermir, T. B. (2004). Meta-learning approaches to selecting time series models. *Neurocomputing*, 61:121–137.

Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., and Cottrell, G. (2017). A dual-stage attention-based recurrent neural network for time series prediction. arXiv: 1704.02971.

Qiu, X., Zhang, L., Ren, Y., Suganthan, P. N., and Amaratunga, G. (2014). Ensemble deep learning for regression and time series forecasting. In *2014 IEEE symposium on computational intelligence in ensemble learning (CIEL)*, pages 1–6. IEEE.

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions. arXiv: 1710.05941.

Redd, A., Khin, K., and Marini, A. (2019). Fast ES-RNN: A GPU implementation of the ES-RNN algorithm. arXiv: 1907.03329.

Reid, D. J. (1972). A comparison of forecasting techniques on economic time series. *Forecasting in Action. Operational Research Society and the Society for Long Range Planning*.

Rice, J. R. (1976). The algorithm selection problem. In *Advances in computers*, volume 15, pages 65–118. Elsevier.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386.

Rumelhart, D. E., Hintont, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Salehinejad, H., Sankar, S., Barfett, J., Colak, E., and Valaee, S. (2018). Recent advances in recurrent neural networks. arXiv: 1801.01078.

SciPy 1.0 Contributors, Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, I., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., and van Mulbregt, P. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272.

Seabold, S. and Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with python. In *Proceedings of the 9th Python in science conference*, pages 92–96.

Shcherbakov, M. V., Brebels, A., Shcherbakova, N. L., Tyukov, A. P., Janovsky, T. A., and Kamaev, V. A. (2013). A survey of forecast error measures. *World Applied Sciences Journal*, 24(24):171–176.

Shumway, R. H. and Stoffer, D. S. (2017). *Time series analysis and its applications: with R examples*. Springer International Publishing.

Siregar, B., Butar-Butar, I. A., Rahmat, R., Andayani, U., and Fahmi, F. (2017). Comparison of exponential smoothing methods in forecasting palm oil real production. *Journal of Physics: Conference Series*, 801(1).

Smyl, S. (2017). Ensemble of specialized neural networks for time series forecasting. Retrieved from https://bit.ly/3fj2owR.

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85.

Snedecor, G. W. and Cochran, W. G. (1989). Statistical methods. *Ames: Iowa State Univ. Press Iowa*, 54:71–82.

Sorjamaa, A. and Lendasse, A. (2006). Time series prediction using DirRec strategy. *Esann*, 6:143–148.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Stan (n.d.). Stan. Retrieved from https://mc-stan.org/.

Taieb, S. B., Bontempi, G., Atiya, A. F., and Sorjamaa, A. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Systems with Applications*, 39(8):7067–7083.

Taieb, S. B., Bontempi, G., Sorjamaa, A., and Lendasse, A. (2009). Long-term prediction of time series by combining direct and MIMO strategies. In *2009 international joint conference on neural networks*, pages 3054–3061. IEEE.

Talagala, T. S., Hyndman, R. J., and Athanasopoulos, G. (2018). Meta-learning how to forecast time series. *Monash Econometrics and Business Statistics Working Papers*, 6:18.

Taylor, R. (n.d.). PyFlux. Retrieved from https://pyflux.readthedocs.io/en/latest/.

Taylor, S. J. and Letham, B. (2018). Forecasting at scale. *The American Statistician*, 72(1):37–45.

Torrey, L. and Shavlik, J. (2010). Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. (2017). Feudal networks for hierarchical reinforcement learning. arXiv:1703.01161.

Wang, X., Smith-Miles, K., and Hyndman, R. (2009). Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series. *Neurocomputing*, 72(10-12):2581–2594.

Werbos, P. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Willer, H., Moeskops, B., Busacca, E., and De La Vega, N. (2019). Organic in europe: recent developments. In *The world of organic agriculture. Statistics and emerging trends 2019*, pages 208–216.

Winters, P. R. (1960). Forecasting sales by exponentially weighted moving averages. *Management Science*, 6(3):324–342.

Wolpert, D. and Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

Yaffee, R. A. and McGee, M. (2000). *An introduction to time series analysis and forecasting: with applications of SAS®and SPSS®*. Elsevier.

Zalando Research (n.d.). PyTorch Dilated Recurrent Neural Networks. Retrieved from https://github.com/zalandoresearch/pytorch-dilated-rnn.

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, 50:159–175.