MACHINE LEARNING AND MACROECONOMICS: FORECASTING EUROPEAN UNEMPLOYMENT ON THE USE OF MACHINE LEARNING TO FORECAST MACROECONOMIC INDICATORS

Word count: 17.071

Karel Pype
Student number: 000150360811

Supervisor: Prof. Dr. Koen Schoors

Master’s Dissertation submitted to obtain the degree of:

Master in Economics

Academic year: 2018-2019
PERMISSION

I declare that the content of this Master’s Dissertation may be consulted and/or reproduced, provided that the source is referenced.

Name student: Karel Pype

Signature:
“The difficulty lies, not in the new ideas, but in escaping from the old ones, which ramify, for those brought up as most of us have been, into every corner of our minds.”

Keynes, 1936
Abstract

For over half a century, econometrics has been the main tool for economists to confirm or undermine their models. However, the rise of Machine Learning and Deep Neural Networks in recent years has meant a true change in the way conventional econometrics is perceived as the ultimate verification tool. Especially when it comes to forecasting methods, Machine Learning more often than not provides more accurate results. In this paper, we will set a clear distinction between machine learning and econometrics, focusing on their ability to provide predictions and forecasts concerning macroeconomic indicators. Afterwards, we test our theoretical experiences with the help of a specific application, namely the forecasting of European unemployment. Furthermore, we will discuss taking Machine Learning to the next level, where it doesn’t only account for accurate forecasting but also for the investigation of causality between macroeconomic variables.

Acknowledgements

I would like to thank prof. dr. Schoors for believing in me and guiding me through this process. I would also like to thank Benjamin Callewaert, Sander Noels, Henri Van den Braembussche, and Anton Van Goethem for helping me with the technicalities in Python and being the best friends possible.
# Table of Contents

**INTRODUCTION** ........................................................................................................... 1

**MACHINE LEARNING** ................................................................................................. 2

A SHORT INTRODUCTION TO ML ...................................................................................... 2
A SHORT INTRODUCTION TO DL ....................................................................................... 8

**MACHINE LEARNING VERSUS ECONOMETRICS** ......................................................... 11

**MACHINE LEARNING, CONSENSUS FORECASTS AND STATISTICAL FORECASTS** ................................................................. 14
MACHINE LEARNING AND CAUSAL INFERENCE .............................................................. 23

**MACROECONOMIC FORECASTING** ............................................................................... 25

A SHORT HISTORY ............................................................................................................ 28
THE USE OF ML TO PREDICT AND FORECAST MACROECONOMIC INDICATORS .............. 30
USING THE RIGHT ESTIMATOR ......................................................................................... 30
REVIEWING THE EXISTING LITERATURE ON MACROECONOMIC FORECASTING WITH ML ................................................................. 32
THE USE OF DL TO PREDICT AND FORECAST MACROECONOMIC INDICATORS .............. 34
MAJOR MODELS .............................................................................................................. 37
REVIEWING THE EXISTING LITERATURE ON MACROECONOMIC FORECASTING WITH DL ................................................................. 41

**APPLICATION: FORECASTING EUROPEAN UNEMPLOYMENT WITH ML** ...................... 44

EMPLOYMENT IN EUROPE: A SHORT OVERVIEW ............................................................. 44
THE DATA ......................................................................................................................... 46
THE SET-UP .................................................................................................................... 50
RESULTS .......................................................................................................................... 51

**CONCLUSION AND AVENUES FOR FURTHER DEVELOPMENT** ................................... 54

**SOURCES** ..................................................................................................................... 57

**APPENDIX** .................................................................................................................. A

1A. (STOCHASTIC) GRADIENT DESCENT ................................................................. A
2A. DECISION TREES AND RANDOM FORESTS .................................................. D
3A. BACKPROPAGATION ................................................................................................. F
4A. COOK & HALL (2017): FIGURES OF DIFFERENT ARCHITECTURES USED ............ H
5A. AREA-WIDE MODEL VARIABLES ......................................................................... J
## List of Used Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIC</td>
<td>Akaike information criterion</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>AR</td>
<td>Autoregressive</td>
</tr>
<tr>
<td>ARMA</td>
<td>Autoregressive Moving Average</td>
</tr>
<tr>
<td>AWM</td>
<td>Area-Wide Model</td>
</tr>
<tr>
<td>BIC</td>
<td>Bayesian information criterion</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>CPI</td>
<td>Consumer Price Index</td>
</tr>
<tr>
<td>DL</td>
<td>Deep Learning</td>
</tr>
<tr>
<td>DSGE</td>
<td>Dynamic Stochastic General Equilibrium</td>
</tr>
<tr>
<td>E-D</td>
<td>Encoder-Decoder</td>
</tr>
<tr>
<td>EA-19</td>
<td>Euro Area - 19 Countries</td>
</tr>
<tr>
<td>ECB</td>
<td>European Central Bank</td>
</tr>
<tr>
<td>EU-28</td>
<td>European Union - 28 Countries</td>
</tr>
<tr>
<td>FFANN</td>
<td>Feed-Forward Artificial Neural Network</td>
</tr>
<tr>
<td>FRED-MD</td>
<td>Federal Reserve Bank of St. Louis – Monthly Database</td>
</tr>
<tr>
<td>GD</td>
<td>Gradient Descent</td>
</tr>
<tr>
<td>GDP</td>
<td>Gross Domestic Product</td>
</tr>
<tr>
<td>LASSO</td>
<td>Least Absolute Shrinkage and Selection Operator</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long Short Term Memory</td>
</tr>
<tr>
<td>MA</td>
<td>Moving Average</td>
</tr>
<tr>
<td>MAE</td>
<td>Mean Absolute Error</td>
</tr>
<tr>
<td>MAPE</td>
<td>Mean Absolute Percentage Error</td>
</tr>
<tr>
<td>ME</td>
<td>Mean Error</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>MLP</td>
<td>Multilayer Perceptron</td>
</tr>
<tr>
<td>MPE</td>
<td>Mean Percentage Error</td>
</tr>
<tr>
<td>OLS</td>
<td>Ordinary Least Squares</td>
</tr>
<tr>
<td>RHS</td>
<td>Right-Hand Side</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root-Mean-Square Error</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent Neural Network</td>
</tr>
<tr>
<td>RSS</td>
<td>Residual Sum of Squares</td>
</tr>
<tr>
<td>SGD</td>
<td>Stochastic Gradient Descent</td>
</tr>
<tr>
<td>SPF</td>
<td>Survey of Professional Forecasters</td>
</tr>
<tr>
<td>SVC</td>
<td>Support Vector Classification</td>
</tr>
<tr>
<td>SVR</td>
<td>Support Vector Regression</td>
</tr>
<tr>
<td>VAR</td>
<td>Vector Autoregressive</td>
</tr>
<tr>
<td>VBGMM</td>
<td>Variational Bayesian Gaussian Mixture</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1: Supervised Machine Learning (source: Vaseekaran (2018)) .................................................. 4
Figure 2: Unsupervised Machine Learning (source: Vaseekaran (2018)) ................................................. 5
Figure 3: Taxonomy of Machine Learning and Corresponding Problem Types ....................................... 5
Figure 4: Three Stages of Model Generation in Machine Learning (source: Chakraborty & Joseph (2017)) ...... 7
Figure 5: Example of a Feedforward Network with Multiple Layers (source: da Silva et al. (2017)) .......... 8
Figure 6: Graphical Representation of a Feedforward Artificial Neural Network (FFANN) within the Inflation Forecasting Framework (source: Chakraborty & Joseph (2017)) .................................................. 10
Figure 7: Errors of a Model due to Bias and Variance (source: Hall (2018)) ............................................ 18
Figure 8: Trade-Off between Errors due to Bias and Errors due to Variance (source: Hall (2018)) ............ 18
Figure 9: Overfitting and Underfitting (source: Singh, 2018) ................................................................. 19
Figure 10: Residual Sum of Squares (Cost Function for OLS Regression) .................................................. 20
Figure 11: Ridge Regression Estimator ........................................................................................................ 21
Figure 12: LASSO Estimator ......................................................................................................................... 22
Figure 13: Elastic Net Estimator ................................................................................................................... 22
Figure 14: Choosing the Right Estimator (source: scikit-learn) ............................................................... 32
Figure 15: Biological Neuron Graph (left) & Artificial Neural Network (right) (source: Ghallou, 2017) .... 35
Figure 16: The Perceptron (source: Cook & Hall (2017)) ......................................................................... 36
Figure 17: An Example of the CNN Architecture (source: Hidaka & Kurita (2017)) ................................. 38
Figure 18: An Example of the Encoder-Decoder Architecture (source: Fritz (2016)) ................................... 40
Figure 19: Unemployed persons, in millions, seasonally adjusted, EU-28 and EA-19, January 2000 - June 2019 45
Figure 20: Handling Missing Data ............................................................................................................. 49
Figure 21: Actual Unemployment (blue) versus Forecasted Unemployment (orange) at 1-Quarter Horizon .... 53
Figure 22: Actual Unemployment (blue) versus Forecasted Unemployment (orange) at 4-Quarters Horizon 53
Figure 23: Coefficients in the Elastic Net model at 1-Quarter Horizon .......................................................... 54
Figure 24: Coefficients in the Elastic Net model at 4-Quarter Horizon ......................................................... 54
Figure 25: Gradient Descent (source: Suryansh (2018)) ......................................................................... B
Figure 26: Big Learning Rate (left) versus Small Learning Rate (right) (source: Donges (2019)) ................... C
Figure 27: The Terminology used for Decision Trees ................................................................................. D
Figure 28: Random Forest with n Decision Trees for a Regression Task (source: Kumar, 2018) ................. F
Figure 29: Feed-Forward (Fully Connected) Model Architecture ............................................................ H
Figure 30: Convolutional Model Architecture ........................................................................................... H
Figure 31: LSTM Model as Tested ............................................................................................................... I
Figure 32: Encoder-Decoder Diagram ....................................................................................................... I
Figure 33: AWM Variables (source: ECB (2018)) ................................................................................... J

List of Tables

Table 1: Econometrics versus Machine Learning .................................................................................. 13
Table 2: Results of the SPF in Comparison with Other Expectations and Projections (source: ECB SPF Q2 2019) 16
Table 3: Time Series (left) versus Supervised Learning Problem (right) (based on Brownlee (2016)) ......... 47
Table 4: Predicting MEASURE 2 with Multivariate Time Series (based on Brownlee (2016)) ................. 48
Table 5: MAE Results for all Horizons with each of the Three Methods .............................................. 51

VI
Introduction

44 trillion gigabytes\(^1\). This is the estimated accumulated amount of data that will be available by 2020 (Turner et al., 2014). If this does not appeal to the imagination, look at it in this way: by 2020, there will be nearly as many digital bits as there are stars in the universe. This trend, of ever-increasing granular data, is referred to as the rise of Big Data and is accompanied by another mega-trend: the massive rise of data-driven modelling techniques, or Machine Learning (Chakraborty & Joseph, 2017). These new techniques provide great solutions for some of the issues encountered using conventional statistical and econometric techniques and will shape the future of data-analysis (Varian, 2014).

With great interest for the assessment of results of a specific policy or the causal relationship in certain theories, macroeconomics has always known a rich history of empirical evaluation and validation. Especially since the mid-1980s, the number of empirical papers has skyrocketed (Hamermesh, 2013). Traditional econometric techniques have always been prominent here and, accompanied by statistical inference to assess the significance of these effects, were devoted to the identification of causal effects.

However, making its way into the macroeconomic analysis via a better predictive fit, applications built on Machine Learning are gaining ground. Here, the most striking difference becomes immediately apparent. In conventional statistics, or econometrics in this case, everything revolves around causal inference. It is a policy analysis, where we compare the effects of a policy (or general treatment) with its counter-factual (Chakraborty & Joseph, 2017). For Machine Learning, on the other hand, the emphasis lies more on a predictive fit. Via data-driven model selection, we try to identify the most meaningful predictive variables. Statistical uncertainty and standard errors fade to the background and more attention is spent on model uncertainty instead.

In this dissertation, readers new to these concepts will be gradually introduced to the fundamentals of these new analytical tools, centred around Machine Learning. We will give a

\(^1\) Or 44 zettabytes.
clear-cut definition of Machine Learning and show how this essentially differs from conventional econometrics. The emphasis in this essay will always be put on macroeconomics. This is why the bigger part of this essay will evolve around the use of Machine Learning to predict or forecast\(^2\) macroeconomic indicators. We will look at the past, present, and future of macroeconomic forecasting. In the end, a special-case study in which we will try to forecast European unemployment will be presented. This will be done using a linear Machine Learning estimator: the *Elastic Net* model.

**Machine Learning**

Data is big. Inspecting, cleansing, transforming, modeling. On a daily basis, thousands of data analysts run through this cycle in the pursuit of obtaining better measurements of economic effects and their outcomes. Consumer demand, cross-country GDP differences, firm sizes, labour market opportunities, ... As the volume of the available data grows, new techniques become increasingly interesting (Einav & Levin, 2014). This is why, next to traditional econometric techniques, new large-data statistical methods have gained popularity. One of these methods is Machine Learning. In this section, we will introduce the novice reader into the wonderful world of Machine Learning, step by step. We will give a clear-cut definition of Machine Learning and prepare you for the technical analysis ahead.

**A Short Introduction to ML**

There are many definitions for Machine Learning (ML) in today's context. However, for this paper, we will predicate ourselves upon a rather narrow definition. As Athey (2018) states, this precarious definition is important to distinguish ML from other regression-based econometric approaches used in applied econometrics. “*Machine learning is a field that develops algorithms designed to be applied to datasets, with the main areas of focus being prediction (regression), classification, and clustering or grouping tasks*” (Athey, 2018). Let’s pretend we have a well-cleansed dataset of different countries’ energy resource mix (input parameters) and their

\(^2\) There is an important distinction between forecasting and predicting, more on this later.
economic growth as measured by GDP (output parameter). We should then be able to predict the respective country’s economic development based on a self-learning algorithm (Cogoljević et al., 2018). We will elaborate on the technical specifics of the above at a later stage in this dissertation.

We can distinguish two main branches (or two types of learning algorithms) in ML: supervised and unsupervised ML (Athey, 2018). Yet, some also identify a third learning style: reinforcement learning. The most popular and most applied in practical Machine Learning is supervised ML. In supervised machine learning you develop an algorithm, which learns the mapping function from the input \((x)\) to the output \((y)\). When this mapping function \((Y=f(X))\) is well-estimated, you should be able to predict output variables \((Y)\) using new input data \((X)\) (Brownlee, 2016). To illustrate, the unemployment rate is our output variable \((Y)\) and other macroeconomic variables, such as GDP per capita and the inflation rate are our input variables \((X)\). We could develop an algorithm that learns how the unemployment rate reacts to these macroeconomic variables. When we then eventually feed new macroeconomic data to the model, it should be able to ‘map’ this information to a predicted unemployment rate. Brownlee (2016) clarifyingly compares the process of an algorithm learning from the training dataset with a teacher supervising the learning process of children. This is where the term ‘supervised’ comes from. The teacher knows the correct answers and accompanies the algorithm until it achieves an acceptable level of performance. An important factor here is that all data should be labeled, and you have a clear distinction between input and output variables. Further grouping this learning style, we can characterize classification and regression problems. Respectively, we then talk about problems where the output variable is either a category (e.g.: man/woman) or a real value (e.g.: euros).

Figure 1 shows how Supervised Machine Learning works. Developing an algorithm which learns the mapping function from input \((x)\) to output \((y)\), basically means using a training data set (cf. infra) with known outputs to learn these examples of situations. Based on this training data, the ‘machine’ (i.e. Machine Learning) will then build the model which can predict the outcome of the new data based on the past examples.
A second learning style is unsupervised Machine Learning. The big difference in this learning style is that all data are unlabeled, and we only have input data (X). The goal of unsupervised learning is to form clusters of observations with similar characteristics, which results in finding the underlying structure in the data. Most common applications are bundling images, articles, etc. with the same properties. The analogy of the teacher cannot be used here. There are no correct answers and thus no need for a “teacher”: it’s up to the algorithm to discover and present the interesting structure in the data. Again, we are able to characterize two problems, clustering and association. Respectively, we choose to discover the inherent groupings in the data (e.g.: group countries by economic development) or the rules that describe large portions of the data (e.g.: people that buy X also tend to buy Y). For the continuation of this paper, the main focus won’t be on unsupervised learning. It’s good to remember, though, that this method can be very useful in intermediate steps as they provide a data-driven way to find similarities.

Figure 2 gives a visual overview of unsupervised Machine Learning. You just get a bunch of unlabeled data and it is up to the algorithm to group this data according to its most prominent characteristics. There are no examples of desired output, or better, there is no target Y given. Figure 3 summarizes the two problem types.
Finally, a third type of ML is reinforcement learning. In this case, the model that you build doesn’t get feedback right away, it only gets feedback if it achieves an explicit goal. We call these goal-oriented algorithms. Moreover, these algorithms get penalized when they make the wrong decisions and rewarded when they make the correct ones. This is why we call it reinforcement learning. On a more mathematical level, you could imagine a situation where new inputs (X) are constantly arriving, but there is no exact output (Y) to match them (Chakraborty & Joseph, 2017). Alternatively, given assumptions are constantly updated sequentially with the aim of maximizing a pay-off function.
Blockeel (2018) beautifully explains the distinction between supervised, unsupervised and reinforcement learning using the example of a bot that learns to beat humans in chess. Using supervised learning, it would receive feedback for every move, using unsupervised learning we would never get feedback. If it uses reinforcement learning, it would only receive feedback if it won the game. Unlike (un)supervised learning, reinforcement learning is a trial-and-error based approach. For example, AlphaGo³ is based on reinforcement learning.

Let’s recapitulate. What is so special about Machine Learning? In Machine Learning, we no longer have to worry about explicitly programming, writing code, building models based on theory, etc. Instead, algorithms learn from examples and experience. They build logic based on the data given. The data take the central role in understanding. As said in the introduction, it is the rise in Big Data that has started this promising trend in Machine Learning. Hereby, it is of huge importance to understand how these data are used. Figure 4 shows this. It’s been hinted when writing about supervised ML (cf. supra) that we will need some training data. Simply put, this is the sample of data used to fit (or train) the model (Brownlee, 2017). Furthermore, part of the dataset is used to calibrate the model. We call this the validation data⁴. It is the sample of data used to provide an unbiased evaluation of a model while tuning model hyperparameters⁵ (Brownlee, 2017). From time to time, a given model is evaluated. This happens frequently. The model occasionally sees this sample of the data, but never learns from it (Shah, 2017). Finally, we have the testing data, which is used for model evaluation. It is the sample of data used to provide an unbiased evaluation of a final model fit on the training dataset (Brownlee, 2017).

In the figure below (figure 4), the dataset split ratios are 60% of your sample for training of the model, 20% for validation, and 20% for model evaluation. Note however that these dataset split ratios are no rules-of-thumb. How you choose which percentages to use will depend on the number of instances (or the amount of data), and the actual model you are training. For example, if you have a model with a lot of hyperparameters, you’ll need a large validation set. More on this later in the dissertation.

³ The first computer programme to defeat a professional human Go player. Courtesy of DeepMind.
⁴ Sometimes, the validation set is split from the training set. This is called Cross-Validation (cf. infra).
⁵ For now, just know that we set these parameters before the learning process begins. You cannot learn these parameters directly from the regular training process. Hyperparameters express “higher-level” properties of the model such as its complexity or how fast it should learn.
To compete this section, it can already be interesting to show how these ML methods differ from traditional econometric and statistical methods. This contrast should also help us better understand what ML actually is. ML methods use *data-driven model selection*. Whereas in traditional econometrics a model is built based on the underlying theory, in ML the analyst provides the list of variables, but the functional form is at least partly determined as a function of the data (Athey, 2017). An algorithm estimates many alternative models instead of only one, and then chooses one based on several criteria. We will dive further into this distinction in the next section. For now, it is important to remember that ML is a data-driven model selector rather than an “author-driven” model selector.

We have now introduced you to the very basics of Machine Learning. A clear-cut definition for Machine Learning was given, and the three kinds of ML algorithms (supervised, unsupervised, and reinforcement learning) were discussed. Furthermore, I told you how the data – which is of huge importance for these problems – are used. Finally, the most important difference between econometrics and Machine Learning has already been tackled.

*Figure 4: Three Stages of Model Generation in Machine Learning (source: Chakraborty & Joseph (2017))*
A Short Introduction to DL

Let’s dive deeper into the most prominent sub-field of Machine Learning: Deep Learning (DL). Inspired by the functionality of our brain cells, or neurons, DL relies on an Artificial Neural Network (ANN). The aim of this Network is to mimic human decision-making capabilities as well as possible. Thus, in Deep Learning the data are being analyzed in a logic structure and conclusions are made in the same way humans would draw conclusions. In this subsection, the basis of Deep Learning is explained. A deeper technical meaning is presented in the section on The Use of DL to Predict and Forecast Macroeconomic Indicators.

To be able to process (or analyze) this massive amount of data, abstraction is needed. The DL model structures algorithms in different layers (da Silva, 2017): The Input Layer, the Hidden Layer and the Output Layer. Each of these (hidden) layers learns specific and increasingly complex features of the data, and processes them to develop a self-learning mechanism. The Input Layer receives the information, or the data. The Hidden Layers, also called intermediate or invisible layers, represent the neurons responsible for abstracting patterns associated with the process or system being analyzed. Their job is to transform the inputs into something the output layer can use. Finally, the Output Layer represents the final neurons, the result of applying different functions in the Hidden Layers to the input. Figure 5 illustrates an Artificial Neural Network (ANN) with an input layer, two hidden layers, and an output layer. For now, don’t worry about the specific properties of this ANN, this just serves as a graphical example.

Figure 5: Example of a Feedforward Network with Multiple Layers (source: da Silva et al. (2017))
When speaking about DL models, we generally do not refer to equations or systems of equations. Unlike traditional forecasting models, Neural Networks were developed in a tradition similar to graphical models (Cook & Hall, 2017). We call this the model architectures. This architecture defines how these so-called neurons are arranged, or placed, in relation to each other (da Silva et al., 2017). As a way of introduction to the different architectures, we will discuss the most important architecture here as an example, belonging to the class **Feedforward Networks**, called **multi-layer perceptrons**.

Chakraborty and Joseph (2017) used a Feedforward Artificial Neural Network (FFANN) to forecast inflation. The underlying graphical representation is showed in *Figure 6*. The input layer consists of a variety of macroeconomic variables (i.e. GDP, labour product, employment rate, ...). Each feature (and not all are shown here) is represented by a node on the left-hand side. These inputs are processed in a step-wise fashion moving through the hidden layer(s) (red) till reaching the output layer (blue) on the RHS. A weight matrix ($W_1$) connects the inputs to the hidden layer in the center of the network. It’s the weights that capture the knowledge of a Neural Network with regard to the world. The weights are the parameters that modify input data, as their signal flows through the neural network. At each node in the hidden layer, the joint signal from the input layer is evaluated by an *activation function* to generate a standardized output as the next layer’s input. Increasingly complex features of the macroeconomic variables are being translated in CPI (inflation). We call these internal inputs *derived features*. In other words, a combined signal is engendered from the input features of the previous layer and a (non-linear) transformation through the activation function. Next, the weight matrix $W_2$ connects the derived features to the output layer. The model parameters establish the links in the network and are contained in the weight matrices $W_1$ and $W_2$. On top, intercepts are represented via the “bias nodes” (gray). These represent a bias for inactivity. It says how high the weighted sum needs to be, before the neuron starts getting meaningfully active. For a more elaborate disquisition, I refer to ‘Case 2’ in Chakraborty and Joseph’s paper on *Machine Learning at Central Banks* (2017).

---

6 Other architectures are discussed in the section on *The Use of DL to Predict and Forecast Macroeconomic Indicators*.

7 Here this is a so-called *Sigmoid function*. 
While this all seems complicated now, an increasingly well understanding will be developed throughout the dissertation. Especially once we have a look at the use of DL to forecast macroeconomic indicators, the different architectures and their associated algorithms will be clarified.

What actually makes the difference between ML and DL? Technically, Deep Learning is Machine Learning. It learns and functions in similar ways, but its capabilities are different (Grossfeld, 2017). Deep Learning outperforms traditional ML approaches, because of its aptitude to process a wider range of data resources, because it requires fewer data processing by humans, and because it provides more precise results (Chui, Kamalnath & McCarthy). Once the Deep Learning algorithm is finished, it won’t need human interference anymore. The algorithms determine on their own if the predictions are accurate or not. On the other hand, with Machine Learning, if the algorithm doesn’t return accurate predictions, the engineer will need to step in and make adjustments.
Even though the main purpose of this paper is to forecast macroeconomic indicators using Machine Learning, a short introduction to Deep Learning was also presented here. We discussed the very basics of Deep Learning and Artificial Neural Networks and tried to clarify these theoretical findings with the help of a macroeconomic example. In the end, we also explained the similarities and differences between Machine Learning and Deep Learning.

**Machine Learning versus Econometrics**

If you want to understand causal relationships between different aspects of the economy, standard econometric models are the way to go (Merler, 2018). Using the right economic theory, we are able to perfectly quantify the causal relationship between dependent (e.g. inflation rate) and independent (e.g. unemployment rate) variables. They serve as supreme policy targeting instruments. Nevertheless, if prediction is the main aim, other techniques (such as Machine Learning) become increasingly interesting and often imply solutions to the traditional econometric shortcomings. In this section we will elaborate on the main differences between ML and conventional empirical work in economics.

To start off, there is already a historical difference between the two (Boelaert & Ollion, 2018). ML techniques heavily rely on computing power, opposed to the traditional statistics, which were developed in times where heavy computing power wasn’t that common. This of course has had some huge consequences. Small samples and heavy assumptions about data and its distributions drive traditional statistics. In ML, on the other hand, less assumptions are needed in advance. There is a certain freedom in finding the solution, and this if often based on *inductive learning*. And for inductive learning, plenty of data are needed (and not necessarily a lot of prior knowledge).

As implied in the introduction, the bulk of the contradiction is about causality versus prediction (yet, this contradiction is disappearing, cf. infra). Predictive performance is the key performance

---

8 Notice that this does not mean that you cannot perform predictions with econometrics, or causal inference with machine learning!
9 Simply said, learning by examples.
indicator in Machine Learning models. We try to obtain an as low as possible generalization error, meaning that we want the algorithm to perform as good as possible when applied to previously unseen data. We use data-driven model selection to identify the most meaningful predictive variables (Einav & Levin, 2014). Less attention is given to statistical uncertainty and standard errors, more attention to model uncertainty. On the other hand, if we are more interested in the “why” instead of knowing where things are going next, econometrics are the way to go. Focusing on the “why” is called causality. With finding causal relationships as main objective, hypothesis testing has a central role in econometrics. It is the statistical way that is much more important than computational techniques.

This preference for causal inference in econometrics brings about some important characteristics of econometric research. It is to say that econometric research often seeks to reduce goodness-of-fit of a model in order to estimate the causal effect (Bogard, 2016). On the contrary, ML techniques are said to ‘let the data speak’. This in turn implies that econometrics more often than not explore linear relationships. In contrast, linear relationships are rarely found in ML (Zheng at al., 2017). They are more prone to combining different models (e.g. aggregation of five different models).

Apart from our distinction based on the applications of both techniques - econometrics is for causal inference and Machine Learning is for prediction - we also have a model specification-based distinction (Woloszko, 2017). Econometrics are said to be model-based. You start with a certain idea of how the ‘world’ works, a certain theory, and afterwards(!) you use the data to calibrate your model. For Machine Learning this is different. ML has a data-first approach, meaning that you let algorithms uncover hidden patterns in the data. These patterns might even not be supported by a theory yet!

Very brief, how do cross-sectional econometrics work? First, the researchers specify one model, and one model only. This model is entirely based on economic theory and these correlations/causalities still need to be tested using statistical theory. Afterwards, they estimate the model on the entire dataset and confidence intervals for the (by then) estimated parameters are determined. The focus is on the estimated effects rather than the goodness-of-fit of the
model. Everything the researchers aren’t able to explain (i.e. insignificant parameters) is put in the residuals. They work their way to an acceptable model with only significant parameters.

However, this traditional approach has some disadvantages. Econometric approaches aren’t capable of handling too huge amounts of data, they are best kept simple and relatively easy to interpret (Merler, 2018). With the boom of big data we are experiencing today (Einav & Leniv, 2014), the scope of data analytics can go further than just linear models (to more flexible relationships) (Varian, 2014). Furthermore, researchers often encounter difficulties in model specification. We can also call this *model uncertainty*. Many papers present regression results in a table with several different specifications: new or other instrument variables, extra dummies to account for fixed effects, etc.

**Table 1** represents our main findings of this section.

<table>
<thead>
<tr>
<th></th>
<th><strong>Econometrics</strong></th>
<th><strong>Machine Learning</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Approach</strong></td>
<td>Statistical</td>
<td>Algorithmic Model</td>
</tr>
<tr>
<td><strong>Driver</strong></td>
<td>Theory</td>
<td>Fitting the Data</td>
</tr>
<tr>
<td><strong>Focus</strong></td>
<td>Hypothesis testing and interpretability</td>
<td>Predictive accuracy</td>
</tr>
<tr>
<td><strong>Model choice</strong></td>
<td>Parameter significance and in-sample goodness-of-fit</td>
<td>Cross-validation(^\text{10}) of predictive accuracy on partitions of data</td>
</tr>
<tr>
<td><strong>Strength</strong></td>
<td>Causal Inference</td>
<td>Prediction</td>
</tr>
</tbody>
</table>

\(^\text{10}\) Cf. infra

Personally, we expect Machine Learning never to fully surrogate traditional econometrics. In commercial banks, for example, they use typical macro models and applied econometrics to capture the bigger picture. It’s a kind of policy system: they look how different lending rates and mortgage practices will change exposure and profitability, but also society’s prosperity. However, Machine Learning is making its way into commercial banks by capturing the small details. These
models are able to discover patterns and links that traditional systems (e.g. FICO credit ranking system) aren’t even capturing. In this way, ML models and traditional econometrics work perfectly complementary instead of being each other’s substitutes.

This leads me to propose some ways of combining ML and econometrics. First off, we could use ML models to feed in the variables that are going to be used in an econometrics model (Zheng et al., 2017). Furthermore, if you have a ML algorithm, fundamental econometric techniques can be borrowed to enhance those of ML. The other way around is also possible. You could use some ML techniques to enhance your econometric model. In the end, both techniques can be used to explain the same phenomenon. These different approaches will give us more accurate results. For example, text mining and sentiment analysis can be used to define different variables. In turn, these variables can become part of your econometric model.

To start this section on Machine Learning versus econometrics, differences in objectives between the two techniques were given. The main objective of ML is said to be predictive accuracy, whilst econometrics’ main objective is to find causal relationships. Furthermore, we briefly touched some other differences and disadvantages of both techniques. In the coming subsections, attention will be spent on various forecasting approaches and brand-new literature on the merger of ML and causal inference.

Machine Learning, Consensus Forecasts and Statistical Forecasts

What follows is a brief introduction to the two traditional ways of forecasting: Statistical Forecasts and Consensus Forecasts. Notwithstanding, the bigger part of this section will revolve around Machine Learning as a new forecasting method and its different estimators.

Traditionally, we can distinguish two ways of forecasting: statistical modelling techniques and consensus forecasting (Hall, 2018). First, we can use statistical modelling techniques typically specialized for time series data. The most basic example for this is the random walk. Here, the prediction will take a random and unpredictable path. Of course, this will be, in most cases, far from accurate and thus often just serve as a control or baseline. Trying to increase the accuracy, we next categorize Autoregressive (AR) models. Simply said, it is the future based on recent
observations. Further, we distinguish models such as Moving Average (MA), Autoregressive (Integrated) Moving Average (ARMA), Vector Autoregression (VAR), etc. We will, however, not get into the technicalities. The interested reader can get an extensive explanation on these models in Gurjarati and Porter’s book on *Basic Econometrics* (2009).

Next to statistical forecasts, one other traditional way of forecasting is consensus forecasting, or *model averaging*. Forecasts of professionals are being aggregated into one consensus prediction (Hall, 2018). In other words, different forecasts using different methodologies are being combined into one aggregate prediction. Most famous examples are the ECB’s *Survey of Professional Forecasters (SPF)* and its American counterpart, the *Blue Chip Economic Indicator*. One huge advantage of this approach is that it does not involve many choices. In contrast to statistical forecasting, where choices concerning model(s) (complexity) need some economic intuition and judgement, consensus forecasting involves much easier choices. Will you use a median or average aggregate? Or will we only average top-performing forecasts?

*Table 2* shows the results of the SPF in comparison with other expectations and projections for the second quarter of 2019 (ECB SPF, 2019). Inflation, real GDP growth, and unemployment are forecasted. This purely serves as an example. Two remarks: (i) longer-term expectations refer to 2023. The forecasts of Consensus Economics and Eurozone Barometer for 2021 and the longer term are taken from the January 2019 surveys; (ii) the unemployment rate is as a percentage of the labour force.
<table>
<thead>
<tr>
<th></th>
<th>Survey horizon</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2019</td>
</tr>
<tr>
<td><strong>HICP inflation</strong></td>
<td></td>
</tr>
<tr>
<td>Q2 2019 SPF</td>
<td>1.4</td>
</tr>
<tr>
<td>Previous SPF (Q1 2019)</td>
<td>1.5</td>
</tr>
<tr>
<td>ECB staff macroeconomic projections (March 2019)</td>
<td>1.2</td>
</tr>
<tr>
<td>Consensus Economics (March 2019)</td>
<td>1.3</td>
</tr>
<tr>
<td>Euro Zone Barometer (March 2019)</td>
<td>1.4</td>
</tr>
<tr>
<td><strong>Memo: HICP inflation excluding energy, food, alcohol and tobacco</strong></td>
<td></td>
</tr>
<tr>
<td>Q2 2019 SPF</td>
<td>1.2</td>
</tr>
<tr>
<td>Previous SPF (Q1 2019)</td>
<td>1.3</td>
</tr>
<tr>
<td>ECB staff macroeconomic projections (March 2019)</td>
<td>1.2</td>
</tr>
<tr>
<td>Consensus Economics (March 2019)</td>
<td>1.2</td>
</tr>
<tr>
<td><strong>Real GDP growth</strong></td>
<td></td>
</tr>
<tr>
<td>Q2 2019 SPF</td>
<td>1.2</td>
</tr>
<tr>
<td>Previous SPF (Q1 2019)</td>
<td>1.5</td>
</tr>
<tr>
<td>ECB staff macroeconomic projections (March 2019)</td>
<td>1.1</td>
</tr>
<tr>
<td>Consensus Economics (March 2019)</td>
<td>1.2</td>
</tr>
<tr>
<td>Euro Zone Barometer (March 2019)</td>
<td>1.2</td>
</tr>
<tr>
<td><strong>Unemployment rate(^2)</strong></td>
<td></td>
</tr>
<tr>
<td>Q2 2019 SPF</td>
<td>7.8</td>
</tr>
<tr>
<td>Previous SPF (Q1 2019)</td>
<td>7.8</td>
</tr>
<tr>
<td>ECB staff macroeconomic projections (March 2019)</td>
<td>7.9</td>
</tr>
<tr>
<td>Consensus Economics (March 2019)</td>
<td>7.8</td>
</tr>
<tr>
<td>Euro Zone Barometer (March 2019)</td>
<td>7.7</td>
</tr>
</tbody>
</table>

*Table 2: Results of the SPF in Comparison with other Expectations and Projections (source: ECB SPF Q2 2019)*
Introducing Machine Learning as a novice forecasting method, it differs on a lot of the above-mentioned properties. First off, Machine Learning is not subject to the discretion of forecasters. Professionals do not need to use their economic intuition and judgement to evaluate a model. Instead, the model is data-driven (cf. supra). One of the only choices the economist needs to make is how he will constrain the model’s complexity. Sources of this complexity are the number of variables, the number of parameters a model “learns”, the number of parameters the user defines and the number of variable relationships a model captures internally.

Directly related to a ML model’s complexity are the sources of errors. A forecasting error can be decomposed in a bias and variance (Hall, 2018). Bias refers to errors due to inaccurate assumptions about probability or data. It is the difference between the average prediction of our model and the correct value which we are trying to predict (Singh, 2018). Variance, in contrast, refers to errors due to a model’s sensitivity to small disruptions in the underlying data. Figure 7 clarifies this difference. The orange dot represents the true output a model is trying to capture, while the blue dots represent the learned outputs of a model under different conditions. Four combinations are illustrated: low bias and variance, low bias and high variance, high bias and low variance, and high bias and variance. Bias indicates how close to the true output (orange dot) the model’s predictions (blue dots) are. Variance indicates how tightly or loosely the predictions are clustered. Models with high bias and low variance yield observations that are tightly clustered but far from the correct prediction. Models with low bias and high variance, on the other hand, yield observations that are only loosely clustered around the correct prediction.

These two kinds of forecasting errors form a trade-off and being able to perfectly balance them guides the user to an optimal forecasting model. Very complex models produce a low bias, but a high variance in the data. This is shown in figure 8. You can see as the complexity of a model rises, the bias is being reduced, but the variance is increasing. It is the dashed line that indicates the minimum total error, and thus the optimal model’s complexity. Meaning, in turn, that the corresponding forecast error for new observations is minimized.
Figure 7: Errors of a Model due to Bias and Variance (source: Hall (2018))

Figure 8: Trade-Off between Errors due to Bias and Errors due to Variance (source: Hall (2018))
Actually, dealing with bias and variance is really about dealing with under- and overfitting (Fortmann-Roe, 2012). The more parameters you add, the more variance becomes a primary concern. At the same time, bias is steadily falling. In comes the danger of overfitting: adding too many parameters in your model equals an increase in error due to variance. Your model does not only capture the underlying pattern in the data, but also its noise. We are right of the dashed line. The opposite, having too few parameters, is called underfitting. The model is underperforming, it is not able to capture the underlying pattern of the data. This often happens when the amount of data is too little, or when we try to build a linear model with a nonlinear data (Singh, 2018). Figure 9 depicts the problem of over- and underfitting.

![Figure 9: Overfitting and Underfitting (source: Singh, 2018)](image)

We can illustrate this by looking at an example of forecasting the unemployment rate using a linear regression model estimated using OLS. We could forecast the unemployment rate using only one explanatory variable: the previous period's observed unemployment rate. It’s obvious to say this model will have huge forecasting errors. The future unemployment rate doesn't only depend on the past unemployment rate, but on a whole bunch of explanatory variables. This model is said to have a large forecasting error from bias. However, the forecasting error from variance will be rather small: the model is robust to small perturbations in the underlying data (i.e., small changes in the observed variable translate to small changes in the forecast). To lower this forecasting error, explanatory variables can be added. The forecasting error from bias will decline doing so, but the forecasting error from variance will increase. If we add too many variables, the explanatory strength will only apply to the given sample. The model will have a poor generalization power. In other words, small perturbations in lots of variables will now translate to disproportionally large changes in the forecast (Hall, 2018). To conclude we could state that a more complex model doesn’t necessarily mean a better performing model.
Are there methods to balance bias and variance? How can we find the intersection between bias and variance where total error is the smallest? In Machine Learning we obtain this by ‘disciplining’ a model. The act of ‘disciplining’ is called Regularization. Here we introduce certain penalties for overfitting the data, we discourage complexity. The problem with overfitting is that your model puts too much time in separating noise from data points that really represent the true properties of your data (Gupta, 2017). Models sometimes get too complex. In this case they perfectly fit what has been observed, but they have a hard time generalizing to data outside of the training sample (due to accidents of the sample) (Owen, 2016). They are said to have poor generalization power. To solve this, we have to shrink certain (non-important) coefficient estimates to zero. We need to artificially discourage these too extreme models. In this way we significantly reduce a model’s variance, without causing an increase in bias.

We know that in standard statistics (take an OLS regression for this example), but also statistical learning, we always try to minimize a certain function. We call this function the loss or cost function. This function represents a certain modification of the difference between the variable to be predicted and the predicted value. Mostly used is the Residual Sum of Squares (RSS). Minimizing this function and thus making sure your estimated model fits the data well is called the fitting procedure. We modify the coefficients in such a way that this cost function is minimized, based on the training data. Figure 10 shows the equation for the RSS function.

$$\text{RSS}(\beta) = \sum_{i=1}^{N} (y_i - f(x_i))^2$$

$$= \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j)^2.$$  

*Figure 10: Residual Sum of Squares (Cost Function for OLS Regression)*

Looking at Machine Learning, the process of Regularization is implemented by modifying the RSS. For the scope of this paper, we focus on the three most prominent estimators for regression-related problems: Ridge Regression, Lasso, and Elastic Net.
Looking back, we know that Regularization implies shrinking certain coefficients estimated towards zero. So, we will need a *shrinkage quantity*. In the case of Ridge Regression, this involves the alteration showed below *(figure 11)*.

\[
\sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2
\]

*Figure 11: Ridge Regression Estimator*

Again, the coefficients are estimated by minimizing this function. However, now the flexibility of our model is penalized (Pedregosa et al., 2011). This is done by the *tuning (or complexity)* parameter \((\lambda)\), which decides how much we will actually discipline the model. The larger this parameter, the greater the shrinkage of the coefficients (coefficient estimated using Ridge Regression are also known as the L2 Norm). This decrease in value of coefficients automatically means a decrease in variance, and thus avoiding overfitting. Nonetheless, the attentive reader must’ve already noticed that an unnecessarily high lambda\(^1\) (tuning parameter) implies an increase in bias. Your model will start to lose important properties. It is then said to be underfitting.

Choosing the right weight of your lambda is of the utmost importance. But, how is this done? How will we balance the expressiveness of a model against the risk of overfitting? Setting the Regularization parameter is done by *cross-validation*.\(^2\) Here you divide your training data and set different values of parameters for each sub-sample. Subsequently, you get different model realisations in each sub-sample and are able to compare their respective test performances. We want to minimize the average of the mean-squared error of the prediction on the sub-samples. You train your model while varying the parameters. Most popular forms of cross-validation are *(Stratified) K-Fold Cross Validation* and *Leave-P-Out Cross Validation*.

---

\(^1\) If lambda is 1, all the coefficients will be zero.

\(^2\) There exist computational cheaper alternatives, such as *Information Criteria* (AIC and BIC).
A second estimator derived here is LASSO (Least Absolute Shrinkage and Selection Operator) (Friedman, Hastie, & Tibshirani, 2010). Here, instead of using squares of beta as a penalty, the modulus of beta is used (L1 Norm). Reason for using the modulus is to only penalize the high coefficients. In this way fewer parameters values than (for example) with the Ridge Regression estimator are selected, many of their estimated coefficients are zero. These models are said to have sparse solutions. See figure 12 for the equation.

\[
\sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| = \text{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|.
\]

Figure 12: LASSO Estimator

The big difference between Ridge Regression and LASSO is that using Ridge Regression the insignificant coefficients are not enforced to be zero (Oleszak, 2018). You won’t be able to get entirely rid of the irrelevant features in your model, they will only be minimized (to almost zero). The LASSO method overcomes this disadvantage. It actually sets the coefficients to zero if they appear to be irrelevant (especially for high values of lambda).

Finally, we take a look at the Elastic Net method, which was actually created as a critique to the LASSO estimator (Oleszak, 2018). LASSO was, or is, said to be too dependent on data for variable selection, making it an unstable estimator. As a solution, in Elastic Net methods, we linearly combine the L1 and L2 penalties of Ridge Regression and LASSO. As the figure below shows (figure 13), the L1 and L2 penalty are combined with the use of alpha, a mixing parameter between ridge (\(\alpha = 0\)) and lasso (\(\alpha = 1\)). Be aware that we now need to tune two parameters, lambda and alpha.

\[
\frac{1}{2n} \sum_{i=1}^{n} (y_i - \hat{x}_i \hat{\beta})^2 + \lambda \left( \frac{1 - \alpha}{2} \sum_{j=1}^{m} \hat{\beta}_j^2 + \alpha \sum_{j=1}^{m} |\hat{\beta}_j| \right)
\]

Figure 13: Elastic Net Estimator

Allow me to summarize. Why was this Regularization process necessary again? OLS models always tend to have some variance in it. It could be that it performs perfectly for the given
training data, but when we take it to new, unseen data this performance drastically declines. In that case, it is said to have a poor generalization power. In comes Regularization. This technique will significantly reduce this variance, without an out-of-order increase in bias. It is the tuning parameter \( \lambda \) that is responsible for this task. The bigger the value of lambda, the smaller the value of the coefficients, and thus the smaller the variance. Of course, we cannot make this value of lambda too high, for that it will give rise to the bias of your model. The model will start to lose important properties. This trade-off between under- and overfitting is of the utmost importance and this is why your tuning parameter should always be very carefully selected. This is done with cross-validation.

In this subsection, we focused on the three most prominent ways to forecast: statistical forecasting, consensus forecasting and Machine Learning. We elaborated on finding the right estimators to predict quantities (regression) using Machine Learning. Which estimators to use when we want to predict a category (clustering and classification) is outside the scope of this paper. For an extensive overview of all estimators, I refer to Scikit-learn (cf. infra).

**Machine Learning and Causal Inference**

Some economists are willing to ignore the huge impact of Machine Learning techniques because of their lack of emphasis on identification of causal effects. And it is true, ML has for a long time been strictly interested in predictive modelling and forecasting. However, this strict distinction has been fading over the last years. The work of, for example, Athey (2018), Athey and Imbens (2015, 2017), Athey and Wager (2015), and Belloni et al. (2015) has contributed to this fading and to the development of inference techniques for Machine Learning. In this subsection, I’ll give you a brief overview of how Machine Learning can be used for causal inference.

Why isn’t causal inference for Machine Learning as evident as we think? Machine Learning, nowadays, is used for “simple”\(^{13}\) problems, such as prediction. It is easy to evaluate these models. We simply look at the goodness-of-fit in the hold-out test set. For causal inference, in contrast,
this isn’t as easy. There does not exist an unbiased estimate of the empirical evidence\textsuperscript{14} available for comparison (Athey, 2018). Or in other words, it is impossible to observe multiple counterfactual worlds at the same time (Athey, 2015). We do not, as an example, see the same consumer exposed to two different prices at the same time. This means that much more complicated algorithms will be needed to assess economic problems. However, in the recent years, causal inference is making its inroads in Machine Learning techniques.

As said in the previous paragraph, ML methods (prediction) are executed in order to optimize the goodness-of-fit. You want your model to perform\textsuperscript{15} as well as possible when applying it to the test data. In opposition to prediction, when estimating causal effects, the goodness-of-fit of a model is often neglected. Researchers often tend to sacrifice predictive accuracy in the current environment to learn about a more fundamental relationship that will help make decisions (Athey, 2018).

Luckily -since it is in fact the fundamental relationship we are most interested in-, this strict distinction is fading over the recent years. Researchers are starting to use the strengths and innovations of ML methods to apply them to causal inference. Unfortunately, doing so doesn’t seem to be obvious since it requires changing the objective function. Yes, the ground truth of the causal parameter is not observed in any test set (cf. supra). This directly implies that statistical theory will play a more important role to determine whether a certain model performs well. This new objective function is about identifying causal relationships. It’s the identification strategies such as instrumental variables, difference-in-difference designs, and regression discontinuity designs\textsuperscript{16} that have been gaining a lot of attention in this new ML/causal inference literature. However, for an elaborate explanation on how this works (and the problems that it entails), I refer to the literature.

A next question we could ask ourselves is, why we actually need causality in Machine Learning? Why don’t we just keep the strict distinction? Econometrics is for causal inference, ML methods are for predictions. Ramnath Chellappa (2017) says that econometrics should keep providing the

\textsuperscript{14} This is called the Ground Truth in Machine Learning, and refers to the accuracy of the training set.
\textsuperscript{15} Good performance meaning here that you want it to predict outcomes as well as possible.
\textsuperscript{16} And plenty more, see Athey (2018) for a thorough summary.
theoretical foundations, but ML could be used as technical assistance. Yes, econometrics and ML can be beneficial to each other. We could, as an example, use ML models to feed in the components of econometric models (cf. supra). However, as stated before, researchers such as Susan Athey and Guido Imbens have narrowed this gap between econometrics and ML. They study and introduce ML techniques that can be used to isolate causal effects. What the future will bring for these developments is still unsure, but causality is and will always be economics’ most fundamental question. Moreover, the most fundamental economic questions, such as how consumer demand varies with price, cannot be answered with purely predictive models (Athey, 2017).

We can conclude that one of the biggest drawbacks of Machine Learning algorithms is that given the lack of a pre-defined model for analysis, Machine Learning algorithms hold limited explanatory power and cannot, as a result, readily explain what factors drive the forecasts (Jung, Patnam, & Ter-Martirosyan, 2018). However, ever-increasing literature is taking care of this disadvantage and introducing causal inference for Machine Learning problems.

An elaborate introduction to Machine Learning was given in the previous sections. Now, for the remainder of this literature study, we get to the essence of this dissertation: the use of ML to forecast macroeconomic indicators. But, first an introduction to macroeconomic forecasting is needed.

**Macroeconomic Forecasting**

It’s generally known that most forecasters fail to predict recessions in advance (Fildes & Stekler, 2002). Even though forecasting accuracy was higher than ever preceding to the Great Moderation (which was more due to a less volatile economic environment than improved techniques), no one seemed to see it coming. Rarely do macroeconomic forecasts produced with macroeconomic models perform better than simply intelligent guesswork (Wren-Lewis, 2014). Still, macroeconomic forecasting has been and always will be important to the entire profession. In this section we will try to tell you why this is. Further, we will answer the following questions:
About what should an economic forecast provide information? How do we evaluate these forecasts? What have been the most prominent forecasting techniques in the last 70 years? …

Why, despite predictions repeatedly clashing with past errors, do we continue to strive for constantly improving forecasting models? One reason is that we feel uncertainty like a pain. Belief in experts’ forecasts can keep persisting, even if the forecasts being offered are “manifestly useless” (Goodwin, 2017). It’s a means of keeping the people calm. Even with the certainty of these future developments being uncertain, people want to believe that somebody really knows. Next to this need for certainty, there is also the experts’ overconfidence (Rozenblit & Keil, 2002). We think we know how things work. Stock markets, economies, financial markets, etc., there is always somebody who believes he really understands. This is why forecasters keep forecasting. Further, policy making bodies, such as the central bank and governments, play an important role. These models often give a good understanding of the whole story. They combine a large share of information and link what has happened to what might happen. To summarize these reasons, a quote by President Obama (2012) near the end of his first term:

"When I think about what we've done well and what we haven't done well, the mistake of my first term -- couple of years -- was thinking that this job was just about getting the policy right. And that's important. But the nature of this office is also to tell a story to the American people that gives them a sense of unity and purpose and optimism, especially during tough times."

There are more reasons, but the ones stated here seem to keep returning. Driven more by psychology and sociology than economic performance itself.

Say we want to predict the level of unemployment for the next five years. Of course, the main aim of this prediction will be to determine the economy’s direction of movement. Will the level of unemployment be higher or lower in five years than it is today? On the other hand, this economic forecast should also provide information about (1) the timing of any critical moment, (2) the vastness of the change, and (3) the length of time over which a movement is expected to...

---

17 From an interview with CBS News (Boerma, 2012).
persever. How these criteria were quantified in the last decades, we will discuss in the coming subsection.

Before looking at the history of macroeconomic forecasting, an understanding of how these forecasts are evaluated is needed. In addition to the prediction itself, it is important to know how accurate it is. This accuracy is represented by the prediction (or forecasting) error \( f_e \). It is the difference between the true s-step-ahead value of the variable in question and its s-step-ahead forecast.

\[
f_{t,s} = y_{t+s} - f_{t,s}
\]

Using this forecasting accuracy, we are able to evaluate different models. To do this the out-of-sample\(^{18}\) forecasting accuracy is quantified in a holdout sample running from \( T_1 \) to \( T^{19} \). Mean Error (ME), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Percentage Error (MPE), and Mean Absolute Percentage Error (MAPE) are distinguished, each with its own strengths and weaknesses. Simply put, the closer our forecasting error to zero, the better. The equation for the Mean Error is shown below, which is used most often. Taking the mean of your error makes the positive and negative errors cancel out. This equation is the perfect indication for over or under estimation.

\[
ME = \frac{1}{h} \sum_{s=1}^{h} f_{T1,s}
\]

It will also be these equations that we will use to measure the performance of our ML model in the research part of this dissertation, but more on that later. Let’s first have a look at the past and present of macroeconomic forecasting. This short history will, however, be very brief\(^{20}\) since the emphasis of this dissertation is on the future of macroeconomic forecasting. Yet, it cannot be denied a clear understanding of its past is necessary to see where we are going to. Note that

---

\(^{18}\) Out-of-sample forecasts are those generated for a set of data that was not used to estimate the model, i.e. do not use all observations in estimating the model and evaluate the model from the forecasting accuracy in the holdout sample

\(^{19}\) Meaning with length \( h = T - (T1 - 1) \)

\(^{20}\) For those who crave more, I refer to Fildes and Steckler (2002), and Diebold (1997)
the following short history will be almost entirely based on Diebold (1997) and mainly summarize his work.

A Short History

Macroeconomic forecasting goes back to the 30s and 40s of the previous century. With John M. Keynes being the most prominent economist of his time, the Keynesian Theory was the first major wave of twentieth century macroeconomic theory (Diebold, 1997). With this theory – The General Theory - dominating the economic landscape, economists (such as Klein (1946) and Klein & Goldberger (1955)) desperately tried to construct, estimate and analyze these models. Thus, structural econometric forecasting came to life in the late 1950s and 1960s. What structural (and nonstructural) models are, will be discussed in this subsection.

Predicting future macroeconomic indicators such as inflation, GDP/capita, unemployment, etc. has always been a subject of great interest. Whilst some say predicting the future is impossible, constantly improving forecasting accuracy is a fact\(^{21}\) (Diebold, 1997). Structural and nonstructural macroeconomic forecasting have been alternating over the last fifty years. Structural models use economic theory to interpret economic data (e.g. DSGE models), which in turn means its popularity too rises and falls with economic theory\(^{22}\). Nonstructural models, on the other hand, are largely based on reduced-form correlations (e.g. ARMA models). Little reliance on economic theory is needed here. Models we will elaborate on in this paper will fit in with the nonstructural models. As ML algorithms search for patterns in the data, they do not rely on pre-determined economic models.

The rise of Keynesian theory (cf. supra) and its implied rise of structural forecasting came to an end in the 1970s and 1980s. It were the cracks in the foundation that triggered this fall. The end of structural forecasting can be contributed to three intellectual dissatisfactions (Diebold, 1997):

---

\(^{21}\) Interesting in the continuation of this paper is the difference between prediction and forecasting. As the definition of ML states, one of the main areas is prediction. Oddly enough, an agreed upon distinction isn’t present in today’s literature. However, agreed is that forecasting would be a subset of prediction. Any time we predict into the future, it is a forecast. This makes prediction the more general statement. The major difference is that forecasting can have error, that is why we have confidence interval of 5%, 2.5%, 1% and residuals for modeling.

\(^{22}\) Often with a lag. See the rise of Keynesian theory (1930s and 1940s) and the accompanied rise of Structural Keynesian macroeconomic forecasting (1950s and 1960s)
(i) Economists became dissatisfied with the lack of foundations for the disequilibrium nature of the Keynesian model; (ii) macroeconomists became disenchanted with ad hoc treatment of expectations, the ‘rational expectations revolution’ quickly took hold; (iii) macroeconomists became dissatisfied with the overall modeling approach\textsuperscript{23} embodied in the programme. Apart from these intellectual dissatisfactions, the economic facts of the 1970s totally erased belief in the Keynesian theory; high inflation and unemployment.

This slump in structural forecasting had to be picked up by something new: nonstructural forecasting. Remember, nonstructural models are largely based on reduced-form correlations. The environment is \textit{unconditional}, meaning that interest often centers on the likely future path of the economy when policy remains unchanged. And it so happens to be that unconditional models do not require a structural model. The revival\textsuperscript{24} of nonstructural forecasting was a fact.

Most popular techniques of nonstructural forecasting are Vector Autoregressive (VAR) type models and consensus forecasts (Cook & Hall, 2017). Unfortunately, both these techniques suffer great limitations, such as linearity and specification sensitivity (cf. supra). However, Neural Networks and Machine Learning provide great solution to these limitations and present a new forecasting approach (cf. infra).

This unconditional environment can be interesting, yet it also often happens we want to analyze scenarios that differ from the conditions presently prevailing, such as the effects of a change in a policy rule or a tax rate. Such conditional forecasts require structural models. After its decline in the 1960s and 1970s, a new structural approach came to life in the 1970s and 1980s: \textit{Dynamic Stochastic General Equilibrium (DSGE)} modeling.

Francis Diebold (1998, p. 189) ends his paper with the following words:

\begin{quote}
“The hallmark of macroeconomic forecasting over the next 20 years will be a marriage of the best of the nonstructural and structural approaches, facilitated by advances in
\end{quote}

\textsuperscript{23} The ‘system-of-equations’ approach

\textsuperscript{24} I say revival because the intellectual development of nonstructural forecasting predates the Keynesian period.
numerical and simulation techniques that will help macroeconomists to solve, estimate, simulate, and yes, forecast with rich models.”

And now, 20 years later, he might just have been right...

As said, I only gave a brief history in this subsection. To get to know the properties, strengths, and weaknesses of the models described above, I strongly recommend reading Diebold’s work on *The Past, Present, and Future of Macroeconomic Forecasting* (1997). Let’s now have a look at the future of which Diebold was writing about.

**The Use of ML to Forecast Macroeconomic Indicators**

How are we able to forecast macroeconomic indicators with the use of ML? We now know that ML is a data-processing methodology that develops algorithms to detect patterns and learn how to make predictions. In this subsection we will look at how you can use these ML techniques for an improved forecasting accuracy of macroeconomic variables. Most frequent macroeconomic indicators are GDP (or GDP per capita), inflation and unemployment. Furthermore, we have a look at technically more advanced methods, such as Deep Neural Networks. We, however, admit that the use of Deep Neural Networks will lead us too far and only give a short introduction.

**Using the right estimator**

Let us resume. When introducing Machine Learning, we distinguished two branches of *Learning*: supervised and unsupervised Learning. We call this learning because we want the developed algorithms to improve their own performance at solving certain problems after receiving additional information about the problem. The first type is used for classification and regression problems. The second one mainly for clustering problems. It goes without saying that for our problem – forecasting macroeconomic indicators – we will mostly be predicting quantities. We want to know how GDP, inflation and unemployment will look like for different time horizons. This in turn means, focus will mostly go to regression problems. The only time we might need to
handle clustering problems is when our data are unlabeled. We then first need to organize our data in different categories.

For these Machine Learning problems, we always need to find the right estimator. This is often the hardest part in solving the problem, as different estimators are better suited for different types of data and different problems (Scikit-learn, 2011). On the flowchart presented in figure 14, a rough guide on how to approach these problems is presented. First and foremost, we should ask ourselves whether we have enough data. Since Machine Learning is a data-processing methodology, at the very least fifty samples are necessary. The next question is whether we will be predicting a quantity or a category. If the answer is a quantity, we find ourselves with the regression problems. The four most recurrent estimators here are the SGD Regressor, LASSO, ElasticNet, and RidgeRegression. The SGD Regressor is well suited for regression problems with a large number of training samples (< 10.000). For other regression problems, the other three estimators are more recommended (cf. supra). Other alternative estimators are SVR (Support Vector Regression) and EnsembleRegressors, but these won’t be discussed here.

If we are not predicting a quantity, but a category instead, we are led to classification or clustering problems. The difference between classification and clustering is that when looking at classification problems, you should already have a set of predefined classes. You want to know which class a new object belongs to. Clustering problems, on the other hand, work with unlabeled data. Objects that are in some relationship with each other are being clustered. So, the logical question here is whether we are dealing with labeled data or not. If we are in fact dealing with labeled data, we are carrying out a classification problem. Again, the same question should be asked: are we working with a large number of training samples or not? If this is the case, the SGD Classifier estimator is advised. For less comprehensive training samples (< 10.000), we generally work with a Linear SVC (Support Vector Classification).

---

26 Stochastic Gradient Descent, see appendix for an explanation on this learning routine
27 Note that the number in the flowchart below is wrong!
28 You can distinguish linear ML (ridge regression, LASSO, elastic net, ...), nonlinear ML (classification and regression trees, support vector regression, ...), and ensemble ML (bagging, boosting, random forests, ...). We will only focus on linear ML in this dissertation.
The last ‘problem’ discussed here are clustering problems. This occurs when your data are not labeled yet. You just have a bunch of unconfined data which still need to be categorized. When you already know the number of categories the *K Means* or *Mini Batch K Means* estimator are the most appropriate estimators, dependent on whether you are working with large datasets (Mini Batch) or not. When the number of categories is unknown, *MeanShift* or *VBGMM (Variational Bayesian Gaussian Mixture)* estimators are best fitted. For very large datasets, there aren’t any estimators able to handle this kind of problem. Dimensionality reduction won’t be discussed here.

**Reviewing the existing literature on macroeconomic forecasting with ML**

As follows, we can have a look at already existing literature on macroeconomic forecasting with ML. If this above-mentioned information is correct, most of these papers should be using the *SGD Regressor, LASSO, Elastic Net, or Ridge Regression* to predict different macroeconomic indicators.
A first paper to analyze is one by Aaron Smalter Hall (2018). He uses the Elastic Net model to forecast the U.S. civilian unemployment rate at a monthly rate. Comparing this with a simple statistical forecasting model and consensus forecasting, he concludes that a more complex model—being an Elastic Net model—outperforms these, given enough data to learn from. He further admits that the key to this result is Regularization, by which he controls for model complexity. It should, in my humble opinion, be noted that this ‘simple’ statistical model is in fact very simple and should not represent the accuracy of statistical models as a whole. An AR model is used. By definition, AR models only use past observations of the unemployment rate to forecast future outcomes.\(^\text{29}\) A lag length of six months is used here, based on the BIC (Bayesian Information Criterion).

Secondly, Garcia, Medeiros, and Vasconcelos (2017) use high-dimensional and Machine Learning models to forecast inflation in real-time for an emerging economy. More specifically, techniques based on LASSO appear to have the smallest forecasting errors for short horizon forecasts. Forecasts were estimated for horizons of five days before the CPI index is published to 11 months plus five days (a total of 12 forecasts). For the second horizon, the adaptive LASSO\(^\text{30}\) is superior to any other model.

Medeiros et al. (2019) also try to forecast inflation using Machine Learning methods. Instead of trying so for emerging economies, they perform a forecast for the U.S. inflation. Contrary to the likes of Stock and Watson (1999, 2007, 2008, 2016) who say that it is impossible to beat benchmark models (e.g. AR model) with Machine Learning methods, Medeiros and his colleagues say the opposite. They performed a monthly U.S. inflation forecast with 16 different ML methods (LASSO family, Bayesian VAR, Bagging, Boosting, CSR, and Random Forests)\(^\text{31}\), of which the Random Forests and shrinkage methods appeared to be the best performing models.

A last paper we will review is one executed by the International Monetary Fund (Jung, Patnam, & Ter-Martirosyan, 2018). They present three different Machine Learning algorithms to compute

\(^{29}\) In contrast, the Elastic Net model here uses 138 macroeconomic variables to forecast the future unemployment rate.

\(^{30}\) Adaptive LASSO serves as a solution for LASSO's deficiencies (such as not serving the oracle property). See Zou (2006) for an explanation.

\(^{31}\) For Bayesian VAR, Bagging, Boosting, and CSR I refer to the literature, for Random Forests see the appendix.
short-term GDP growth forecasts for seven countries\(^\text{32}\) across geographies and levels of economic development. One of these algorithms is the Elastic Net. They beautifully summarize the working of the Elastic Net algorithm as follows (2018, p. 6):

“The Elastic Net algorithm provides a regression-based approach to forecasting that has the advantage of classic OLS, in terms of interpreting the contribution of each variable, but additionally offers a flexible way to conduct variable selection and prevent forecast overfitting”.

They also conclude that the Elastic Net model was able to consistently outperform a benchmark forecast performance by the IMF’s World Economic Outlook forecasts.

Even though still in its starting blocks, it becomes clear researchers all over the world are committing to Machine Learning algorithms as a novice forecasting method. Given the right requirements, such as enough data, Machine Learning algorithms almost always perform better than traditional forecasting methods.

The Use of DL to Forecast Macroeconomic Indicators

In the first section on Machine Learning, we briefly introduced Deep Learning. Now, in our search for papers using ML techniques to forecast macroeconomic indicators, we also encountered papers applying DL instead of ML, taking it to an even further level. For example, Thomas R. Cook and Aaron Smalter Hall (2017) worked with Deep Neural Networks to forecast certain macroeconomic indicators (such as the unemployment rate). In this subsection , We will reintroduce Deep Learning and guide you through a deeper understanding of the concept. Afterwards, we’ll review some papers using these techniques to forecast macroeconomic variables. Notice that we will not be working with Artificial Neural Networks in later applications since these are likely to pose technical challenges as they grow in size (Giesecke, Sirignano, & Sadhwani, 2016).

\(^{32}\) Germany, Mexico, Philippines, Spain, United Kingdom, United States, and Vietnam
Let’s look back. What is Deep Learning and how does it differ from Machine Learning? Deep Learning is a type of Machine Learning. Yet, we call it ‘deep’ because it is able to process a wider range of data resources, requires less data preprocessing by humans, and can often produce more accurate results than traditional Machine Learning approaches (Chui, Kamalnath, & McCarthy, 2018). When writing about Deep Learning, we think about Deep Artificial Neural Networks. A Neural Network with neurons interacting, like the biological system in our brains, is artificially imitated. These ‘neurons’ are software-based calculators and find themselves in interconnected layers, forming the Neural Network. In this manner, the capability to learn - like us humans - is imitated. A feature which normally isn’t adapted by computers (Kriesel, 2007). Using the words of Arthur Samuel (1959): Deep Learning is a “field of study that gives computers the ability to learn without being explicitly programmed”. In the following paragraphs the technicalities behind this will become more clear. Figure 15 gives you the similarities between biological neural networks (left) and artificial neural networks (right).

![Biological Neuron Graph (left) & Artificial Neural Network (right)](source: Ghallou, 2017)

It is important to notice that Neural Networks were developed in a graphical manner. Instead of choosing to represent the models as equations, preference went to directed graphs. Since we are dealing with highly complicated models, this helps to represent increasingly complex models. This is also why, when writing about Neural Networks, we do not refer to model specifications – such as in economic or forecasting models -, but instead refer to the architecture of a model. These directed graphs work as follows: you feed certain information as an input, and it flows to
the output through the structure of a graph. The different graph nodes in between serve as operations executed on the data as it moves from input to output (we call these sets of nodes between the input and output hidden layers). And it is the architecture of a model which decides the configuration of these nodes (and the interconnection between them) and which operations should be performed at each node (Cook & Hall, 2017).

Let’s visualize this in the easiest way possible: a set of nodes representing model inputs, a set of computational nodes, and a set of nodes representing model outputs. These sets are also called layers and this basic architecture is called the perceptron (with only one set of computational nodes). We can easily link this to forecasting. Imagine your model inputs being lagged values of a certain macroeconomic variable. Each lagged variable will be appointed a certain weight \( w \): how important is this value in the calculation of the future macroeconomic indicator of interest? These weights are being adjusted throughout the training process. The linear combination of inputs is in turn transformed through some function (the one hidden layer) to forecast the indicator at some prediction horizon. The eventual forecast is presented in the output set. This is visualized in figure 16.

![Figure 16: The Perceptron (source: Cook & Hall (2017))](image)

We will, however, focus more on multilayer perceptrons (MLP). The same method of operations applies here, but with more layers that learn increasingly complex features of the data. The network learns specific characteristics of the data and assigns certain weights to them. Training your network means learning whether these determinations are correct, and thus changing the weights in a specific manner. If correct the network is able to use what it has learnt from the data.

---

33 Nodes, neurons, or processing units.
34 ‘Deep’ is a technical term. It refers to the number of layers in a Neural Network.
to make determinations about new data. In other words\textsuperscript{35}, the network can learn to map patterns presented at the input layer to target values on the output layer (Önder, Bayir, & Hepşen, 2013). The performance of weight adaptation is also called \textit{learning or training} the algorithm\textsuperscript{36}.

Major Models

In what follows, we will briefly elaborate on the most essential Deep Learning models, or architectures: \textit{Convolutional Neural Networks (CNN)}, \textit{Recurrent Neural Networks (RNN)}, and \textit{the Encoder-Decoder Network}.

\textit{Convolutional Neural Networks (CNN)} are most popular when analyzing visual images (image recognition)\textsuperscript{37}. What is special about these Networks, opposed to other ANN’s, is that it has some type of specialization for being able to pick out or detect patterns and make sense of them. This seems not only an attractive virtue for image recognition, but also for macroeconomic forecasting. Moreover, these Networks are feed-forward. Data are allowed only to move in one direction, from input to output. A third property is that the neurons in each layer are not connected to every neuron in the previous layer. Instead, they are only connected to neurons close to it. An example is shown in figure 17.

But how does this differ from our standard MLP? A CNN has hidden layers called \textit{convolutional layers}, which perform convolutional operations on the input that it’s been served. To understand this, we need to know what a convolution is. A convolution is how the input is modified by a filter (Suhyun, 2015). These filters ‘scan’ your input and learn different characteristics of the input. A filter can, for example, slice through an image searching for certain shapes (circles, edges, ...). Each time a match is found, this is mapped out onto an output image. Basically, this means that the convolutional layers will filter for specific patterns in the data. Taking this to macroeconomic forecasting (and other predictions), the CNN will distinguish which patterns maximize predictive

\textsuperscript{35} We do know a lot of text is put in the repetition of certain concepts. However, we deem this very necessary since the upswing of related literature brings along a lot of different explanations for the same concepts. The novice reader will easily become confused when trying to get his head around the wonderful world of Deep Learning (and other learning mechanisms).

\textsuperscript{36} See appendix for a description on the most popular learning algorithms; gradient descent, and backpropagation

\textsuperscript{37} Explaining the technicalities behind image recognition is, however, outside the scope of this paper. For that I refer to the literature (…)}
accuracy and search for these patterns (Cui, Chen, & Chen, 2016; Nouri, 2014). After summarizing the different major models, we will see this applied in a case that predicts the unemployment rate.

![Figure 17: An Example of the CNN Architecture (source: Hidaka & Kurita (2017))](image)

**Recurrent Neural Networks (RNN)** are in turn most popular when dealing with time series data. They are designed to recognize patterns in sequences of data (such as texts, but also macroeconomic time series). This is why it seems obvious that these will perform best when forecasting macroeconomic indicators. Special about the Networks is that information can be stored in so-called **context nodes**. RNNs have memory! They learn data sequences, allowing the Network to output a number or another sequence. In other words, it ‘remembers’ what data it’s been fed (and in what sequence), enabling the RNN to accurately predict the output. This is done by assigning probabilities to all possible outputs based on the previously fed input.

The term ‘recurrent’ perfectly reflects the most important characteristic of RNNs. It refers to the way information is channeled through the. Opposed to **feedforward nets**\(^{38}\), which never let a certain node be ‘used’ twice or more, RNNs feed information through a loop. As implied in the previous paragraph, Recurrent Networks do not only use the current input, they also use what they have perceived previously in time. They use their memory.

\(^{38}\) Such as the Convolutional Neural Networks.
RNNs most prominent\textsuperscript{39} variation is the \textit{Long Short Term Memory (LSTM)} network. But, to develop a deeper understanding of these networks, we first need to elucidate a disadvantage of standard RNNs: the problem of \textit{long-term dependencies} (Olah, 2015). One of the great possible advantages of RNNs is that they can connect previous information to present tasks. However, this advantage depends on the gap between the relevant information and the place where this information is needed. If this gap is small, RNNs can easily use this past information. If not, RNNs become unable to learn to connect the information\textsuperscript{40}. These larger gaps, where RNNs depend on past information for present tasks, are called \textit{long-term dependencies}. And the larger these gaps, the less efficient the RNN becomes in learning to connect the information. If you want to know the fundamental reasons why this is the case, we strongly recommend reading literature by Hochreiter (1991) and Bengio et al. (1994).

Back to LSTM networks (which were by the way introduced by the same researchers that explored the problems of long-term dependencies, Hochreiter & Schmidhuber (1997)). Apparently, they do not have the problem of long-term dependencies. Or better, they are able to learn these long-term dependencies. Or even better, remembering information for long periods of time is their default behaviour! This is where the term ‘long’ comes from, LSTM networks store long-running memory about the sequence. ‘Short’, on the other hand, refers to the short-running memory of the most recent network outputs. The network is able to combine contextual features of the data together with information provided by only the most recent elements in a sequence.

Last but not least, we take a brief look at the \textit{Encoder-Decoder Network}. Being a member of a broader class of networks, the \textit{sequence to sequence models}, this architecture is an extension of the LSTM networks. Sequence to sequence models lie behind a lot of popular applications, such as Google Translate, Apple’s Siri, and online chatbots. Google coined this term for the first time in 2014, saying these models aim to map a fixed length input with a fixed length output where

\textsuperscript{39} Almost all exciting results based on recurrent neural networks are achieved with LSTM networks.

\textsuperscript{40} An example by Olah (2015): In the sentence “The clouds are in the...”, the RNN doesn’t need a lot of past information to know that the next word will probably be ‘sky’. The gap is said to be small. In the sentence “I speak fluent...”, on the other hand, the RNN will probably need some extra previous information (such as “I grew up in France”) to be able to predict that the next word will probably be ‘French’. The gap is said to be large.
the length of the input and output may differ (Sutskever, Vinyals, & Le, 2014).

Moreover, words in the output can be predicted considering both the context of the individual words in the input, and the context of the words that have already been predicted in the output (Cook & Hall, 2017).

Given the origin of Encoder-Decoder Networks (being sequence to sequence models, introduced by Google for its Translate service), let’s dive deeper into their specific working and architecture. Encoder-Decoder networks are made of two components. Both components (the encoder and the decoder) are LSTM architectures. The Encoder comes first. As said when discussed LSTM (and RNN) networks, its task is to consider both the input sequence as the response of the encoder to the preceding portions of the sequence. The Decoder in turn uses the output of the Encoder as well as its own responses to these encoder-produced sequences. In the words of Cook and Hall (2017, p. 15): “the decoder allows for the model to make predictions that fit with the context established during earlier predictions”. An example of the use of Encoder-Decoder Networks for online chatbots is showed below (figure 18).

![Encoder-Decoder Architecture](source: Britz (2016))

This was the general use of Encoder-Decoder (E-D) Networks, for example when applied in language models. Now, let’s have a look at the use of E-D Networks in forecasting models. Imagine we want to predict the unemployment rate, not for the next period, but for a long forecast horizon. Normal LSTM models usually predict the next element in the sequence. In other words, traditional LSTM models perform quite well when the prediction horizon is one. However,

---

41 An example by Kostadinov (2019): “What are you doing today?” contains 5 words. If this is the input, and you want to translate it to Chinese (the output) you get: “今天你在做什么？”. Which is 7 symbols long. Regular LSTM networks aren’t able to map sequences that differ in length. Sequence to sequence models provide a solution.
when the prediction horizon is two or more, it will skip any consideration of the next element in the sequence, directly focusing on predicting the second-to-next (or further) element in the sequence. Consequences are that the forecasting accuracy will be far less accurate when forced to skip some elements in the sequence. A solution is provided by adding a decoder module. Now the encoder module (which is essentially a LSTM architecture) will produce a one step ahead prediction. Next, the decoder module will iteratively extrapolate that one step prediction out to the desired prediction horizon (Cook & Hall, 2017). In other words, the decoder component assists with predicting long forecast horizons.

In what preceded, we discussed the most important Deep Learning architectures. Most of the attention went to how many times the particular (sets of) nodes (or hidden layers) are run through. The Convolutional Neural Networks (CNN) were feed forward networks: data only move in one direction through the network. Recurrent Neural Networks (RNN), in contrast, were said to have memory. They do not only use the direct input, but also previously fed input which has already been run through the hidden layer(s). We distinguished different RNNs. One variation, the LSTM models, and one extension of these LSTM models, the Encoder-Decoder Network. Purely based on the theory, we could state that RNNs will perform better at macroeconomic forecasting than CNNs, since they are specially developed for time series data. Let’s have look at the literature and real-life cases to find out!

Reviewing the existing literature on macroeconomic forecasting with DL

In what follows, I will try to use the existing literature to vouch for my theoretical findings. Four existing papers on macroeconomic forecasting with Artificial Neural Networks will be summarized.

Of course, I start off with Cook and Hall’s paper on *Macroeconomic Indicator Forecasting with Deep Neural Networks* (2017), which was the source of inspiration for my dissertation. Focus goes to predicting civilian unemployment using models based on four different neural network architectures: Fully Connected architecture, Convolutional Neural Network, LSTM architecture, and Encoder-Decoder Network. Each of these models outperforms benchmark models at short time horizons. Benchmark models here are SPF forecasts and a Directed Autoregressive Model
(DARM). The performance goals are for the models to provide forecasts with lower mean absolute error than the SPF forecasts and for models to exhibit lower variance in repeated runs than the variance across individual forecasts in the SPF. One model based on an Encoder-Decoder architecture outperforms benchmark models at every forecast horizon (up to four quarters). See graphical representations of these models in the appendix!

Galeschchuk (2016) looks at the performance of Neural Networks in exchange rate prediction. To do so, he uses a *three-layer perceptron* (feed-forward and fully connected) and trains this with the backpropagation algorithm. The results show that the short-term prediction method provides good accuracy of the prediction and can be used in practical systems to predict the exchange rate for one step ahead. Even though this is not mentioned in the paper, we could state that looking at our theoretical findings- this model won’t be able to perform well for a long forecast horizon (cf. supra). In this case, an Encoder-Decoder model would be more advised.

Jung, Patnam, and Ter-Martirosyan (2018) did not only perform macroeconomic forecasting with an Elastic Net model (cf. infra), they also used a Recurrent Neural Network (RNN) to compute short-term GDP growth forecasts. Even better, to account for the additional cross-sectional interdependence of various economic variables, they employ a multivariate extension to the traditional RNN. Something that has not been done before in the field of economic forecasting. Again, more accurate results were obtained when compared to benchmark models, which were in this case the IMF’s own *World Economic Outlook (WEO)* models.

Finally, Chuku, Oduor, and Simpasa (2017) employ an Artificial Neural Network (ANN) to forecast economic time series in African countries and compare this to the performance of the more standard Box-Jenkins (e.g. ARMA models) and structural econometric modelling approaches. Quarterly data ranging from 1970 to 2016 is used. For the forecasting of GDP growth in selected frontier economies, ANN models performed slightly better than these structural econometric and ARIMA models. They conclude that these results are somewhat consistent with the results for developed economies obtained in, for examples, Tkacz (2001), Qi (2001), and Heravi et al. (2004).
The literature shows that novel methods based on Deep Learning provide highly accurate forecasting results. In a next section, we make the transition from literature study to research. Having already admitted that Deep Learning models would be too complex for this dissertation, we solely base our research on Machine Learning methods.
Application: Forecasting European Unemployment with ML

Time to put the money where the mouth is. In what preceded, we formulated the theoretical foundations for what promises to be a ground-breaking technology in our search for highly accurate predictive performance. Now, in this last section, we would like to take the theory to practice.

Together with the expected rates of inflation and the real GDP growth, the unemployment rate forms one of the key statistics that indicate the health of the euro area. Forecasting these indicators helps linking what has happened to what might happen. This is important not only for policy makers, who tend to rely on these forecasts, but also for the people in general making important choices based upon these predictions.

In what follows, we will use a promising Machine Learning estimator to perform a regression in order to forecast European unemployment: the Elastic Net model. Forecasting accuracy will be measured by the Mean Absolute Error (MAE). We will compare the predictive performance of our ML model to a consensus forecast (SPF since we are working with European data) and a benchmark model (an AR model). Before doing so, we will give you a short overview of employment in Europe since the second half of the previous century. We will look at the past and form (purely theoretical) expectations about the future. Afterwards, we will guide you through the technicalities behind our ML algorithm and discuss the chosen data with you.

Employment in Europe: A Short Overview

European unemployment has known a very divergent past. Before the Financial Crisis of 2007-2008, European unemployment was characterized by double digits. Especially compared to our North-American counterpart, unemployment rates were terribly high, with a rigid and inflexible job market at the base (Nickel, 1997; Blanchard, 2004). However, until the 60s of the last century, Europe was referred to as the “unemployment miracle”. For all that, in this subsection focus will be on European unemployment for the period of 2000-2019, since interest goes to the not so distant past and especially the future.
Let’s start with two remarks. First off, it is important to note that it matters ‘which’ Europe we will be evaluating. The euro area or Eurozone, and the European Union are both different according to included countries and thus also unemployment rate. Our research is based on the euro area (cf. infra), and when stating ‘European unemployment’ we will always be referring to euro area unemployment, unless otherwise stated. Secondly, these statistics do not say anything about the individual member states. Overall, we are experiencing a continuing progress (cf. infra), yet Greece (for example) is posting nearly ten times the rate of unemployment as the Czech Republic (Eurostat, 2019).

*Figure 19* shows the number of unemployed persons from January 2000 to June 2019, for both the EA-19 and EU-28 (Eurostat, 2019). Let’s first notice that the rate has reached an all-time low at the end of 2018. This is the result of a permanent decline since 2012, due to the end of the financial and economic crisis.

![Unemployment Chart](source: Eurostat (2019))

We know that unemployment behaves anticyclical when looking at the cyclicality of the various components of real GDP. This means that when the economy is doing well, unemployment will decline and vice versa. Accordingly, we can conclude unemployment in the euro area knew a vast increase, twice. Once during the financial crisis, once four years later during the European debt crisis. Since then, unemployment has known a serious recovery.
What about the unemployment rate expectations? For this, we take a look at the results of the ECB Survey of Professional Forecasters (SPF) on table 2 (cf. supra). We can see that for the coming years, a further decline is expected. It’s noted that labour markets appear to remain resilient, despite the prospect of slowing growth (ECB SPF, 2019).

After this brief look at the state of affairs regarding unemployment in the EU, we can proceed to the essence of this section: forecasting European unemployment using ML methods.

The Data

The essence of this dissertation is on creating bridges between economics and data science. In order to uncover hidden patterns in data -often patterns for which we don’t have a theory yet- a vast amount of data is needed. Of course, this elaborate dataset should also meet certain requirements. Here, we will discuss which dataset was used, its main properties, how it was prepared for our Machine Learning problem, and what some of the shortcomings are.

Finding the right dataset was no easy task. As our American colleagues often rely on the FRED-MD for big data macroeconomic research, a European counterpart does not exist to our knowledge. FRED-MD is a large, monthly frequency, macroeconomic database with the goal of establishing a convenient starting point for empirical analysis that requires big data. It consists of 134 monthly macroeconomic variables, starting from 1959:01, based on the FRED database. For our European assignment, we will rely on the Area-wide Model (AWM) Database instead.

The AWM Database represents a quarterly estimated structural macroeconomic model for the euro area (Fagan, Henry, & Mestre, 2005). The dataset covers a wide range of quarterly Euro Area macroeconomic time series and has become a standard reference for empirical studies on the Euro Area economy. The updated AWM database starts in 1970Q1 (for most variables) and is now available until 2017Q4. For an overview of all captured variables, I refer to the appendix.

---

42 See McCracken and Ng (2015) for a description.
43 Here representing 19 fixed countries: Austria, Belgium, Cyprus, Estonia, Finland, France, Germany, Greece, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Portugal, Slovakia, Slovenia, and Spain.
Immediately, two disadvantages compared to the FRED-MD become apparent. Both the number of included variables and the number of observations is smaller. FRED-MD consists of 134 macroeconomic variables, compared to the 46 macroeconomic indicators of the AWM Database. Moreover, FRED-MD has a monthly frequency (already starting in 1959), compared to a quarterly frequency (only starting from 1970) of the AWM Database.

Our output (dependent) variable will be the Unemployment Rate (URX) as a percentage of civilian workers (all ages, both female and male). This is seasonally adjusted, but not working day adjusted data. The input variables are all 45 other macroeconomic variables (see appendix).

Forecasting (regression) is a supervised learning problem (cf. supra), which means that before we can use Machine Learning algorithms, we will need to convert our time series dataset to a supervised learning dataset (Brownlee, 2017). In other words, our time series data must be re-framed. We do this using the *sliding window method* or *lag method*. Basically, you use previous time steps as input variables and use the next time step as the output variable. An example is shown in *table 3* (Brownlee, 2016). Here we can see that the previous time step is the input (X) and the next time step is the output (y) in our supervised learning problem. Further we must note that we have no previous value that we can use to predict the first value in the sequence. This means that we will delete this row as we cannot use it. We can also see that we do not have a known next value to predict for the last value in the sequence. We may want to delete this value while training our supervised model also.

<table>
<thead>
<tr>
<th>TIME</th>
<th>MEASURE</th>
<th>X</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>?</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>110</td>
<td>100</td>
<td>110</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>110</td>
<td>108</td>
</tr>
<tr>
<td>4</td>
<td>115</td>
<td>108</td>
<td>115</td>
</tr>
<tr>
<td>5</td>
<td>120</td>
<td>115</td>
<td>120</td>
</tr>
</tbody>
</table>

*Table 3: Time Series (left) versus Supervised Learning Problem (right) (based on Brownlee (2016))*
The same example can be given with multivariate time series. This is important, since our real-life application will also be conducted with multivariate time series. For the example in table 4, we are concerned with predicting MEASURE 2. The window width here is one, meaning that we will use the previous time step values of MEASURE 1 and MEASURE 2, and the next time step value for MEASURE 2.

<table>
<thead>
<tr>
<th>TIME</th>
<th>MEASURE 1</th>
<th>MEASURE 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2</td>
<td>88</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>89</td>
</tr>
<tr>
<td>3</td>
<td>0.7</td>
<td>87</td>
</tr>
<tr>
<td>4</td>
<td>0.4</td>
<td>88</td>
</tr>
<tr>
<td>5</td>
<td>1.0</td>
<td>90</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>?</td>
<td>0.2</td>
<td>88</td>
</tr>
<tr>
<td>0.2</td>
<td>88</td>
<td>0.5</td>
<td>89</td>
</tr>
<tr>
<td>0.5</td>
<td>89</td>
<td>0.7</td>
<td>87</td>
</tr>
<tr>
<td>0.7</td>
<td>87</td>
<td>0.4</td>
<td>88</td>
</tr>
<tr>
<td>0.4</td>
<td>88</td>
<td>1.0</td>
<td>90</td>
</tr>
<tr>
<td>1.0</td>
<td>90</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

*Table 4: Predicting MEASURE 2 with Multivariate Time Series (based on Brownlee (2016))*

The preparation of our data will in a first step involve the sliding window method. Once this is done, there are still a few steps left before we can implement our Machine Learn algorithm: handling the missing data, rescaling or standardizing the data, and splitting the data into the training and test set.

Handling missing data can be a tricky task. Building your model, you are always given two options: deletion or imputation. ‘The more data, the better’ implies a preference for imputation, however doing so isn’t always as obvious. Figure 20 shows the different options when confronted with missing data. Not all will be explicitly explained here, only those which we were confronted with.

There are 8 variables showing missing data, all at the start of measurements: HEX, HEG, HICPSYA, HEXSYA, HEGSYA, HEGWEI, CAN_YEN, and NFN_YEN44. Since all eight of these variables show a rather linear course (both individually and in relationship to each other), the obvious thing to do seemed linear interpolation. Here, several predictors of the variable with missing values are identified using a correlation matrix (Swalin, 2018). The best predictors are selected and used as

---

44 See the appendix for the full names of these variables.
independent variables in a regression equation. The variable with missing data is used as the dependent variable. This regression equation is sequential used to predict the missing values.

In our model we chose to discard the missing data instead. First, because linear interpolation carries some drawbacks, such as the deflation of standard errors\footnote{When you predict missing values from other variables, they tend to fit together too well.}. Second, because we assumed these variables were either unimportant to the forecasting of unemployment or already implicitly represented in other variables. Discarding missing data can either be done by deleting columns or deleting rows. Since deleting rows (deleting observations) would have halved our dataset, we chose to delete columns and thus removed these variables out of the dataset.

Finally, a choice was made on the splitting ratio between training and test data\footnote{Note on (Cross-)Validation: Initially, we did not explicitly incorporate a validation set. Having only 192 observations, adding another split would deplete any satisfactory variance in our estimates. Moreover, our ElasticNet class (sklearn) chooses the best hyperparameter values for us. Another solution is to go with cross-validation. In a later stage, we could replace the ElasticNet class with the ElasticNetCV class. This model is able to set the hyperparameters (alpha and lambda) by cross-validation. Doing so we avoid overfitting.}. This split ratio is specific to each case and there does not exist a clear rule-of-thumb. We chose to opt for a ratio of 80% training data and 20% test data since the number of observations is rather small. Also note that we did not standardize or rescale the data.
The dataset is now ready to use. We shifted the data from a time series dataset to a supervised learning dataset, handled missing data, and chose an appropriate split ratio. In a next section, the implementation of the Elastic Net model will be discussed, as so the building of our AR model.

The Set-Up

This section will extensively discuss the implementation of our simplest model: an Elastic Net model without cross-validation, using the dataset with deleted columns. Furthermore, the selected Autoregressive (AR) model will be briefly discussed.

Having already prepared the dataset, most of the work was already done. For the data analysis we used Pandas, a toolkit which provides high-performance, easy-to-use data structures and data analysis tools for the Python programming language (McKinney, 2008). Since we are dealing with a multivariate forecasting problem (i.e., we have observations of multiple different measures and an interest in forecasting one of them), the idea was to map all of the macroeconomic variables (X) on the unemployment rate (Y) and to explore the number of lags as a hyperparameter. Or in other words, we reframed the multivariate time series into (one-step and multi-step) supervised learning problems.

We used a self-written function\(^{47}\) to transform sequential data to a Pandas data frame. This function enabled us to set deliberately both the number of input lags and the horizon to be forecasted as parameters. Unemployment rate forecasts were formed for 3 months ahead (one-step), 6 months, 9 months, and 1 year (multi-step). Furthermore, the number of input lags was played with, enabling us to select the best performing model. However, for the results presented in this dissertation, we chose to work with only one lag. It is, nonetheless, important to realise that the algorithm used all datapoints to learn the mapping function, but to predict the next quarterly unemployment rate (t+1), it only uses the variables at time t as an input. The results will be presented in the following section.

\(^{47}\) Series_to_supervised() based on Brownlee (2017)
An Autoregressive (AR) model was also developed to serve as a benchmark model. In an AR(p) model, URX_t depends on its own lagged values (and a white noise error term), where p denotes the number of lags. No other variables are included. In other words, the value for URX_t is a linear combination of past values plus a white noise error term ε_t. We estimated 10 AR models (from AR(1) to AR(10)) using Maximum Likelihood estimation\(^{48}\). Afterwards, we selected the best model using the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC). Both AIC and BIC pointed towards a maximum lag length of 3 quarters (AR(3)).

In this brief section, the set-up of our Elastic Net model in Python, together with the development of an AR model in R, were discussed. When presenting the results in the next section, we will thus compare the forecasting performance of our Elastic Net model with the forecasting performance of an AR(3) model.

Results

Time to present the results. Forecasting accuracy is represented by the Mean Absolute Error (MAE) for the Elastic Net model, the AR model, and the consensus forecast (SPF). In addition, we also look at which variables the Elastic Net model identifies.

Table 5 shows the MAE for our three forecasting methods. The Mean Absolute Error was chosen for model evaluation because the same weight is given to all errors. In other words, it does not allow large positive errors to offset large negative errors.

<table>
<thead>
<tr>
<th>Horizon (quarters)</th>
<th>AR</th>
<th>SPF</th>
<th>Elastic Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.12</td>
<td>0.22</td>
<td>0.18</td>
</tr>
<tr>
<td>2</td>
<td>0.27</td>
<td>0.33</td>
<td>0.27</td>
</tr>
<tr>
<td>3</td>
<td>0.45</td>
<td>0.45</td>
<td>0.37</td>
</tr>
<tr>
<td>4</td>
<td>0.67</td>
<td>0.56</td>
<td>0.47</td>
</tr>
</tbody>
</table>

Table 5: MAE Results for all Horizons with each of the Three Methods (source: Federal Reserve Bank of Philadelphia (2019) and own calculations)

---

\(^{48}\) Note on estimating AR models: we used the Maximum Likelihood (ML) option rather than the Conditional Sum of Squares (CSS) or CSS-ML method since ML naturally uses the full sample. This means T is constant over all models and thus their likelihoods and information criteria are comparable.
Before discussing the results, some remarks are needed. First, a comparison between an AR model and an Elastic Net model is difficult since the first one is a time series model and the second one is a supervised model. We tried to take this issue into account by creating a rolling window forecast for our AR model in which we always carried out predictions for respectively 1, 2, 3, and 4 quarters ahead starting from 2008\(^{49}\). After forecasting the next period, our AR model was re-estimated including the just forecasted period, now repeating this process for the next period. We did this for all years from 2008 until 2018 and calculated the MAE for each forecasting horizon respectively. Second, the MAEs for the SPF method were obtained from a document by the Federal Reserve Bank of Philadelphia, called *Error Statistics for the Survey of Professional Forecasters for Unemployment Rate* (Stark, 2019). Normally, to make comparisons as fair as possible we must subject all models to the same data environment. Note however that whilst the test set for the Elastic Net model runs from 2008 until 2018, the sample period for the calculations of the MAEs of the SPF model is from 1985 until 2017.

It can be observed that the Elastic Net model outperforms SPF forecasts for all horizons. Compared to the AR model, it only performs better for longer horizons. This makes sense, since in the short run, unemployment will mostly be affected by past unemployment rates. The further we go in the future, the more important other factors become. Yielding even more accurate forecasts than the SPF, it can be said that the Elastic Net model is able to learn the true underlying relationship between economic series. *Figure 21* gives the difference between the real values and the forecasted values in our test set for one quarter ahead. *Figure 22* gives the difference between the real values and the forecasted values in our test set for four quarters ahead.

One of Elastic Net’s advantages was that it is able to drive some coefficients to exactly zero (cf. Regularization). For one-step ahead forecasts, the Elastic Net model picked 14 variables and eliminated the other 23. *Total Employment (LNN), Labour Force (LFN), Employees (LEN),* and *Gross Fixed Capital Formation (ITR)* are contributing most to the Unemployment Rate. For multi-step ahead forecasts (horizon = 4), the Elastic Net model picked 12 variables and eliminated the other 25. *Here, Total Employment (LNN), General Government Final Consumption Expenditure (GCR), Gross Fixed Capital Formation (ITR), and Individual Consumption Expenditure are the most*

\(^{49}\) Note that we split our dataset chronologically, instead of randomly. This is why the starting data in of our test set is 2008.
important variables when forecasting the Unemployment Rate. The relative importance of all variables is shown in Figure 23 & 24.

![Graph](image1)

*Figure 21: Actual Unemployment (blue) versus Forecasted Unemployment (orange) at 1-Quarter Horizon*

![Graph](image2)

*Figure 22: Actual Unemployment (blue) versus Forecasted Unemployment (orange) at 4-Quarters Horizon*

When finally taking the theory to practice, we were able to show that Machine Learning models do in fact yield highly accurate forecasts. Even though admitting that the model presented here was very basic, it succeeded in beating consensus forecasts up to a one-year horizon. This, however, should serve as a stepping stone for further research, and is just the mere basis for what is yet to come.

---

50 Nothing is said on causality.
Figure 23: Coefficients in the Elastic Net model at 1-Quarter Horizon

Figure 24: Coefficients in the Elastic Net model at 4-Quarter Horizon
Conclusion and Avenues for Further Development

Just like the universe contains some sort of elusiveness, so does the infinite amount of data. However, humanity is enshrined in an unbreakable urge for omniscience. As the volume of data available grows, increasingly complex techniques to manipulate and analyze big data find their way.

Already having an established value in computer science and natural sciences, Machine Learning has been making its inroads in social sciences, and more specifically economics. This is precisely why the main purpose of this dissertation was to serve as an introduction for economic scientists new to these concepts and techniques. Machine Learning and Deep Learning were both proposed as new and highly accurate forecasting methods. Moreover, we also tried to vouch for these theoretical findings by conducting a real-life experiment in which we tried to forecast European unemployment.

Using an Elastic Net model, the findings were in line with the expectations; forecasting accuracy was relatively higher for the Elastic Net model when compared the consensus forecast (SPF). One surprise however was that our AR(3) model was better performing for short horizons. Two reasons can be given for this observation. First, as said before, in the short run unemployment will mostly be affected by past unemployment rates. It’s only for longer horizons that other variables will start to become important. Second, remember that we only discussed the most basic Elastic Net model. We can further increase accuracy by: (i) choosing imputation rather than deletion for the missing data; (ii) setting the hyperparameters (e.g. lambda and alpha) with the help of cross-validation; and (iii) possibly changing our training-test split ratio.

Furthermore, it should be noted that this was just a mere stepping stone for further research. We only investigated one linear ML model. Other linear models, but also other nonlinear, ensemble, and deep learning models, were left on the background.

A number of difficulties were encountered when carrying out our research. One of the most challenging -and highly unexpected- complications was finding the right dataset. Machine
Learning requires a lot of data, both in number of observations as in number of variables. The dataset we worked with, the AWM database, was possibly too limited. Especially when compared to the American counterpart, FRED-MD. This is why further research could, and should, be devoted to the development of a macroeconomic dataset with the goal of establishing a convenient starting point for empirical analysis that requires big data. The biggest challenge here is dealing with the general low number of observations for most traditional macroeconomic variables.

Also note that nothing was said on causality, we solely conducted predictions. This is not only a fact in this dissertation, but can be generally observed in Machine Learning experiments. Researchers are mostly concerned with prediction and causal inference fades into the background, and this is a problem. It’s a problem because causality is, and always will be, the fundamental question in macroeconomics. Luckily, there is a vast literature emerging, with Susan Athey at the forefront, which is concerned with causality in Machine Learning.

I conclude by citing Harari (2016, p. 65):

“Some complex systems, such as the weather, are oblivious to our predictions. The process of human development, in contrast, reacts to them. Indeed, the better our forecasts, the more reactions they engender. Hence paradoxically, as we accumulate more data and increase our computing power, events become wilder and more unexpected. The more we know, the less we can predict. Imagine, for example, that one day experts decipher the basic laws of the economy. Once this happens, banks, governments, investors and customers will begin to use this new knowledge to act in novel ways, and gain an edge over their competitors. For what is the use of new knowledge if it doesn’t lead to novel behaviours? Alas, once people change the way they behave, the economic theories become obsolete. We may know how the economy functioned in the past – but we no longer understand how it functions in the present, not to mention the future."

Of course, using ML algorithms, we won’t have to decipher the basic laws of the economy. We will simply adjust to the ever-changing laws of the economy. Luckily (or not), ML enables us to immediately react to these changes.
Sources


1A. (Stochastic) Gradient Descent

Gradient Descent (GD) is often perceived as the most popular learning algorithm in Machine Learning (Suryansh, 2018). It is how (almost) any ML model learns and is referred to as an optimization algorithm. In this subsection of the appendix, I’ll be briefly explaining how this learning algorithm works. Furthermore, we’ll be seeing some variations of the algorithm, such as Batch GD, Stochastic GD, and Mini Batch GD.

What actually is a gradient? With a gradient, you are capable of measuring the change in an output of a function when changing its inputs a little bit. It is what we call the slope of a function. The higher this certain slope, the faster the change in output. Meaning the model will be able to ‘learn’ at a faster rate (Donges, 2019). In mathematical terms, we call this the partial derivative.

Again, we are working with a certain cost function that we’ll try to minimize. I say ‘again’ because we’ve been doing the same for RSS functions (cf. supra). This cost function represents the error in predictions of your ML model. Most popular type of cost function is the Mean-Squared Error (MSE) cost function. The search here is for the lowest error value possible. Whilst searching for this minimum\(^1\), the parameters (or coefficients) of your model are being tweaked to the optimal model. In other words, Gradient Descent is a convex function. It is an iterative method used to find the values of the parameters of a function that minimizes the cost function as much as possible.

*Figure 25* clarifies these concepts. The cost function we will try to minimize is shown here as the u-shaped line. Having a two-dimensionally graph, means we are working with a model with only one parameter (or weight). The dot on the graph depicts the initial weight, the initial value of the parameter. The dashed line represents the Gradient, or the slope of the function. It shows in what direction we will have to move to reach the minimum.

\(^1\) We will be focusing on two-dimensional graphs and models with only one parameter, hence minimum.
But, how does this ‘tweaking’ work? In the previous paragraph we talked about the slope of a function. This slope always points to the local minimum. So, if we calculate the slope of our cost function – the error value of your ML model - , we know the direction we’ll need to move towards to. We are now one step closer to optimizing our model.

Next up is deciding how much we will nudge our parameters (weights). A new concept is introduced: the **Learning Rate**. This determines the magnitude of the steps that GD takes into the direction of the local minimum. How fast will we move towards the optimal weights? The perfect balance between a big learning rate and small learning rate is needed. A too big of a learning rate will make your steps constantly bounce between the hills of the valley (your cost function), whilst a too small of a learning rate will take too much time in finding the local minimum. This is visually shown in **Figure 26**.

Learning is finding the right weights (and bias$^{52}$). We calculate the Gradient, or distance towards the minimum, and take a step towards this local minimum. The size of this step is decided by your **hyperparameter**, or the Learning Rate. Every time you do this, these weights (and bias) are updated in order of the new point we’ve arrived at in the cost function. This process is repeated until the local minimum is reached.

---

$^{52}$ A bias for inactivity: How high is the weighted sum supposed to be before the neuron starts getting meaningfully active?
To summarize, I present to you the three most popular types of Gradient Descent: Batch GD, Stochastic GD, and Mini Batch GD. We mainly distinguish these based on the amount of data they use. This is represented by the term batch which denotes the total number of samples from a dataset that is used for calculating the gradient for each iteration.

The Batch Gradient Descent, also called vanilla GD, is in a computational point-of-view probably the most efficient. It calculates the error for each training example. Once all the training examples have been evaluated, the model gets updated. Thus, the batch is here taken to be the whole dataset. Even though this seems to be the most precise process, difficulties arise when your datasets get really huge. You always need to use every sample for completing only one iteration, and this needs to be repeated until the minimum is reached.

A more computationally expensive approach is the Stochastic Gradient Descent (SGD). Here, the weights are updated for each training example one by one. Meaning that the batch size in SGD is one to perform each iteration. These training examples are always randomly chosen (hence stochastic) and result in a very noisy path towards the local minimum. However, the minimum is often reached in a significantly shorter training time.

Note that our distinction between computationally efficient and expensive might be confusing. When we say that the Batch GD is computationally more efficient than the Stochastic GD, we mean that the Stochastic GD usually takes a higher number of iterations to reach the minimum. This is because of the randomness in descent. On the other hand, you will sometimes read that
the SGD is computationally less expensive. Reason for this is that you, in this case, only need one sample to calculate the gradient.

Finally, we have a look at the Mini Batch Gradient Descent, which is actually the perfect combination\textsuperscript{53} of both previous mentioned Gradient Descents. Here, the training dataset is divided into small batches. For each of these batches an update is performed. This algorithm is mostly used within Deep Learning.

2A. Decision Trees and Random Forests

Random Forests are very simple and straightforward Machine Learning algorithms (Donges, 2018). They are used for both classification and regressions tasks. The Random Forest is a collection of Decision Trees, so to define a Random Forest a prior introduction to Decision Trees is needed. Although not given much attention in this paper, decision tree learning is among the most popular methods for ML and data mining (Blockeel, 2018). In this subsection of the appendix, I’ll define random forests and related concepts, such as Decision Trees.

What’s in a name? Decision Trees use a tree-like model of decisions. A decision tree will be used visually and explicitly to represent decisions and decision making. In other words, we want to build a tree with a set of hierarchical decisions which eventually gives us a final result. This result will be our classification or regression prediction.

\textsuperscript{53} Perfect combination since it seeks to find balance between the robustness of stochastic GD and the efficiency of batch GD (Brownlee, 2017)
Figure 27 illustrates the terminology used for Decision Trees. As said, a decision tree is a tree-shaped structure. The nodes of this tree are called Internal Nodes if, and only if, they have outgoing edges (or branches). If this is not the case, they are called Leaves or Terminal Nodes (this is the final decision). The node without any incoming edges is called the Root. The number of roots is always one. This visual representation immediately indicates one of Decision Trees’ biggest advantages, namely being highly interpretable. It’s their clarity of information representation that makes them this big in the realm of ML (Seif, 2018).

Now, how do we create these Decision Tree models in Machine Learning? Two steps are necessary: Induction and Pruning. The first one, induction, is the building of the tree itself. Choices on all the different hierarchical decision boundaries are needed. This is done based on the data, but will often leave you with way too many features (or attributes, or columns). The structure of the decision tree will be too elaborate, which will lead to major overfitting. In comes Pruning, which is the process of removing unnecessary structure from a decision tree. The branches that make use of features having low importance are being removed (Gupta, 2017). This will suffice as an introduction, for the interested reader I would like to refer to Breiman et al. (1984).

Going further to Random Forests. As said, the Random Forest is a collection of Decision Trees. Moreover, Random Forest builds multiple Decision Trees and merges them together to get a more accurate and stable prediction (Donges, 2018).

With Decision Trees, the most important feature is searched for when splitting a node. What questions do we need to ask to lead to an accurate prediction? The same is applied to Random Forest, but now the best feature is selected from a random subset of features. The Random Forest will add additional randomness to the model. Put differently, each decision tree in the forest considers a random subset of features when forming questions and only has access to a random set of the training data points (Koehrsen, 2017). The eventual prediction will be based

---

54 In practice, this is done by training: Decision Trees and Random Forest are Supervised Machine Learning models. They learn to map data to outputs. In other words, the model learns any relationships between the data (known as features in machine learning) and the values we want to predict (called the target), which in turn leads to the formation of a certain structure of the Decision Tree (Koehrsen, 2017).
on the average\textsuperscript{55} of all individual decision tree estimates. See Figure 28 for a visualization of a Random Forest for a regression task.

![Random Forest with n Decision Trees for a Regression Task](image)

*Figure 28: Random Forest with n Decision Trees for a Regression Task (source: Kumar, 2018)*

Simple explanations on Decision Trees and Random Forest were given. Important to remember is that the Random Forest is an ensemble of Decision Trees, of which the features and training data were randomly chosen. For regression tasks, the outputs of n Decision Trees are averaged. For classification tasks, a major voting is taken.

### 3A. Backpropagation

In the first subsection of the Appendix, we discussed a certain optimization algorithm named Gradient Descent. It’s the algorithm that can determine in what direction we should adjust the weights and biases for Neural Networks. Essentially, the gradient of a cost function - representing the error of predictions in our ML model - was calculated. However, I did not explicitly say how this Gradient is computed. In this subsection of the appendix, we will see a fast algorithm to compute such gradients: the *backpropagation algorithm*.

What the algorithm actually does, is telling the Network whether the prediction has been correct or not. It *propagates* (or transmits) information related to the error produced by the Neural

\textsuperscript{55} Taking averages is for regression tasks. In the case of classification tasks, we will majority vote.
Network. But, to come to this point it first needs to propagate a data instance through the network’s parameters towards the prediction. When arriving at the output (the prediction) an output error can be measured. We can have a look at the distance between the prediction and the effective value, or the ground truth. Afterwards, the Neural Network backpropagates the information about the error through the network so that it can alter the parameters one step at a time.

Put in the words of Al-Masri (2019): “Backpropagation is the essence of neural net training. It is the practice of fine-tuning the weights of a neural net based on the error rate (i.e. loss) obtained in the previous epoch (i.e. iteration).” We do this by calculating the loss at every node in the Neural Network, because essentially the total error of a neural net is the error caused by all the nodes accumulated into one number. We look for the nodes that show the biggest error in every layer. In turn, we penalize these nodes by giving them smaller weights. After deciding on which nodes show the biggest error in every layer, it is time to update the weights. In comes the Gradient Descent formula.\(^56\) The Gradient Descent weight update is used on all the weights. That is to say, we calculate the gradient and then integrate this gradient to know how the weights should change, which is referred to as doing backpropagation.

Simply put, ANNs are fine-tuned by repeatedly guessing the output and in turn fixing the error to get a better guess. This fine-tuning is done by backpropagating the information through the system again to find a more fine-tuned error. It is the part of Gradient Descent where you infer the error and correct it that is known as backpropagation (Sood, 2018).

That will suffice as an introduction to backpropagation. I deliberately omitted the mathematical derivations as they would increasingly complicate an understanding of the concept.

---

\(^{56}\) W* = W – alpha . J'(W) where W is the weight at hand, alpha is the Learning Rate, and J'(W) is the partial derivative of the cost function J(W) with respect to W.
4A. Cook & Hall (2017): Figures of different architectures used

Below, the different architectures used in Cook and Hall’s *Macroeconomic Indicator Forecasting with Deep Neural Networks* (2017) are shown. Figure 29 portrays the Feed-Forward model. Figure 30 provides us with the convolutional model used. Figure 31 depicts the LSTM model as tested. Finally, figure 32 illustrates the Encoder-Decoder architecture that they use for testing.

Each rectangle always consists of a layer of neurons. The size of these rectangles in turn suggests the number of nodes in each layer.

*Figure 29: Feed-Forward (Fully Connected) Model Architecture*

*Figure 30: Convolutional Model Architecture*
Figure 31: LSTM Model as Tested

Figure 32: Encoder-Decoder Diagram
### 5A. Area-Wide Model Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CAN_YEN</strong></td>
<td>Current Account Balance as a Share of GDP. Calculated as the ratio of the sum of balance of trade (exports minus imports) and net factor income from abroad, and nominal GDP [CAN_YEN = (Balance of Trade + NFN_YEN - YEN) / YEN].</td>
</tr>
<tr>
<td><strong>COMPRI</strong></td>
<td>Commodity Prices. U.S. dollars. Calculated as the weighted sum of all prices and non-oil commodity prices.</td>
</tr>
<tr>
<td><strong>EEN</strong></td>
<td>Nominal Effective Exchange Rate (NER). Euro Euro area-19 countries vis-a-vis the NER-38 group of main trading partners. Base year 1999 (1999Q1 = 1).</td>
</tr>
<tr>
<td><strong>EXR</strong></td>
<td>Euro-per-USD Exchange Rate.</td>
</tr>
<tr>
<td><strong>GCD</strong></td>
<td>General Government Final Consumption Deflator. Index, Index base year 1996 (1996 = 1). Defined as the ratio of nominal, and real general government consumption expenditure.</td>
</tr>
<tr>
<td><strong>GCR</strong></td>
<td>General Government Final Consumption Deflator. Index, Index base year 1996 (1996 = 1). Defined as the ratio of nominal, and real general government consumption expenditure.</td>
</tr>
<tr>
<td><strong>GON</strong></td>
<td>Gross Operating Surplus. Calculated as the residual term of the difference between nominal GDP and the sum of compensation of employees and taxes on production and imports net of subsidies [GON = YEN - WNT - TN].</td>
</tr>
<tr>
<td><strong>HEG</strong></td>
<td>HICP Energy. Index, Neither seasonally nor working day adjusted data, Index base year 1996 (1996 = 100).</td>
</tr>
<tr>
<td><strong>HBSSTA</strong></td>
<td>HICP Energy. Index, Seasonally and working day adjusted data, Index base year 2015 (2015 = 100).</td>
</tr>
<tr>
<td><strong>HBSWEE</strong></td>
<td>Weight of the HICP Energy on Overall HICP. Parts per thousand. MNP total = 1000.</td>
</tr>
<tr>
<td><strong>HEX</strong></td>
<td>HICP - All Items Excluding Energy. Index, Neither seasonally nor working day adjusted data, Index base year 1996 (1996 = 100).</td>
</tr>
<tr>
<td><strong>HEXSTA</strong></td>
<td>HICP - All Items Excluding Energy. Index, Working day and seasonally adjusted data, Index base year 2015 (2015 = 100).</td>
</tr>
<tr>
<td><strong>HICP</strong></td>
<td>HICP - Overall Index, Index, Neither seasonally nor working day adjusted data, Index base year 1996 (1996 = 100).</td>
</tr>
<tr>
<td><strong>HICPSY</strong></td>
<td>HICP - Overall Index, Index, Working day and seasonally adjusted data, Index base year 2015 (2015 = 100).</td>
</tr>
<tr>
<td><strong>ITD</strong></td>
<td>Gross Fixed Capital Formation Deflator, Index, Index base year 1995 (1995 = 1). Defined as the ratio of nominal, and real gross fixed capital formation.</td>
</tr>
<tr>
<td><strong>ITR</strong></td>
<td>Gross Fixed Capital Formation. Millions of euros, Chain linked volume, Calendar and seasonally adjusted data. Reference year 1995.</td>
</tr>
<tr>
<td><strong>LEN</strong></td>
<td>Employees. Thousands of persons, Calendar and seasonally adjusted data.</td>
</tr>
<tr>
<td><strong>LFN</strong></td>
<td>Labour Force. Thousands of persons. Defined from total employment and the employment rate [LFN = LWN / (1 - URX)].</td>
</tr>
<tr>
<td><strong>LNN</strong></td>
<td>Total Employment. Thousands of persons, Calendar and seasonally adjusted data.</td>
</tr>
<tr>
<td><strong>LPPROD</strong></td>
<td>Labour Productivity. Calculated as the ratio of real GDP, and total employment [LPPROD = YER / LWN].</td>
</tr>
<tr>
<td><strong>LTN</strong></td>
<td>Nominal Long-Term Interest Rate. Euro area 10-year Government Benchmark bond yield. Percent per annum.</td>
</tr>
<tr>
<td><strong>MTD</strong></td>
<td>Imports of Goods and Services Deflator, Index, Index base year 1996 (1996 = 1). Defined as the ratio of nominal, and real imports of goods and services. Based on the gross concept, i.e., both extra- and intra-area trade flows are accounted for.</td>
</tr>
<tr>
<td><strong>MTR</strong></td>
<td>Imports of Goods and Services, Millions of euros, Chain linked volume, Calendar and seasonally adjusted data, Reference year 1995. Based on the gross concept, i.e., both extra- and intra-area trade flows are accounted for.</td>
</tr>
<tr>
<td><strong>NFCN</strong></td>
<td>Net Factor Income from Abroad. As a Share of GDP. Defined as the ratio of the sum of primary-income balance (Balance of Payments and International Investment Position), secondary-income balance (Balance of Payments and International Investment Position), and the capital-account balance, and nominal GDP [NFCN = (Primary-income Balance + Secondary-Income Balance + Capital-account Balance) / YEN].</td>
</tr>
<tr>
<td><strong>PCD</strong></td>
<td>Individual Consumption Deflator, Index, Index base year 1995 (1995 = 1). Defined as the ratio of nominal, and real individual consumption expenditure.</td>
</tr>
<tr>
<td><strong>PCOMIU</strong></td>
<td>Non-oil Commodity Prices, ECB commodity price index US dollar denominated, Import weighted, Total non-energy commodity. Neither seasonally nor working day adjusted data.</td>
</tr>
<tr>
<td><strong>PCR</strong></td>
<td>Individual Consumption Expenditure, Millions of euros, Chain linked volume, Calendar and seasonally adjusted data. Reference year 1995.</td>
</tr>
<tr>
<td><strong>SAX</strong></td>
<td>Gross Household Saving Rate. Percentage. Calendar and seasonally adjusted data. Defined as the ratio (multiplied by 100) of gross saving, and gross disposable income adjusted for the change in the net equity of households in pension funds reserves [SA.X = (Gross Saving + Gross Disposable Income + Net Equity of Households in Pension Funds Reserves)] * 100.</td>
</tr>
<tr>
<td><strong>STN</strong></td>
<td>Nominal Short-Term Interest Rate, Euro area 3-month, Percent per annum, Last trade price.</td>
</tr>
<tr>
<td><strong>TIM</strong></td>
<td>Taxes on Production and Imports Less Subsidies. Millions of euros. Current prices, Calendar and seasonally adjusted data.</td>
</tr>
<tr>
<td><strong>ULC</strong></td>
<td>Unit Labour Costs. Calculated as the ratio of compensation of employees, and real GDP [ULC = WNN / YER].</td>
</tr>
<tr>
<td><strong>UNN</strong></td>
<td>Number of Unemployed. Thousands of persons, Total (all ages), Total (male and female), Seasonally adjusted but not working day adjusted data.</td>
</tr>
<tr>
<td><strong>URX</strong></td>
<td>Unemployment Rate. Percentage of civilian workforce, Total (all ages), Total (male and female), Seasonally adjusted, but not working day adjusted data.</td>
</tr>
<tr>
<td><strong>WIN</strong></td>
<td>Compensation of Employees. Millions of euros. Current prices, Calendar and seasonally adjusted data.</td>
</tr>
<tr>
<td><strong>WNN</strong></td>
<td>Wage per Hour. Calculated as the ratio of compensation of employees, and total employment [WNN = WNN / LWN].</td>
</tr>
<tr>
<td><strong>XTD</strong></td>
<td>Exports of Goods and Services Deflator, Index, Index base year 1995 (1995 = 1). Defined as the ratio of nominal, and real exports of goods and services. Based on the gross concept, i.e., both extra- and intra-area trade flows are accounted for.</td>
</tr>
<tr>
<td><strong>XTR</strong></td>
<td>Exports of Goods and Services, Millions of euros, Chain linked volume, Calendar and seasonally adjusted data, Reference year 1995. Based on the gross concept, i.e., both extra- and intra-area trade flows are accounted for.</td>
</tr>
<tr>
<td><strong>YED</strong></td>
<td>GDP Deflator. Index, Index base year 1995 (1995 = 1). Defined as the ratio of nominal, and real gross domestic product (GDP).</td>
</tr>
<tr>
<td><strong>YER</strong></td>
<td>GDP. Public Product (GDP) at market prices, Million Euro. Chain linked volume, Calendar and seasonally adjusted data. Reference year 1995.</td>
</tr>
<tr>
<td><strong>YFD</strong></td>
<td>GDP at Factor Costs Deflator. Index. Defined as the ratio of nominal, and real GDP at factor costs.</td>
</tr>
<tr>
<td><strong>YFN</strong></td>
<td>GDP at Factor Costs. Calculated as the sum of compensation to employees and gross operating surplus (YFN = WIN + GON).</td>
</tr>
<tr>
<td><strong>YIN</strong></td>
<td>GDP, Income Side. Calculated as the sum of GDP at factor costs and taxes on production and imports less subsidies (YIN = YFN + TIM).</td>
</tr>
<tr>
<td><strong>YWD</strong></td>
<td>World 1 GDP Deflator, Index, Index base year 1995 (1995 = 1). Defined as the ratio of nominal, and real &quot;world&quot; GDP.</td>
</tr>
<tr>
<td><strong>YWDX</strong></td>
<td>World Demand Deflator. Composite Indicator. Calculated as the weighted sum of &quot;world&quot; GDP deflator expressed in euros and the euro area export deflator [log(YWDX) = w[YWD] * EEN + EEE + w[XFT] * log[XFT]].</td>
</tr>
<tr>
<td><strong>YWR</strong></td>
<td>World 1 GDP. Millions of US dollars. Calculated as the weighted sum of the GDP of the main trading partners of the Euro Area at the time of the creation of the model. These countries are the US, the UK, Japan and Switzerland. log(YWR) = w[US] * log(GDP[US]) + w[UK] * log(GDP[UK]) + w[CH] * log(GDP[CH]), where w[US] + w[UK] + w[CH] = 1.</td>
</tr>
<tr>
<td><strong>YWRX</strong></td>
<td>World Demand. Composite Indicator. Calculated as the weighted sum of &quot;world&quot; GDP and domestic demand net of exports for the euro area [log(YWRX) = w[YWR] * log(YWR) + w[FDX-XTR] * log[FDCD-XTR], where w[YWR] + w[FDX-XTR] = 1].</td>
</tr>
</tbody>
</table>

*Figure 33: AWM Variables (source: ECB (2018))*