BANK OF ENGLAND

Staff Working Paper No. 674

Machine learning at central banks

Chiranjit Chakraborty and Andreas Joseph

September 2017

BANK OF ENGLAND

Staff Working Paper No. 674
Machine learning at central banks

Chiranjit Chakraborty[1] and Andreas Joseph[2]

Abstract

We introduce machine learning in the context of central banking and policy analyses. Our aim is to give an overview broad enough to allow the reader to place machine learning within the wider range of statistical modelling and computational analyses, and provide an idea of its scope and limitations. We review the underlying technical sources and the nascent literature applying machine learning to economic and policy problems. We present popular modelling approaches, such as artificial neural networks, tree-based models, support vector machines, recommender systems and different clustering techniques. Important concepts like the bias-variance trade-off, optimal model complexity, regularisation and cross-validation are discussed to enrich the econometrics toolbox in their own right. We present three case studies relevant to central bank policy, financial regulation and economic modelling more widely. First, we model the detection of alerts on the balance sheets of financial institutions in the context of banking supervision. Second, we perform a projection exercise for UK CPI inflation on a medium-term horizon of two years. Here, we introduce a simple training-testing framework for time series analyses. Third, we investigate the funding patterns of technology start-ups with the aim to detect potentially disruptive innovators in financial technology. Machine learning models generally outperform traditional modelling approaches in prediction tasks, while open research questions remain with regard to their causal inference properties.

Key words: Machine learning, artificial intelligence, big data, econometrics, forecasting, inflation, financial markets, banking supervision, financial technology.

JEL classification: A12, A33, C14, C38, C44, C45, C51, C52, C53, C54, C55, C61, C63, C87, E37, E58, G17, Y20.

(1) Bank of England. Email: chiranjit.chakraborty@bankofengland.co.uk
(2) Bank of England. Email: andreas.joseph@bankofengland.co.uk

The views expressed in this paper are those of the authors, and not necessarily those of the Bank of England or any of its committees. We are grateful to colleagues and friends who give many valuable comments during the development process of this paper. Particular thanks to David Bholat, Paul Robinson, James Barker, Eryk Walczak, Jonathan Fullwood, Aidan Saggers, George Kapetanios, Djyldyz Djumalieva and Jialin Chen. All (forecasting) errors are those of the authors.

Information on the Bank's working paper series can be found at
www.bankofengland.co.uk/research/Pages/workingpapers/default.aspx

Publications and Design Team, Bank of England, Threadneedle Street, London, EC2R 8AH
Telephone +44 (0)20 7601 4030 email publications@bankofengland.co.uk

ISSN 1749-9135 (on-line)

Contents

1 Introduction

We are seeing a critical juncture of two mega-trends. On the one side, there are large advances in data-driven modelling techniques. Applications built on *machine learning*, which combine elements from computational statistics, mathematical optimisation, pattern recognition, predictive analytics and artificial intelligence, are transforming everyday life. Processing and storage capabilities have increased dramatically alongside the advancement of analytical tools. On the other side, there is a rapidly increasing amount of granular data, often referred to as *Big Data*[1]. Recently available data cover a very broad range of sources, such as online markets and social media, but also a multitude of offline activities traceable through various means of payment, communication, transportation, etc. (Mayer-Schönberger and Cukier (2013)). Most of these developments have been driven by the private sector. However, there is no reason why the public sector and (economic) policy makers could not and should not equally benefit from them (Bholat (2015); Einav and Levin (2013)).

Intuitively, the movements of the macro-economy, which central banks are particularly interested in (Bank of England (2015)), must be the outcome of microeconomic decisions and interactions. Provided appropriate granular data and adequate analytical tools are used, it should be possible to trace these actions and their aggregated effects. Many central banks and regulators have been endowed with new responsibilities in terms of supervision and market oversight since the global financial crisis 2008/09. These responsibilities came with the collection and access to a wealth of new data sources, which moves central banks into the realm of Big Data. These are not just bigger in volume, but also contain more granular information, come in a wider range of formats, e.g. text, and are updated more frequently. Key examples include the access to transaction-level data in over-the-counter derivatives markets (Cielinska et al. (2017)), the evaluation of prepositioned collateral by financial institutions and detailed information on their balance sheets, as well as an increasing set of micro-market datasets ranging from mortgages, over news sentiments to developments in financial technology (fintech; Bholat and Chakraborty (2017)).

In this paper, we discuss the above juncture from a technical point of view. We introduce practical aspects of machine learning and how the associated methodologies can be applied in the context of central banking and policy analysis. Our aim is to give an overview broad enough to allow the reader to place machine learning within the wider range of statistical and computational analyses and provide an idea of its scope and the limitations of different techniques. The relevant literature is provided along the way, where we give an overview of the underlying technical sources and the nascent literature applying machine learning to economics and policy

[1]The term Big Data is often used ambiguously. We will give several technical definitions below.

problems. We present a series of relevant applications and discuss how these could feed into decision processes. These include three case studies relevant to central banks, which demonstrate and compare various modelling techniques and their potential applications. They range from the operational to the conceptual. Larger gains are expected to emerge first at the operational end of the spectrum by the improvement of work flows. Conceptual aspects are an area of active research and may prove particular insightful in combinations with novel granular datasets.

First, we model the occurrence of hypothetical alert scenarios on the balance sheets of financial institutions in an environment of incomplete information, i.e. only using a subset of the available information. We demonstrate that machine learning models generally outperform conventional approaches in this setting of banking supervision as well as simple heuristics.

The second case study presents a modelling framework to forecast UK CPI inflation on a medium-term horizon of two years. For this, we introduce an approach to use machine learning techniques in a time series setting. A simple forecast combination (Timmermann (2006)) of advanced machine learning models is seen to perform the best. Methods of model introspection, what we call "model forensics", allow to obtain a basic understanding of input-output relations for complex model types. This is important for two reasons. First, machine learning approaches are often criticised for being opaque or of a "black-box nature", which would hinder their applicability to inform decisions. Second, statistical inference is still an open research question for advanced machine learning models, such that the significance of a variable cannot be assessed by standard statistical tests.

In the third case study, we investigate global funding relations between firms and investors within the technology sector. This ecosystem of financial relations exhibits a very rich structure pointing to different modes of funding. Clustering techniques are being used to identify high-potential start-ups by comparing them to previously successful firms, the so-called *unicorns*[2]. This analysis is performed with a special focus on the rapidly growing industry of financial technology (fintech), where spotting potentially disruptive technological trends is of relevance to long-term financial stability.

The remainder of this work is structured as follows. Section 2 introduces general concepts of the machine learning toolbox, such as problem classes and learning types. We define the general notion of a machine learning system, discussing different ingredients and important aspects. Section 3 gives an overview of the most common machine learning models, highlighting the potential merits and limitations of different techniques. These include, among others, artificial neural networks, tree models, random forests and support vector machines. Section 4 introduces

[2]Firms with a valuation of at least one billion USD.

important concepts such as the bias-variance trade-off, optimal model complexity, regularisation and cross-validation. A detailed description and evaluation of all three case studies is given in Section 5. We summarise and conclude in Section 6. Finally, a concise model summary is given in the Appendix.

2 Machine learning concepts

2.1 Learning systems

There is not one machine learning model. Rather, every *machine learning system* consists of a set of components: (1) a problem, (2) a data source, (3) a model, (4) an optimisation algorithm and (5) validation & testing. We briefly introduce the role of each component, while a more detailed formal treatment will be given in subsequent sections.

Machine learning techniques are concerned with general pattern recognition or the construction of universal approximators (Cybenko (1989)) of relations in the data in situations where no obvious a priori analytical solution exists. To help judge a situation when machine learning may provide benefits, the following section provides key definitions and standard formulations, surrounding the problem and data characteristics (1, 2).

Point (3) concerns the model type, such as logistic regression or artificial neural networks (see Section 3). The given dataset and problem at hand often suggest certain choices as we will see in the case studies. The in-depth treatment of optimisation algorithms (4) is outside the scope of this paper. Efficient off-the-shelf implementations of optimisation algorithms for common machine learning models are included in packages for common programming languages such as `R` and `Python`. However, we also want to stress that one should keep in mind that optimisation algorithms can run into problems and return far-from-optimal solutions. This may happen when model *or* algorithms parameters are set inappropriately or if there exist multiple good solutions, i.e. multiple local optima in a non-convex optimisation problem (further discussed below).

A main concern in machine learning is a model's *generalisation properties*, i.e. its performance according to some criterion applied to unseen data, hence *out-of-sample testing*. Commonly used models, such as ordinary least square (OLS) regression models, may have poor generalisation properties. To get around this, machine learning heavily focuses on model calibration (*validation*) and evaluation (*testing*) (5). Validation involves the tuning of hyper-parameters or the inclusion of a meta algorithm to enhance test performance which we discuss Section 4. This leads to a tripartite modelling cycle of *training*, or fitting a model, and the subsequent validation and testing.

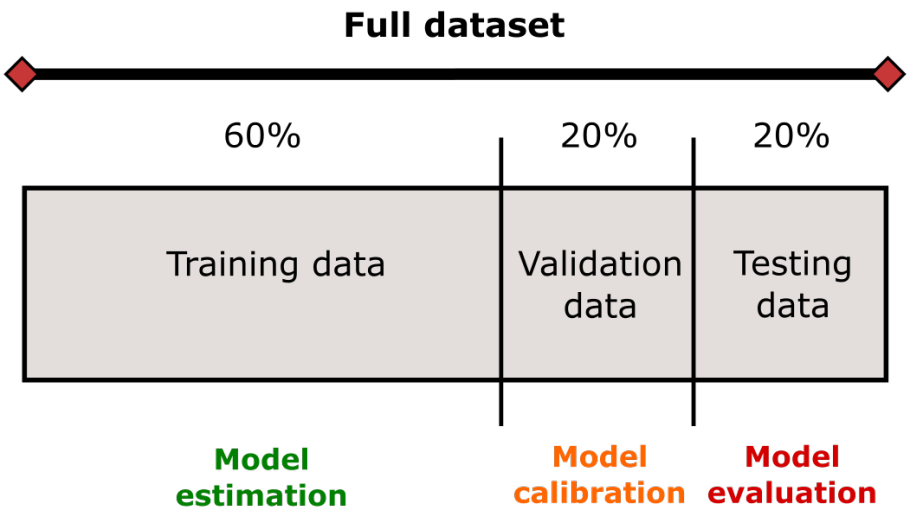

Figure 1: Schematic representation of the three stages of model generation in machine learning: training (model fit), validation (calibration) and testing (evaluation). The given data fraction offer a starting point if no cross-validation is involved (see Section 4.3).

Here, an important aspect of machine learning is that these three parts are done on different parts of a dataset to avoid over-confident conclusion from one's model. The strict separation of the test data sample is one of the key principles when building a machine learning system. This is shown in Fig. 1, where the given proportions offer a reasonable starting point for data fraction which may be used to perform the three analysis steps. This allows one to consistently estimate the generalisation properties of the final model based on the test sample. Note that splitting of a dataset should always be done on the basis of random sampling to avoid biases. Special care may be taken when faced with time series data, which will be briefly discussed in Section 4.6.

2.2 Policy problems and prediction

Machine learning applications have been traditionally built around prediction, where tracing down sufficiently strong correlations between variables is good enough. Predictions stemming from machine learning techniques may be more accurate than those derived from conventional approaches (Bajari et al. (2015)). Conversely, econometric and, especially, policy analysis is built around causal inference. An elaborate set of tools has been developed in econometrics for this, such as instrumental variables, regression discontinuity, difference-in-difference analysis, as well as natural and designed experiments (Angrist and Pischke (2008); Varian (2014)). Causality is usually inferred by comparing the effect of a policy (or a general treatment) with its counter-factual, ideally given by a randomised control group. In this sense, machine learn-

ing and econometrics can be seen as mutual extensions of each other. Recent developments of inference techniques from machine learning approaches are given in Athey and Imbens (2015); Wager and Athey (2015); Belloni et al. (2013).

More generally, a policy problem can be divided into a *prediction* and a *causal inference* component. Following the lines of Kleinberg et al. (2015), we may assume that a known payoff function π depends on a policy variable X, an outcome Y and other control factors Z, which are assumed independent of X and Y. Decisions with respect to X depend then on the total derivative

$$\frac{d\pi(X,Y,Z)}{dX} = \left(\frac{\partial \pi}{\partial X} \underbrace{\Big|_Y}_{\text{prediction}} + \frac{\partial \pi}{\partial Y} \underbrace{\frac{\partial Y}{\partial X}}_{\text{causal inference}} \right) \Big|_Z . \tag{1}$$

Assuming we know the controls Z, the two unknowns in this equation are Y and $\partial Y/\partial X$, corresponding to the prediction and inference component, respectively.

Assuming also that our policy action cannot influence the outcome, i.e. $\partial Y/\partial X = 0$, we are left with a pure prediction problem, where a precise knowledge of Y provides the information on the exact payoff of a policy action X (machine learning realm). Assuming we know Y, inference problems are then given when we are interested in the strength of the relation between the outcome and the policy variable $\partial Y/\partial X \equiv \beta$, where the paramater, mostly a coefficient β, describes the strength between the relation between the outcome and the policy variable (econometrics realm). Note that a machine learning model will also produce values for β, which will, however, be biased in the general case (see Section 4.2) and have to be interpreted with caution. The above formalisation makes clear that the general policy problem requires prediction and causal inference, as neither of the two terms may be assumed zero or assumed to be known.

A brief discussion of capital requirements for banks illustrates the emergence of and difference between the two types of policy problems in a central banking context. Let national income be the outcome variable Y within a social welfare function π (payoff) depending on Y among other factors. Let the level of equity funding required be the policy variable X. At normal times, an often raised question regarding capital requirements is if and to what extent they are a drag on the financial and, consequently, on the real sector of the economy, potentially lowering income levels (Brooke et al. (2015); Firestone et al. (2017); Dagher et al. (2016); Bridges et al. (2014)). This is an archetypal inference problem in a policy context. However, at times of financial turmoil or deep recessions, the prediction of drops in income may be crucial, given a certain level of capital requirements, e.g. by gauging the survival of banks. Both types of policy problems are being seen to arise jointly in this example, as predicted levels of income are likely to depend on the conditions and rules in the financial sector. Machine learning techniques are

likely to help answer the second type of question, while ways to address causal relations are a topic of active research (Athey and Imbens (2015); Wager and Athey (2015)). Also, traditional, two-stage instrumental variables approaches can be seen as a mixed problem (Mullainathan and Spiess (2017)). The first stage uses the instrument to predict the target, while the second stage infers the reaction to the policy variable only using the first-stage prediction.

2.3 Technical problem formulation

In this section, we provide a more formal description of machine learning problems. A comprehensive treatment of introductory and advanced material can be found in (Friedman et al. (2009); Abu-Mostafa et al. (2012); Ng (2015)). A machine learning problem is mathematically formulated as an optimisation problem, whose solution determines a set of model parameters:

$$F(X, Y, \beta; \lambda) \quad \xrightarrow[\text{algorithm}]{\text{optimisation}} \quad \beta\,. \tag{2}$$

X and Y are the input data and target variable, respectively. β represents the set of *model parameters* usually depending on the input and the model class. The model is estimated by optimising the *objective function* $F(\cdot, \beta)$, with respect to the parameters β (training). The set of hyper-parameters λ depends on the model class and is used to improve model test performance. It is determined via validation (calibration). A popular choice is the smoothing of model output (regularisation). The latter two aspects will be discussed in Section 4.

The input data X and target variable Y are of dimensions $m \times n$ and $m \times 1$, respectively,

$$X = \begin{pmatrix} (x_{1,1} & x_{1,2} & \cdots & x_{1,n}) \\ (x_{2,1} & x_{2,2} & \cdots & x_{2,n}) \\ \vdots & \vdots & \ddots & \vdots \\ (x_{m,1} & x_{m,2} & \cdots & x_{m,n}) \end{pmatrix} \quad \text{and} \quad Y = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}. \tag{3}$$

The *number of observations* is denoted by m and the *number of independent variables* by n. In machine learning, independent or right-hand-side (RHS) variables are denoted as *features*. We will use these expressions interchangeably. The use of single lower indices, especially for X, refers to single observations $x_i \in \mathbb{R}^{(1\times n)}$ if not stated otherwise. Note that the terminology in machine learning and econometrics varies, but often refers to the same entities. Features or inputs generally correspond to independent variables, covariates or the right-hand-side (RHS), while the output, target or response refers to the dependent variable or left-hand-side (LHS).

Note also that the number of features does not necessarily equal the number of independent explanatory variables in the raw data. Like other modelling approaches, machine learning techniques can include any functional form of one or several variables, such as polynomials or interaction terms, which are called *higher-order features*. Having only a moderate number of independent variables, the number of possible features that can be created from them is theoretically unbounded. This highlights the importance of adjusting for overfitting and feature selection in the process of algorithmic evaluation and model selection.

For the technical implementation of constructing X and Y, one has to account for different *variable types*. Variables can be of numeric (e.g., income), categorical (e.g., job category), ordinal (e.g., age bracket) or Boolean (e.g. employment status) type. The main difference between numeric and categorical variables is that categorical variables cannot be ordered, but only be mapped to a discrete set of size n_{cat} values. When representing categorical variables with $n_{cat} > 2$ within the input X and target Y it is common to represent them as n_{cat} dummy variables.

Convexity of the cost function, that is, that the problem in Eq. 2 can be written as a minimisation problem with a unique solution is generally desired. However, this cannot be guaranteed in general. In many circumstances, especially for more complex models, the existence of local minima and saddle points cannot be ruled out. Solutions to this situation include repeated random initialisation of algorithms and sampling.

2.4 Problem and learning types

2.4.1 Supervised and unsupervised learning

Supervised and unsupervised learning are the *two main learning types*. The difference between the two is the existence of a target variable Y. Supervised learning is the classical case of modelling Y using X. That is, the chosen learning algorithm tries to fit the target using the given input features. One also says that data are labelled, meaning that each input observation x_i is matched with an observed output observation y_i. Examples are the modelling of individuals' employment status given their mobile usage (Toole (2015)), the modelling of house prices based on house characteristic or inflation forecasting based on the macroeconomic environment (see Section 5.2).

There is no target Y is the case of unsupervised learning. Algorithms merely aim to find structure in the data, for example by grouping observations. These learning methods may also be called latent-variable-only models, as one does not pre-specify any model relations. In this sense, clustering-type problems are typically addressed by unsupervised learning where the learning method segments the input data X into groups according to a general distance measure. An

example would be the segmentation of individuals according to their consumption behaviour, which may then coincide with their employment status or any other socio-economic property. In practice, a mixture of both types of algorithms may be used to achieve a superior result. In some cases, unsupervised techniques can be used first to add labels (Y) to the observations in X or extract new features, such as the cluster affiliation. Then, supervised learning algorithms are used to work with these inputs[3] (see Section 5.3).

2.4.2 Classification and regression problems

Within the realm of supervised learning, classification and regression problems are the *two main problem classes*. The difference between the two lies with the type of output variable Y. In a classification problem, the output consists of a discrete set of outcomes which cannot be ordered. That is, each element of Y represents a class label. Like above, trying to infer the employment status of individuals (Y: employed or unemployed) on the basis of their communication or consumption habits (X) would be a typical classification problem. A learning method would return one of the two states or labels for each observation in x_i.

On the contrary, regression problems try to match and return a continuous output variable. An example of a typical regression problem would be the modelling of house prices or inflation. Note also that the meaning of the term *regression* in machine learning is different from its usage in econometrics. Regression in a machine learning sense refers to a problem class and not to a type of statistical model, such as a linear or logistic regression.

Again, the distinction between classification and regression problems may not be strict in all cases. Some questions may be be approached from either side. When modelling employment, one can imagine a continuous range between unemployment, under-employment and over-employment within a population (see e.g. Office for National Statistics (2014)) with fuzzy boundaries[4] between categories.

An overview of the taxonomy of learning types and problem classes is shown in Fig. 2. We present a case study for each of the three branches in Section 5.

[3] A third type of learning, which is probably closest to the human experience is *reinforcement learning*. Reinforcement learning can be found in *online learning* situations, where new inputs (X) are constantly arriving, but there is no exact (Y) to match them. Instead a given hypothesis is updated sequentially with the aim of maximising a payoff function, for example by comparing newly arriving inputs with previous values or lagged outputs Y with learned parameters. A natural appeal of reinforcement learning in a policy context is the explicit inclusion of time, which many other techniques ignore (see also Section 4.6).

[4] Fuzzy classification is a problem class on its own. The membership is defined by degree of its elements, i.e. a member can be part of different classes specified by a certain distribution.

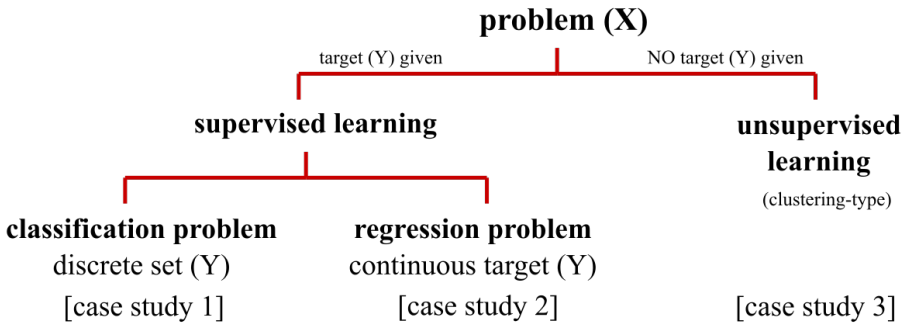

Figure 2: Taxonomy of machine learning and corresponding problem types. The main difference consists between supervised and unsupervised learning. The former fits a model to target data given input feature, while the latter is generally concerned with data clustering.

2.5 Data and feature selection

2.5.1 Data characteristics

The performance of a learning algorithm can only be as good as its inputs; making careful selection of features and data cleaning crucial. We refer synonymously to the raw data and used features as the data or inputs. The data can be characterised according to the dimensions m and n and their proportion to each other. One speaks of a *tall/long*, *fat/wide* or just a *big* data set if $1 \ll m \gg n$, $1 \ll n \gg m$ or $m \approx n \gg 1$, respectively. The product $m \times n$ may be referred to as the *data volume*.

The definition of the term *Big Data* is ambiguous. We give a couple of possible definitions while others can be thought of depending on the context. Coming from an econometrics point of view, one can ask what properties of an estimator are still valid for different limits of the ratio n/m. For example, maximal likelihood estimators in the context of exponential distributions are asymptotically normal when $n^2/m \to 0$ as $m, n \to \infty$ (Portnoy (1988)). So we may say that we deal with big data when $n > \sqrt{m}$. One the other hand, large datasets may contain distributions with fat tails, show heteroscedasticity or non-stationarity in the context of time series. Such properties can make the estimation of covariance matrices challenging (Dendramis et al. (2017)).

Definition based on the above are model dependent and are mostly not well defined in the context of machine learning approaches. Here, practice-oriented definitions can be useful. A possible definition of big data in line with many machine learning applications is large-volume data with a low information density. That means that potential richnesses can be gained from big data sources, but only given a large number of observations and proper filtering and cleaning (Daas et al. (2015)).

More generally, big datasets are often said to have the properties of high *velocity, variety, volume, veracity and value*, referring to the high frequency, lack of structure, large size, heterogeneous quality and commercial value of the data, respectively. All of these properties have to be investigated and taken into account when building a learning system. An example for regulatory data, which ticks all of these boxes and which have become available in the wake of the latest financial crisis are single-transaction derivatives market data under the European Market Infrastructure Regulation (EMIR) framework (European Commission (2012)). EMIR led to the daily submission of tens of millions of reports on activities in derivative markets which offer completely new opportunities to investigate financial markets, but also new challenges.

Major concerns, which may be aggravated by *many-V* data sources are *privacy* and *selection bias*. Privacy concerns are justified on the grounds that, even when using anonymised micro-data, the cross-section of several dimensions may allow one to easily identify an individual person or firm (Mayer-Schönberger and Cukier (2013)). Selection bias often relates to the way data are collected and may seriously affect the representativeness of results and the conclusions drawn from them (Angrist and Pischke (2008); Crawford (2013)). For instance, the online market for a product may not represent the average seller-buyer relation of that product in the overall market.

Data source type	Examples
Online market places	Real-estate, retail, P2P lending (Atz and Bholat (2016))
Social media	Online social networks (Bakshy et al. (2012); Anstead and O'Loughlin (2015))
Communication data	Phone, email or messenger (meta) data (Blumenstock et al. (2015))
Usage data	Electricity (Harding and Lamarche (2016))
News data	News streams or aggregates (Tetlock (2007))
High-frequency data	Market tick data, payment systems data (Kirilenko et al. (ming))
Scanner data	Micro retail data (Harding and Lovenheim (2014))
Regulatory data	EMIR (Cielinska et al. (2017)), Solvency II (European Commission (2014))
mobile/app data	Trends, demographics, technological changes (Mbiti and Weil (2011))
Open government data	Regional economic data (The Economist (2015))
Administrative data	Demographics, tax returns (ADRN (2017))
Retail banking	Financial statements, loans, mortgages (Bracke and Tenreyo (2016))
Logistics data	Freight, shipment, deliveries (Chan et al. (2016))
Infrastructure data	Traffic, travel and transport (Daas et al. (2015))
Environmental data	Granular geo-imaging (Njuguna and McSharry (2017))
"Internet of things"	Smart meters, wearable devices, etc. (Albert and Rajagopal (2013))

Table 1: Non-exhaustive list of relatively recently available data sources with potential to be exploited in the context of big data and machine learning.

2.5.2 Feature selection

Economic rationale is often used to select the features of a model. As the number of possible variables increases even in the face of a tall dataset, pure rationalisation gets harder and model uncertainty increases.

Principled approaches which partially avoid these problems and address the problem of model selection are PCA, LASSO, LARS (Friedman et al. (2009); Abadie and Kasy (2017)), BMS/A[5] (Steel (2011)) and double selection (Belloni et al. (2012)). These methods are part of the machine learning and econometrics toolboxes and will not be discussed further. However, shrinkage methods, such as LASSO or Ridge regression can be used for model calibration and will be addressed in Section 4.

A data-driven approach also provides information about relations among variables and the cross-correlation properties of features. That is, having n possible features (including the target variable), one considers the $(n \times n)$ *cross-correlation matrix*, e.g. using the Pearson product correlation coefficient, or the $(n \times n)$ *scatter-plot matrix* to measure the strength of linear cross-relations and the existence of specific patterns between pairs of variables, respectively. This can assist in the selection of features, and also in the choice of a promising learning algorithm. For instance, higher-order polynomial features and algorithms better suited for the investigation of non-linearities should be considered when scatter plot shows a complex structure. Yet another method is to test the statistical significance of features and remove them if they turn out less relevant than random probes (Kursa and Rudnicki (2010)). More generally, *ex-post* feature selection can be done if inputs turn out to be irrelevant. Efficient feature selection can also save computational and memory resources, in addition to improving model performance.

2.5.3 Feature standardisation

Input preprocessing is an important aspect which may crucially affect a learning method's performance. Several techniques, especially those combining inputs, like artificial neural networks (see Section 3.1.4), are sensitive to differences in numerical ranges between different features. A standard way to address this is *feature standardisation*, bringing all features to comparable numerical ranges or levels of variability. This can be achieved by calculating *z-scores*, i.e. the subtraction and division of each observation by the mean and standard deviation of that feature, respectively. More elaborate ways to achieve approximate standard normality involve transformations such as taking the logarithm or a Box-Cox transformation (Joseph and Chen (2014)),

[5] Principal component analysis, least absolute shrinkage and selection operator, least angle regression and Bayesian model selection/averaging, respectively.

which are also part of the standard repertoire of econometricians.

Another approach to address non-linearities and heterogeneities in data is binning (Finlay (2014)). Variables are taken from a continuous spectrum of values, which may span many orders of magnitude, to categorical variables while the bin width does not need to be uniform. This transformation is of particular interest if different variable ranges have different relations to other variables or the output, such as outliers or extreme values.

3 Machine learning models

We review a set of popular models within the machine learning toolbox. We focus on a concise and intuitive presentation accompanied by relevant examples. Some standard techniques from the statistical and econometrics approaches, such as logistic regression, are either part of the this toolbox or form the basis of mode sophisticated models. Machine learning models can be parametric or non-parametric. That is the input data either only determine the values of a fixed set of parameters or they influence the model's "shape", e.g. the number of parameters. Note that there is no single best method for a given problem in most cases. Different model types are often likely to perform equally well, which is called the *flat maximum effect* (Finlay (2014)). Rather, aspects of a specific approach may be more amenable in a given situation. A comparison of different model types is presented in Section 5, where we present a series of case studies.Theoretical underpinnings and advanced concepts can be found in Friedman et al. (2009); Abu-Mostafa et al. (2012); Goodfellow et al. (2016).

3.1 Supervised learning

We introduce a set of commonly used machine learning techniques for supervised learning where the target Y is given. Namely, *naïve Bayes, k-nearest neighbours, decision trees and random forests, artificial neural networks* and *support vector machines*. The defining characteristic of these approaches is that they take the target variable Y as an input to the cost function. Given a hypothesis $h(X)$ representing the model, a commonly-used objective function is the mean squared error (MSE). Problem (2) takes the general form,

$$ERR(X,Y,\beta) \overset{\text{MSE}}{=} \frac{1}{m}\sum_{i=1}^{m} e_i^2 \equiv \frac{1}{m}\sum_{i=1}^{m}\Big(h(x_i,\beta)-y_i\Big)^2 \quad \xrightarrow[\text{algorithm}]{\text{optimisation}} \quad \beta\,, \tag{4}$$

where we dropped the regularisation parameter λ and the error term[6]. The response variable Y is of dimension $(m \times 1)$, i.e. for each input there is one output observation or label[7]. Any machine learning approach consists of two parts, the hypothesis, e.g. an artificial neural network, which needs to be formalised in $h(\cdot)$ and an optimisation algorithm to minimise (4), and which is suited for the hypothesis.

A central question when designing a machine learning system is what learning hypothesis to choose and when learning is feasible. By the *feasibility* of learning, we mean that our model is capable to perform well on unseen data. Note that other factors, such as the input data and the selected features, will play crucial roles too. Generally, the availability of a larger amount of data allows one to consider more complex models, for instance artificial neural networks rather than a logistic regression. For supervised learning, a *rule of thumb* is that one should have at least ten times the number of observations than the number of (effective) parameters in one's model (Abu-Mostafa et al. (2012))[8]. For example, considering a linear regression model with three covariates and an intercept, means that we should start with at least 40 observations.

A basic optimisation algorithm which demonstrates the working and trade-offs of more sophisticated methods is *gradient descent*. Consider the familiar problem of a linear regression by minimising the squared sum of residuals. Eq. 4 takes the form

$$h(X) \,=\, X \cdot \beta \quad \rightarrow \quad ERR(X, Y, \beta) \,=\, \frac{1}{m} \sum_{i=1}^{m} \Big(x_i \cdot \beta - y_i \Big)^2 . \tag{5}$$

An intercept can be included via a column of ones in X. The summation on the right hand side runs over all training examples. Error function (5) states a convex minimisation problem. Under the conditions of the *Gauss-Markov theorem*, the best linear unbiased estimator has the closed-form solution

$$\beta^\star \,=\, (X^T X)^{-1} X^T Y. \tag{6}$$

However, the general dataset X may not fulfill the assumptions of the Gauss-Markov theorem (Hayashi (2011)), which forms the starting point for many developments in econometrics more widely. With an eye on more complex hypotheses, this problem can also be solved iteratively, i.e. using an algorithm. Indeed, a defining characteristics of *optimisation algorithms* is the explicit

[6]The properties of the error term, such as the normality of residuals, can tell us much about the quality of our specification. Analysis of residuals can or should be performed in a machine learning setup, but will not be pursued further here.

[7]Problems may involve multiple dependent outputs, i.e. $Y \in \mathbb{R}^{m \times r}$, $r > 1$. In this case, parts of the learning method may have to be adapted. As this generally does not require fundamental chances to the approach, we will stick to the case $r = 1$ in this paper.

[8]this is the so-called VC-dimension (named after its originators Vladimir Vapnik and Alexey Chervonenkis), which measures the largest set of points a model can *shatter* (differentiate), which generally increases with complexity.

design around iterative steps leading to a good solution while convergence is achieved efficiently. Most algorithms are built to accommodate a trade-off between computational cost and accuracy, such that a satisfying solution to a problem can be found in an acceptable amount of time.

Gradient Decent The most efficient minimisation of the cost function (5) can be imagined as finding the quickest way to the bottom of a paraboloid embedded in an (n+1)-dimensional space when starting from some randomly chosen point $\hat{\beta}_0$. This path is given by the steepest gradient of the cost function defined by the vector of partial derivatives in the direction of each independent variable, $-\nabla_\beta ERR(X;\beta) \equiv -(\partial/\partial\beta_0, \ldots, \partial/\partial\beta_n)ERR(X;\beta)$. The optimal values of the β_i are approximated by performing iterative downhill steps in direction of the gradient,

$$\hat{\beta}^{k+1} = \hat{\beta}^k - \gamma\nabla_\beta ERR(X,Y,\hat{\beta}^k) \overset{\text{OLS}}{=} \hat{\beta}^k - \gamma\frac{2}{m}\sum_{i=1}^{m}\Big(x_i\hat{\beta}^k - y_i\Big)x_i \xrightarrow{k\to\infty} \beta^\star, \qquad (7)$$

The parameter γ is called the *learning rate* and determines the speed of convergence to $\beta^\star$. Because the OLS problem (5) is convex, every iteration of (7) is guaranteed to bring us closer to the optimal solution $\beta^\star$, assuming that the learning rate γ is not chosen too large. Here we can see the emergence of the trade-offs between speed and precision which many numerical optimisation procedures face. Firstly, a larger value of the learning rate means faster convergence. Too large a value, however, can prevent the algorithm from converging, because $\hat{\beta}$ may start oscillating around its optimal value due to repeated "overshooting" . This is illustrated in Figure 3. Choosing too small a value of γ will slow the algorithm down unnecessarily and make it inefficient. Inefficiency can be a serious problem when faced with complex problems. Second, assuming that the learning rate is chosen appropriately, the time to convergence in terms of the number of algorithm iterations it takes to reach optimal parameter values (vertical dahsed line in Figure 3) will depend on the desired level of precision. This can be set set according to some stopping criteria comparing the relative change in $\hat{\beta}$, when going from step k to $k+1$[9]. For instance, we could tell the algorithm to stop iterating as soon as the ratio $|(\hat{\beta}^{k+1} - \hat{\beta}^k)/\hat{\beta}^k| < 10^{-6}$.
While we want to stress that it is necessary to understand the working of a chosen algorithm, one often does not need to be concerned with its detailed implementation. There exists an array of dedicated software packages in commonly used programming languages which readily provide a set of optimisation algorithms for different machine learning techniques (see e.g. Pedregosa (2011)).

[9]More sophisticated algorithms often use or approximate the second derivative of the objective function (Hessian matrix) which allows for controlling the learning rate or to locally analyse the topology around a given point x (Solomon (2015)).

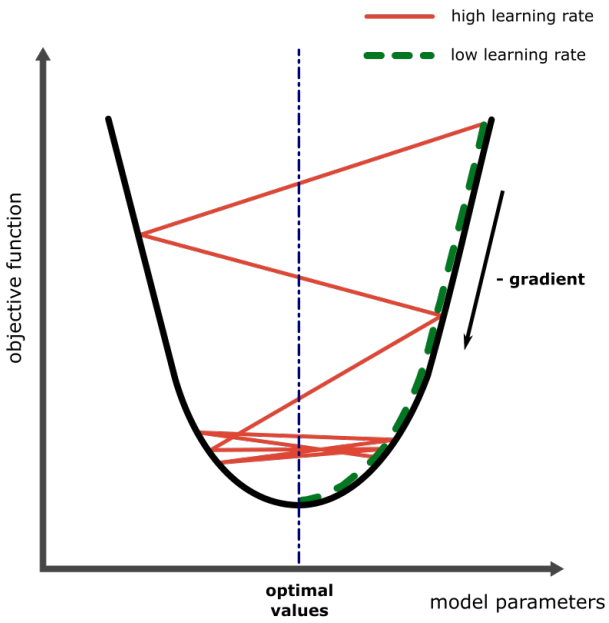

Figure 3: Schematic representation of the implications of large and small learning rates on the behaviour of gradient-based optimisation algorithms. Large learning rates (red line) initially converge fast but may lead to "over-shooting" and non-convergent oscillations about the optimal values of model parameters. Setting a small learning rate (green dotted line) promises more accuracy in the final result, but may be slow to converge, i.e. inefficient.

3.1.1 Naïve Bayes classifier

Naïve Bayes is a simple, often surprisingly powerful, method for solving classification problems. One of its most appealing features is its low computation cost. It also can be taken as a *baseline* to evaluate more complex methods. The naïve Bayes classifier is based on applying the Bayes' rule for conditional probabilities under the simplistic (naïve) assumption that all features are independent from each other. Assuming C classes $1, \ldots, C$, the classifier for an observation x_i can be defined by the *maximising-a-posteriori* (MAP) decision rule,

$$\begin{aligned} x_i \mapsto c_i : \quad & \underset{c}{\text{argmax}}\, P(c|x_i) = \underset{c}{\text{argmax}} \left(\frac{P(x_i|c)\, P(c)}{P(x_i)} \right) \\ \propto \quad & \underset{c}{\text{argmax}}\, P(c) \prod_{j=1}^{n} P(x_{i;j}|c)\,, \end{aligned} \tag{8}$$

where the product runs over all n features and $P(c_i)$ is constant. The class-conditional likelihood factors $P(x_{i;j}|c)$ for x_i bing labeled c according to feature j are calculated assuming a certain probability distribution of the data, such as Gaussian, binomial or multinomial. The choice depends on either having a continuous, Boolean or categorical variable, respectively. The final result is given by the product of probabilities (independence assumption) and the class which maximises it. Note that a single factor $P(x_{i;j}|c)$ may be zero for categorical variables, setting the

whole expression to zero. In this case, one can use smoothing techniques to guarantee a finite output. For instance, *Laplace add-1 smoothing* adds one to the numerator of relative frequencies of occurrences (Bholat et al. (2015)).

A hypothetical example which highlights potential issues arising from the feature independence assumption is the following. Assume we are interested in modelling if individuals from a certain tribe, say economists, are unemployed or not (class label Y) given their educational attainments and current income (feature matrix $X \in \mathbb{R}^{(m\times 2)}$ for m individuals). Next we assume that the probability of an individual being employed can be modelled by a bi-variate normal distribution $p(x_i) \propto \exp(-\frac{1}{2}(x_i-\mu)\Sigma^{-1}(x_i-\mu)^T)$, where x_i is the length-2 feature vector for individual i. μ is the vectors of population or sample means and Σ is a 2-by-2 covariance matrix containing the variances of both features and a cross-correlation term proportional to the feature correlation ρ_{12}. Setting $\rho_{12} = 0$, the naïve Bayes classifier for an individual would assign the class label (employment status) which maximises the product of two univariate Gaussian distributions determined by the class means μ^c and variances σ^c, respectively. The degree of "naïvity" of this model is now based on how much the actual value of ρ_{12} deviate from zero.

3.1.2 $k-$Nearest neighbours

k-Nearest neighbors (or k-NN) is a non-parametric method for both classification and regression problems. The underlying idea is simple. An observation is modelled as its k nearest other observations in the feature space. For a classification problem, an observation is assigned to the majority class of its nearest observations. For a regression problem, an observation is assigned to the mean value of its nearest neighbours, where weights may be applied. This is formalised as follows. The output value for a single observation x_i is obtained as follows:

1. Neighbour selection: Calculate the distance of x_i to all other points in the feature space and determine its k closest neighbours $\{x_j\}_i^k$. Euclidean distance is commonly used as a distance measure, but alternatives such as the L-1 norm or cosine distance may be considered (see Section 3.2.2).

2. Value assignment:

   - k-NN classification: Assign the output y_i the class membership (i.e., $y_i \in \{C_1, C_2, \ldots, C_c\}$ $\forall\, i = 1, 2, \ldots, n$ where c is number of class levels) by the majority vote of its k nearest neighbors. If $k = 1$, then the y_i is simply assigned to the class of its single nearest neighbor.
   - k-NN regression, y_i is assigned the average value of its k nearest neighbours, $y_i =$

$\frac{1}{k}\sum_{j=1;x_i\in\{x_j\}_i^k}^{k} x_j$. One could also consider distance-weighted averages.

The performance of k-NN can be evaluated by looking at the error rate of miss-classified examples for classification or squared errors for regression problems, respectively. For k-NN to be robust, one should work with features of comparable numerical ranges (feature standardisation) and keep the number of features limited (curse of dimensionality). When considering irrelevant features, the feature space may become sparse and a clustering structure between similar observations, whose proximity one seeks to exploit, may get lost. Intuitively, the patterns or signals one would like to pick up may get diluted by irrelevant information. The number of nearest neighbours k is a hyper-parameter of the model and can be chosen using cross-validation (see Section 4).

3.1.3 Tree models and random forests

Tree models are a popular non-parametric machine learning technique for regression and classification problems. They can handle complex relations within data in an accessible and interpretable way. The idea is to consecutively divide (branch) the training dataset based on the input features until a assignment criterion with respect to the target variable into a "data bucket" (leaf) is reached. The procedure works as follows for a decision tree[10]. The aim is to minimise the entropy $\mathcal{H}(Y|X)$ (objective function) within areas of the target space (buckets) conditioned on the features X. Starting with the full set X of m observations, one identifies the feature x which leads to the highest *information gain* ($\mathcal{I}$) when using it as a split criterion.

$$
\begin{aligned}
&\mathcal{I}(Y|x) = \mathcal{H}(Y|X) - \textstyle\sum_{v\in\{x\}} \frac{|X_v|}{|m|}\,\mathcal{H}(Y|X_v) && \text{information gain (classification)} \\
&\mathcal{H}(Y|X) = -\textstyle\sum_{c=1}^{C} p(Y=c|X)\log\big(p(Y=c|X)\big) && \text{entropy (classification)} \\
&\mathcal{H}(Y|X) = \tfrac{1}{m}\textstyle\sum_{j=1}^{k}\sum_{i=1}^{m_j}(y_i - \mu_j|_{x_i\in X_j})^2 && \text{MSE (regression)}
\end{aligned}
\tag{9}
$$

where $p(Y=c|X)$ is the relative frequency of class c observations in X. The first sum runs over all distinct values $\{x\}$ of feature x_i. $|X_v|$ is the number of observations which take on each value. Note that the sum can only be applied for features which are categorical. For a feature which takes continuous values, one performs a binary split (two resulting branches). The split point is determined by ordering the *finite* set of observations $\{x\}$ and selecting the value where the information gain is maximal. In a regression setting, the entropy can be replaced by the

[10] This is a flavour of the C4.5 algorithm for the construction of decision trees (Kotsiantis (2007)).

mean squared error (MSE) and splits are performed along the dimensions which most reduce the error (last row of Eq. 9). The outer sum goes over all k data areas $\{Y_j, X_j\}$ of size m_j. The inner sum runs over squared differences between the target y_i and their mean value μ_j for all observations in area j.

A schematic representation of a tree model with two features, x_1 and x_2, is given in Figure 4. On the upper most level L_1, feature x_1 provides the best split at x_1'. We have two branches on the next level L_2, where the best split points are x_2' and x_2''. This results in dividing the input data into four parts $A_1 - A_4$. The final class or value assignment is based on the target values Y in these areas. The model consists of the sequence of splitting rules and the final value assignment. The general, and potentially complex, assignment function $Y(A_i)$ (left) can be visualised as the tree (right). This correspondence helps to interpret a potentially deep tree model with multiple layers and a large number of branches. However complex, most fit relations between RHS and LHS can be explained via simple *if-else statements*. For instance, the tree model's output can be described by statements like, 'if x_1 smaller than x_1' and x_2 bigger than x_2', then Y is given by the average over A_1. Alternatively, each full branch of the tree can be seen as an interaction of dummy variables, i.e. with a unit value over their respective support, and the value of the target variable as the coefficient of that term (Mullainathan and Spiess (2017)). Note, however, that the size of this coefficient does not say anything about its relative importance compared to other branches. This may be better judged by the number of observations it applies to or the covered area of the respective region it represents as shown in the left-hand side of Figure 4.

The intuition behind the working of classification trees is as follows. The entropy $\mathcal{H}(X)$ is associated with the amount of disorder in the data X with respect to the C class labels, i.e. how much they are mixed up. Each branching of X reduces this disorder. At each step, we choose the feature with the highest reduction in disorder, i.e. which provides the most order in sorting the C classes. The full decision tree is grown by repeating this procedure on all subgroups, where each step k results in a new layer L_k of branches subdividing our dataset further. Finally, one decides on when to stop growing branches, resulting in final class assignments (leaf nodes). There are some popular options for stopping. One is to grow the tree until a leaf node is *pure*. Another is to define a stopping rule related to the size or decomposition of leaf nodes. Examples for stopping rules can be a minimal size $|X^k|$, like five, or a minimal entropy in y^k, i.e. a minimal "class order". A general classification rule is to assign all observations in X^k to the majority class.

The above procedure is similar for a continuous target Y (regression tree). Two main differences are, the error function usually takes the form of a squared error (last line in Eq. 9) and the final

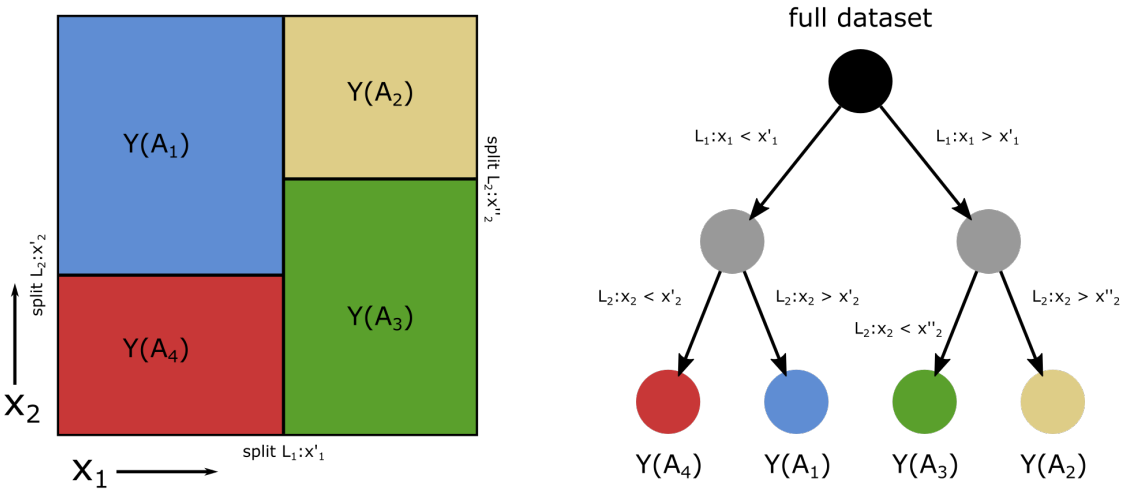

Figure 4: Schematic representation of a tree model with two features x_1 and x_2. Left: The target variable space is systematically segregated by the tree model based on the input features. Right: Tree representation of the final model. The tree is grown from the top to the bottom. Coloured leaf nodes at the bottom correspond to coloured areas in the LHS panel.

value assignment is usually calculated as a (weighted) average of the leaf node values μ_k.

One problem with tree models is over-fitting, particularly if large trees are grown. This can be understood as, longer the branches become, more detailed patterns in the data they can take into account. However, the optimum level of complexity, i.e. where the test error is minimised, will be surpassed at some point as some of those detailed patterns in the training set data can be just noise. Growing large trees may also lead to a violation of the rule of thumb that one should have at least ten times the number of observations than parameters in one's model. Ways to address this issue are *pruning* of trees (see Section 4.4.2) or limiting the maximal size a tree model is allowed to reach (early stopping rule). The later can be achieved, for example, by only allowing a maximal tree depth (number of splits along a branch).

Yet another way to overcome some of the problems of standard tree models are *random forests* which we discuss next.

Random forests Growing decision trees in the above form may lead to severe over-fitting. Over-fitting means that the variance σ^2, and subsequently the error, of a model for out-of-sample testing is high (see Section 4). Random decision/regression forests are a powerful extension of the above tree model which addresses this with relatively little extra work. The underlying idea is to grow a set of roughly independent tree models, which jointly perform better than a single tree model. Classification decisions and regression values are then given by the majority vote or average prediction of all single trees, respectively. An algorithm for growing a random forest of

F trees is (Friedman et al. (2009)),

1. Randomly select a sample X' of size $m' < m$ from X, with m' about $70\% - 90\%$ of m.

2. Grow a decision tree using X' as above with the difference that the split feature at each branching is chosen from a random sample S_r of $n' < n$ features. The feature set S_r is chosen anew at each split point. A standard value is $n' \approx \sqrt{n}$.

The fact that a random decision forest leads to improvement compared to a single tree, can be seen as follows. The trees in the forest may not be completely independent from each other, as they are derived from the same underlying data. Assuming a positive pairwise correlation of $\rho \ll 1$ between each pair of trees (Bernard et al. (2010)), the variance of the forest average is expected to decline according to (Friedman et al. (2009)),

$$\sigma_F^2 \propto \rho\sigma^2 + \frac{1-\rho}{F}\sigma^2 . \tag{10}$$

The second term can be neglected as the number of trees F may be very large, leaving a reduction in variance proportional to the remaining correlation between trees. The number of trees F to be grown can be decided during runtime. Constructing each tree, only the sample X' is used. The remainder of the data can be used to perform out-of-sample tests on trees grown before and unused observations. The average of this *out-of-bag error* will first decrease with the number of trees and then level off. One may stop growing trees, as no further gains are expected for unseen data at this saturation point.

A general drawback of random forests, as compared to single trees, is that they are hard to interpret due to the built-in randomness. A way of interpreting random forests and related methods based on *ensembles* is to see them as a manifestation of *crowd intelligence* (Galton (1907)). Many interfering random decisions lead to a coherent picture. Even so, it may be possible to extract economic rationale in the form of a set of probabilistic rules from a random forest. For instance, some paths down the tree and certain branch splittings will be more abundant than others, as they lead to a larger aggregated reduction in the cost function. Consequently, certain variables will play a more important role to fit the target than others. In this way, decision trees and random forests can be used to make reliable predictions and generate insight.

3.1.4 Artificial neural networks

Artificial neural networks (ANN) are one of the most powerful statistical learning algorithms. Originally designed to simulate the functionality of the human brain, neural networks have since

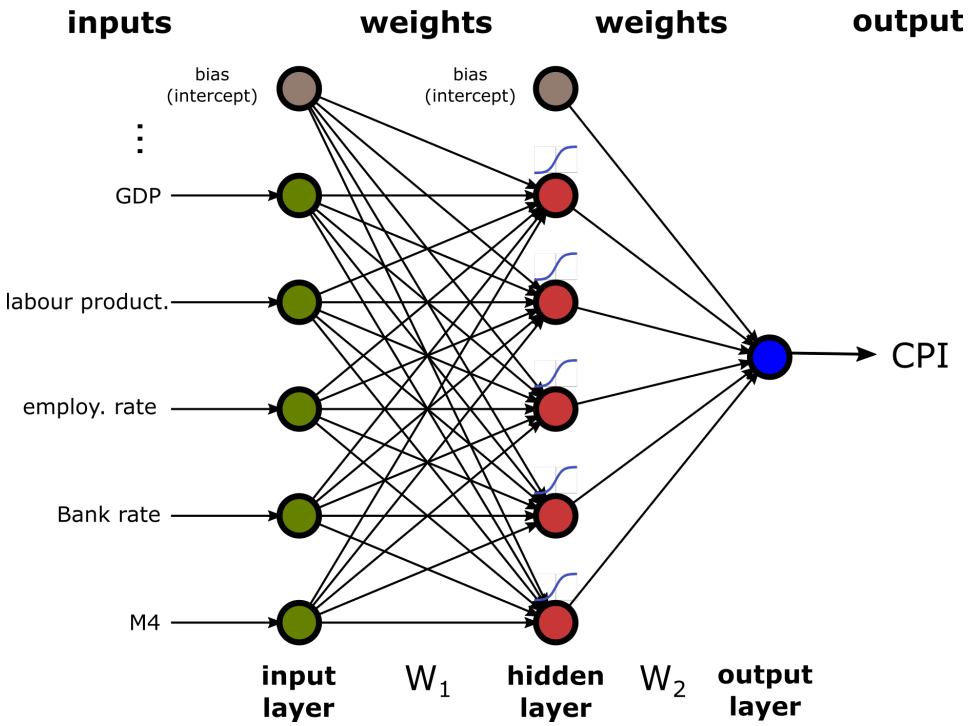

Figure 5: Graphical representation of a feed-forward artificial neural network (FFANN) within the inflation forecasting framework from case study 2 (Section 5.2; not all features are shown). The inputs (LHS; green) are processed in a step-wise fashion, moving through the hidden layer(s) (red) till reaching the output layer (blue) on the RHS. At each node, a combined signal is generated from the input features from the previous layer and a (non-linear) transformation through the activation function. The model parameters constitute the links in the network and are contained in the weight matrices W_1 and W_2. Intercepts are represented via the "bias nodes" (gray).

become a widely used tool in machine learning for classification and regression problems. ANN are also at the forefront of current developments in artificial intelligence (Silver (2016); The Economist (2016)). We focus our discussion on the widely used class of feed-forward networks (FFANN), called *multi-layer perceptrons*. These are versatile models because of their ability to approximate very general continuous functions (Cybenko (1989)). More complex structures include, for example, recurrent neural networks (Lipton et al. (2015); Sibanda and Pretorius (2012)). Neural networks can be described as semi-parametric models. On the one hand, a model's parameters are given through the structure of the underlying network. On the other hand, the input data decide which of the model parameters actually matter. This is even more true when the network becomes bigger and more complex.

The network representation of a FFANN taking macroeconomic variables as inputs (LHS) to model CPI inflation is shown in Figure 5 (see Section 5.2). There is an input layer with a node for every feature on the left-hand side. The coefficient matrix W_1 (weight matrix) connects the inputs to the *hidden layer* in the middle of the network. At each node in the hidden layer, the

joint signal from the input layer is evaluated by an *activation function* to generate a standardised output as the next layer's input, hence a feed forward signal. These internal inputs are called *derived features*. The weight matrix W_2 connects the derived features to the output layer. The coefficients of the node activation functions are the elements of the weight matrices W. That is, for the i^{th} node of the k^{th} layer, the input to the activation function is the inner product of the output of the $(k-1)^{th}$ layer and the i^{th} row of coefficients in the weights matrix. A commonly used activation function, which allows for the modelling of non-linear relations, is the logistic or sigmoid function. When we need to develop a classification model of more than two classes, we can use the softmax function (Friedman et al. (2009)). The network model in Figure 5 is formalised as follows.

$$\begin{aligned} LG(X \cdot W) &\equiv \tfrac{1}{1+e^{-X \cdot W}} &&\in (0,\,1)\,, \\ Y &= LG\big(X \cdot W_1^T\big) \cdot W_2^T &&\equiv h(X, \beta)\,, \end{aligned} \tag{11}$$

where $X \cdot W_1^T$ denotes the inner product of the features X and the transpose of the weight matrix W_1. As for logistic regressions, it is possible to include an intercept at every layer. This is done by adding a unit column to X and all intermediate outputs, as well as to adjust the dimensions of the weight matrices W. Graphically, one adds *bias nodes* with a unit input to all layers except the output layer. Having a total of n features, the inputs, weight matrices and output, $X\,, W_1^T\,, W_2^T$, Y in Eq. 11 are of dimensions $(m \times (\text{n+1}))\,, ((\text{n+1}) \times (\text{n+1}))\,, ((\text{n+1}) \times 1)$ and $(m \times 1)$, respectively. The "+1", represents the columns of ones and the weights connecting the bias nodes (intercepts) in Figure 5, respectively. Note that some entries in the weight matrices W are set to zeros as there are no forward connections into the bias nodes. Thus, assuming 11 features like in case 2, the final model in (11) has a total of $(12 \times 11) + 12 = 12^2 = 144$ parameters. This already suggests that one needs a substantial amount of input data compared to previously introduced techniques given only a moderate number of nodes and layers.

The error function for a FFANN for a two-class classification can be written in the form of a logistic regression,

$$ERR(X, Y, \beta) = -\frac{1}{m} \sum_{i=1}^{m} \Big(y_i \log\big(h(x_i, \beta)\big) + (1 - y_i) \log\big(1 - h(x_i, \beta)\big) \Big)\,. \tag{12}$$

The sum goes over all observations. Note that one of the two terms in (12) always vanishes for a $\{0, 1\}-$classification problem. Alternative choices for the neuron activation function are the hyperbolic tangent, step function, linear, rectified linear unit function[11], or even bell-shaped

[11] ReLU: $f(x) = max(0, x)$.

curves like a Gaussian.

(12) generally poses a non-convex problem in contrast to the standard logistic regression. This is related to the elaborate form the hypothesis $h(x, \beta)$ in Eq. (11) can take. This means that an iterative algorithm based on local optimisation is not guaranteed to converge to the global minimum. One way to get around this is to repeatedly solve equation (12) numerically for different random initialisations[12] for the weight matrices W and the values uniformly chosen from the middle part, say interval $[-0.7, 0.7]$ (Friedman et al. (2009)). The intuition behind this is to start with an approximately linear model, as the sigmoid function is approximately linear around the origin, and take non-linearities into account with increasing absolute values of the entries of the weight matrices W. Thus, complex non-linear structures will be "learned" by the model as required. This procedure also provides a *density forecast* for the target variable Y, where one can either pick the best one or take the average as a point forecast. Note that one should not take the average value for the entries of W due to the non-linearity of the logistic function. As random initialisation treats all inputs as equals, it is important that all features are standardised in some way, e.g. by taking their z-scores (see Section 2.5). One possibility to minimize (12) approximatively would be through gradient descent, where the way to calculate the derivative with respect to the parameters W_{ij} is called *back propagation*. Back propagation can be computationally slow, therefore faster methods, like *conjugate gradients*, may be preferred. Dedicated software packages usually provide several efficient implementations.

A number of comments regarding the design and implementation of ANN are in place. The number of free parameters is determined by the number of hidden layers (where networks with more than one hidden layer are called *deep*, hence *deep learning* (LeCun et al. (2015); Goodfellow et al. (2016)), and the number of nodes in each layer. Although a larger number of free parameters is expected to lead to better performance, it may also lead to over-fitting. This is a problem with neural networks, especially when the given dataset is not sufficiently large. Hence, a form of regularisation may be needed. Additionally, larger networks are computationally more demanding than smaller ones, which may pose constraints. It is recommended to start with a smaller network and enlarge it if it leads to a better model. This can also be addressed with cross-validation (see Section 4). The optimal design will depend on the dataset, the question at hand and the experience of the modeller.

A potential problem with ANN (as with other advanced machine learning techniques) is their potential lack of their interpretability. That is, the economic intuition of the coefficients in the weight matrices W is not fully clear. Furthermore, these are dependent on the topology of the

[12]Note that a random initialisation is required in ANN to obtain non-trivial solutions.

network model, where different topologies will have a different number of weights given the same inputs. One way to imagine the working of neural networks is to think of them as providing complex truth tables for the representation of almost arbitrary complex patterns within the data. A "neuron" (node) in the network is firing, i.e. giving a *true* output, if the combination of inputs and their weights W are strong enough. Also, looking at the weight structure can provide an intuition of important features and their interactions.

Artificial neural networks are especially suited for situations with a large number of possible predictors (features), potentially complex relations among them, and a large number of observations. These conditions are likely to be given when faced with micro-based data, such as single-contract or single transaction data for a particular asset or agent, which are more and more available to researchers. An illustrative example is given by Giesecke et al. (2016), where the dynamic state of mortgages in the US is investigated based on a large set of predictors from about 100 million observations between 1995 and 2014. Given that mortgage contracts constitute a major part of banks balance sheets, understanding and predicting their evolution is crucial for financial institutions and regulators alike. Neural networks have also been applied to historically relevant topics, such as financial distress propagation and firm failures (Tam and Kiang (1992); Altman et al. (1994); Coats and Fant (1993); Lacher et al. (1995); Salchenberger et al. (1992)) or credit scoring (Blanco et al. (2013); Khashman (2011)).

An intuitive explanation why, particularly deep, neural networks are so powerful in describing complex relationships is their design around simple functional approximation and the exploitation of hierarchy (Mehta and Schwab (2014); Lin et al. (2016); Schwab and Mehta (2016)). That is, the network uses simple local approximations, like lower order polynomials, and combines these at different layers, describing the system under study at different length scales. Thus, a complex system is described as a combination of simple ones. The final result is more than the sum of its parts. An example is image or face recognition, where various objects or features thereof, e.g. eyes, are identified first and recombined to take a final decision of what or who is shown.

3.1.5 Support vector machines

Support vector machines (SVM) are a powerful technique for both classification and regression type problems, preferably applied to classification. Traditionally, two-class classification problems are modelled by logistic regressions. Inputs are evaluated according to their position to hyperplane in the feature space and projected to a (0,1) interval which can be interpreted as probabilities of class membership.

SVM refine this picture and address two potential issues arising with this approach. Firstly, data for classification may not be linearly separable, i.e. by a straight line or hyperplane. Secondly, the exact position of the separation boundary is not fixed, even if one knows the class membership of all observations. To solve the problem of non-linear separability, the data can be projected into another, eventually higher dimensional feature space. Here, the data can become linearly separable if the transformation is chosen appropriately. This projection is a major component of the SVM algorithm. The second part is how to choose the best line. An SVM defines the best line simply as the one which has the maximum margin, i.e. vertical distance to its closest observations. This maximum margin classification has an additional benefit. Only the closest data points to the line have to be remembered in order to specify the model and classify all points. These data points are the so-called *support vectors*, hence the name support vector machine. Simplifying, one can say that SVM is basically a modification of logistic regression augmented by a feature transformation and geometry.

These ideas are depicted in Figure 6 for a two-class classification problem, which we concentrate our discussion on. The basic concept is analogous for multi-class classification (Friedman et al. (2009)). SVM regression will be briefly discussed at the end of this section. The left panel shows two types of observations (green and red dots) in some feature space. An SVM will try to find a decision boundary to separate these two classes by the *maximal margin*, represented by the yellow band. Finding the separating boundary (black line), which maximises the margin, is, however, not straightforward in the general case. Points may not be linearly separable in most cases. That is, we need to take our data to another space such that the green and red dots can be separated by a straight line, as shown in the right panel. Hence, the idea behind SVMs is now two-fold. First, one tries to find a representation of the feature space in which the data are linearly separable and, second, one identifies the points in the input space which define, or support, the maximal margin, the support vectors.

Assuming that we are in a feature space where our data are linearly separable (right-hand side of Figure 6), the separating boundary, decision rule and error function for two-class classification can be defined as

$$
\begin{aligned}
|x_{SV}^T \cdot \beta| &\equiv 1 && \text{decision boundary (support vectors)} \\
h(x_i, \beta) &= sign(x_i^T \cdot \beta) && \text{hypothesis} \\
ERR(X, Y, \beta) &= -\tfrac{1}{2m} \textstyle\sum_{i=1}^{m} \Big(|y_i - h(x_i, \beta)| \Big) && \text{error function}\,.
\end{aligned}
\tag{13}
$$

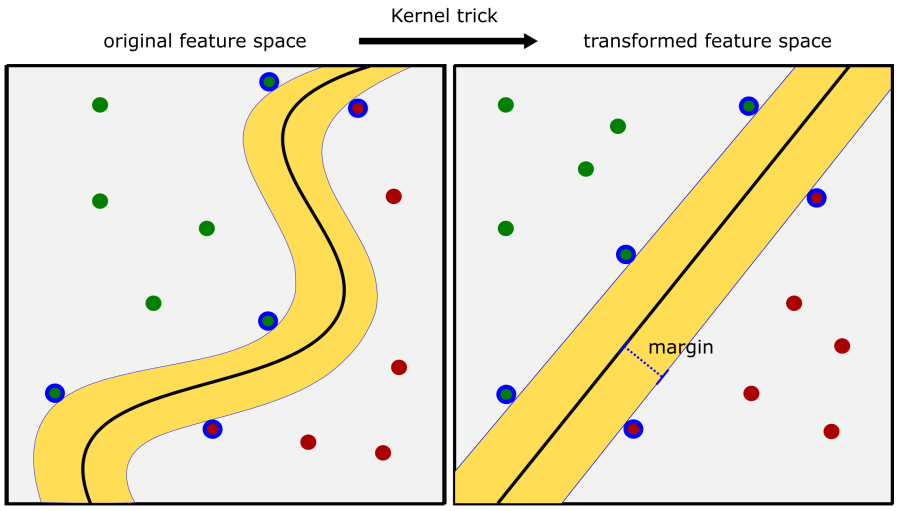

Figure 6: Schematic representation of a two-class (green and red dots) support-vector classifier in a two-dimensional feature space. Left: Features may not be easily classifiable (linearly separable) in the original feature space. Right: Using a kernel transformation, the SVM classifier is based on a maximal-margin boundary (yellow band) tangential to the support vectors (observations with blue edges).

The coefficients β define the hyperplane satisfying the decision boundary equation above (x_{SV} are the support vectors lying on the boundary of the yellow band in the figure Figure 6) and the target values take the values $y_i \in -1, 1$. Through geometrical considerations (Abu-Mostafa et al. (2012)), the coefficients β can be found by solving the (dual) optimisation problem:

$$\begin{aligned} \mathcal{L}(\alpha) \quad &= \textstyle\sum_{i=1}^{m} \alpha_i \, - \, \frac{1}{2} \, \alpha^T \cdot H \cdot \alpha \, , \\ \text{with} \qquad &\beta = \textstyle\sum_{i=1}^{m} \alpha_i y_i x_i \, , \quad H_{ij} \equiv y_i y_j \; x_i^T \cdot x_j \, , \end{aligned} \tag{14}$$

under the constraints $\alpha \cdot Y = 0$ and $\alpha_i \geq 0$. Considering the Lagrangian $\mathcal{L}(\alpha)$ (objective function), it looks as though there is a large number of free parameters α_i, namely one for each observation x_i. However, it turns out that only those α_i have non-zero values which correspond to a support vector as defined in Eq. 13 and indicated by the blue-edged points in Figure 6. As the support vectors are only the closest points to the decision boundary, the number of support vectors compared to the number of observations will be small in general. The above optimisation problem can be solved using quadratic programming. Note that this also means that one may run into difficulties when facing large datasets, because one has to handle a potentially non-sparse and large matrix H (high memory requirement).

The support vector classifier described so far finds linear boundaries in the input feature space. In the general case of non-linear separation, we can transform data by enlarging the feature

space using basis expansions. That means, we are converting data from lower dimension to higher dimension, such that the data become linearly separable. Finding the right transformation which separates the data sounds challenging. However, there is a mathematical trick to elegantly solve this problem. The input data X enters the above Lagrangian only via an inner product, resulting in a scalar. This allows one to define transformations $T(\cdot)$ and inner products via a so-called *kernel*[13],

$$\mathbb{K}(\hat{x}, \hat{x}') = \langle T(x), T(x') \rangle \,. \tag{15}$$

One of the most common choices for a kernel function is the *radial basis function* (Gaussian kernel)

$$\mathbb{K}(x, x') = exp(-\gamma ||x \, - \, x'||^2) \tag{16}$$

From the polynomial expansion of the exponential function, $exp(x) = \sum_{n=0}^{\infty} x^n/n!$, one can see that this kernel includes a general polynomial transformation of the feature space, in fact one of *infinite order*. Other commonly used kernel functions are linear or polynomial kernels, although a large number of functions can be used in practice.

Our description has focused on so-called *hard margin* classification so far, which does not allow any misclassification of training examples. This may not lead to the best generalisation properties of the final model, because some observations are likely to be genuine outliers and should not be bothered with when misclassified. The above optimisation (14) problem can be slightly modified to allow some leniency. By introducing an upper bound C for α_i, one also obtains non-margin support vectors for which $\alpha_i = C$, leading to a *soft margin classifier*. The optimal choice for the kernel or regularisation parameter C can again be set via cross-validation (see Section 4).

Support vector machines can also be used for regression problems targeting a continuous variable. The underlying principles are similar to those of the classification case (see e.g. Smola and Schölkopf (2004) for a detailed description). In this case, the model takes a similar form to a linear regression model with the difference that new inputs z are transformed by the (non-linear) kernel function (16),

$$Y(z) = \sum_{i=1}^{m} \alpha_i \mathbb{K}(x_i, z) + \beta_0. \tag{17}$$

The model parameters β_i are non-zero for the support vectors. Thus, Eq. (17) can be interpreted as an approximately linear regression within the kernel-transformed input space (RHS of Figure 6). Similar to most of the sophisticated techniques, the interpretability of the model

[13]Formally, a valid kernel needs to fulfill the Karush-Kuhn-Tucker (KKT) conditions within quadratic programming (Abu-Mostafa et al. (2012)).

and its outputs is an issue for SVM. Using a non-linear kernel, such as the Gaussian kernel, the support vectors and its components may not be interpreted directly. However, the feature space and decision boundaries can always be explored numerically, involving the position of support vectors and decision boundaries for a classification problem[14]. This may provide one with enough insight to interpret one's model.

3.2 Unsupervised learning

We now move on to commonly used unsupervised machine learning techniques. The difference to supervised methods is that there is no target variable Y. Most unsupervised algorithms aim to find a clustering or group structure in the data, which has to be interpreted by the researcher. This can have the advantage of leading to a less biased analysis, as unsupervised methods often rely on non-parametric approaches. The resulting patterns may reproduce some stylised facts, raise new questions or provide the inputs for subsequent steps in the analysis. For example, group labels could be taken as a feature or response variable in a supervised machine learning model.

3.2.1 $k-$Means clustering

This is probably the most commonly used clustering algorithm. Like all clustering methods, k-means tries to group observations close to each other in the feature space, indicating similarity. The similarities and dissimilarities between different such groups or clusters can provide insights about commonalities between observations which may previously have been unknown.
Technically, k-means is a centroid-based algorithm. The goal is to find the cluster assignment of all m observations to C clusters[15], which minimises the within cluster distances (often measured by Euclidean distance, other distance measures can be used; see next section) between each point x_i and its cluster centre μ_c. The corresponding cost function is

$$ERR(X, C) = \frac{1}{m} \sum_{c=1}^{C} \sum_{x_i \in C_c} ||x_i - \mu_c||^2 . \tag{18}$$

We normalised the sum of squares by the number of observations. This is needed to compare clusterings of different sized samples. Setting a fixed number of clusters C, a simple heuristic to find a clustering which approximately minimises equation (18), consists of alternating steps

[14] Individual feature weights can be extracted directly for linear kernels, where decision boundaries are hyperplanes in the feature space. In the non-linear case, input feature space segmentation and rules based on linear feature space combination may be applied (Navia-Vazquez and Parrado-Hernandez (2006)).

[15] Usually, k indicates the number of clusters in k-means clustering, but is reserved for iteration indexing here.

of cluster assignment and centroid shifting.

Choose C observations at random and assign them to be the initial cluster centroids $\mu_j, j \in \{1, \ldots, C\}$, in the n-dimensional feature space. When going from step k to $k+1$, repeat until no observation is re-assigned to another cluster:

1. **Cluster assignment:** Assign each observation x_i to its closest centroid C_i:

$$C_i^k = \{x_i \mapsto \mu_i^k | \; ||x_i - \mu_i^k|| \leq ||x_i - \mu_j^k||, \forall j \in \{1, \ldots, C\}\} \tag{19}$$

2. **Centroid shift:** Calculate the new position of each centroid C_c:

$$C_c^{k+1} = \frac{1}{|C_c^k|} \sum_{x_i \in C_c^k} x_i \, , \forall c \in \{1, \ldots, C\}, \tag{20}$$

   where $|C_c^k|$ stands for the number of observations assigned to cluster c. Generally the cluster centres do not coincide with observations after the first shift.

Features should be standardised (see Section 2.5) and highly-correlated inputs avoided. Non-standardised features can result in skewed distance measures across dimensions. Correlated features may lead to spurious clustering, as points will naturally cluster along those dimensions. As Eq. 18 poses a computationally hard problem, there is no guarantee that the above algorithm converges to the global minimum. It may get stuck in a local minimum, given some random initialisation of centroids. Therefore, the quality and robustness of a clustering can be assessed by running the above algorithm several times while comparing assignments and "goodness-of-fit" measures, such as average intra-cluster distance (see also below). One may choose the best or most frequently occurring clustering, but still do a manual inspection to check the sensibility of assignments. By comparing the outcomes from different initialisations, one can also investigate the *fuzziness* of cluster boundaries. Some points may repeatably be placed into different clusters, hence showing a certain instability. This can help to identify the boundaries of clusters and the overlaps among them without needing to switch to more involved procedures.

Finally, one needs to decide on the number of clusters C. There are several ways to do so and to spot eventual mis-assignments associated with an inappropriate value of C. The appropriateness of an assignment $\{x_i \mapsto C_c\}$ can be evaluated by its *silhouette coefficient*

$$S_i \equiv \frac{\bar{x_i} - \hat{\bar{x}}_i}{\max(\bar{x_i}, \hat{\bar{x}}_i)} \quad \in \quad [-1, 1] \, , \tag{21}$$

where $\bar{x_i}$ is the average distance of observation i to all other points in its cluster (as a measure for similarity to its cluster) and $\hat{\bar{x}}_i$ is the distance of i to points in the next nearest cluster.

Large positive values of S_i indicate a clear assignment of x_i to this cluster. On the contrary, a negative value indicates that x_i is more similar to observations in its neighbouring cluster. Graphically, the global clustering can be evaluated by vertically ordering all points according to their silhouette coefficient within each cluster and plotting their values horizontally, i.e. the x-axis being the silhouette coefficient (silhouette plot). A large fraction of observations with negative silhouette numbers within one cluster indicates a partially inappropriate clustering. This may be due to the properties of the data[16], an inappropriate number of clusters or a high-laying local minimum resulting from approximating (18). We have already presented solutions for causes one and three. An appropriate number of clusters C can be found by looking at the average absolute value of the silhouette coefficient $\bar{S}_C \in [0, 1]$ for different C. Assuming that the data can be clustered reasonably, $\bar{S}_C$ may first rise as C is increased, resulting in a maximum best value before leveling off. Note that this maximum may not be pronounced in practice leading to potentially ambiguous results.

An alternative to using the silhouette coefficient is the so-called "elbow method". The squared error function (18) or distance from the cluster center always decreases when increasing the number of clusters. This decrease will go from substantial to marginal with an increasing number of clusters. Plotting the error against the number of clusters, the resulting curve is likely to look like a bent arm where the optimal number of clusters lies in the elbow. The intuition behind this approach is that adding new clusters beyond the elbow point is not reflected anymore in the actual pattern of the data, but only increases the resolution of the already attained clustering.

A hypothetical example of a clustering analysis is shown in Figure 7. The original data (middle panel) exhibit a clear structure of four cluster by groups of one and three. We apply k-means clustering for two to ten clusters keeping track of the silhouette coefficient and the average distance of points from their associated cluster centers. The panel on the left shows the silhouette plot for five clusters. The negative coefficient values in cluster four (red circle) show that there is some misalignment of observations. This is in line with the silhouette coefficient (blue line in the right panel) peaking at a cluster number of four, the right number. Adding a fifth cluster, one of the actual clusters (three here) is split up increasing the resolution of this cluster. We obtain the same result when using the elbow method. The normalised cluster distance (green line in the right panel) decreases rapidly up until four clusters. Further decreases in average distance of points from their respective cluster centers are marginal. Hence, both methods retrieve the correct number of clusters. There is, however, one difference. The results obtained from using the silhouette coefficient could also be interpreted as only needing two clusters to correctly separate

[16]Like any statistical model, clustering can always be imposed on the data even if there is no structure to be found.

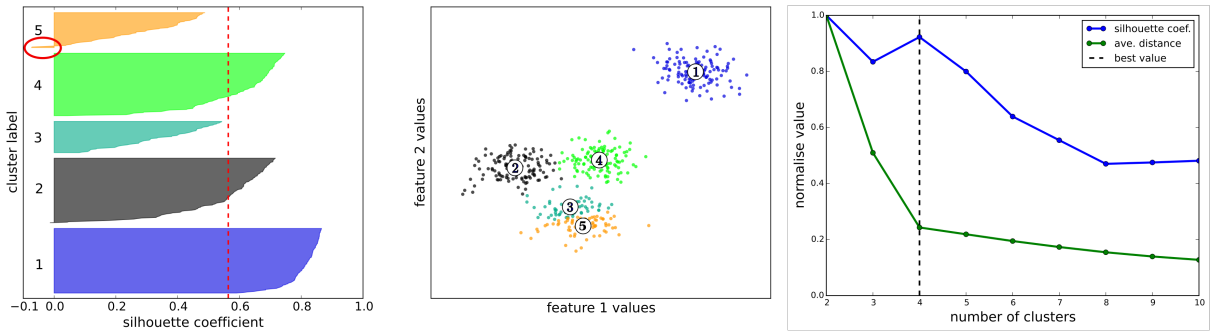

Figure 7: k-means cluster example. Left: Silhouette plot. Negative values (red circle) indicate the miss-assignment of observations. The vertical red line indicates the average silhouette coefficient for this clustering. Middle: Scatter plot of data of simulated data in a two dimensional feature space. Assigned cluster labels are shown for $k = 5$. Right: Silhouette and "elbow" number dependent on the number of clusters. Both methods point to an optimal number of $k = 4$. Source: Pedregosa (2011) and authors' calculations.

the data, which reflects the grouping of clusters in our data. This demonstrates that additional information may be needed to interpret the results of unsupervised learning techniques as there is no definitive evaluation score such as a prediction error. Another interpretation is that our data show a hierarchical structure of nested clusters. Cluster groups 1 and 2-5, are two clusters at low resolution, while four or more clusters are uncovered at higher resolutions.

3.2.2 Hierarchical clustering

Hierarchical clustering can be seen as a generalisation of k-means clustering. The basic idea of hierarchical clustering analyses (HCA) is to generate a spectrum of clusters at different resolutions with different numbers of clusters. That is, having m observations, one can generate clusterings with the number of clusters being between 1 (one giant cluster) and m (all clusters are singletons of one observation). Entropy measures or a Gini coefficient can serve as resolution parameters. HCA provides additional insights compared to a fixed number of clusters because one obtains a nested structure of clusters. For instance, a dataset may contain information about two properties A and B, encoded within a set of features. Performing HCA, data may cluster by properties A for a low number of clusters and property B for a high number, i.e. at a higher resolution. In a policy context, this may provide insights on the consequences of either affecting A or B. Different such properties may be lender or geographical information in mortgage data, or various counterparty information in financial transaction data. An HCA can be characterised by three components: the clustering direction, rules for cluster formation and a (dis)similarity measure (distant metric) between clusters. The main distinction within HCA

is made by the clustering direction where one distinguishes two types.

- **Bottom-up or agglomerative clustering:** One starts with each observation to form its own cluster (singletons). At each step, one merges the two most similar clusters, eventually subject to constraints. As a result, the number of clusters is one less and one moves one step up in the hierarchy. This process is repeated until all clusters are joint into a single giant cluster.

- **Top-down or divisive clustering:** One starts with all observations in a single cluster. At each step, one moves down the hierarchy by splitting one cluster at a time in a way that the dissimilarity between its two siblings is maximised. One ends up with m singleton clusters after $m-1$ steps.

The top-down approach is computationally more intensive and, therefore, bottom-up approaches are usually preferred. This can be seen as follows. Given m observations, it is infeasible to compare all possible splits of a given cluster even for small m. Doing so for joining two clusters is more tractable. This means that divisive approaches need to rely more on heuristics, or "clever algorithms", than agglomerative techniques (Fortunato (2010)).
We will focus on agglomerative clustering for the rest of the discussion. When evaluating the distance between two clusters for merging, the dissimilarity or distance between two clusters is based on the distances between individual observations in each cluster. Common such *linkage criteria*, are

- **Single linkage:** Minimal distance of two points in different clusters (*nearest-neighbour approach*).

- **Complete linkage:** Maximal distance of two points in different clusters (*furthest-neighbour approach*).

- **Average linkage:** Average distance of two points in different clusters.

- **Ward criterion:** Minimal within cluster distance variance of the joint cluster.

The single linkage and complete linkage criteria are the extreme cases of the average linkage case and may have the disadvantage of joining otherwise quite dissimilar observations (Friedman et al. (2009)). On the other hand, if there is a clear clustering structure in the data, all approaches will lead to comparable results.
Given two points x and z in the n-dimensional feature space, commonly used *distance or similarity metrics* are

- **Euclidean distance:** $\sqrt{\sum_{i=1}^{n}(x_i - z_i)^2}$
- **Cosine distance:** $1/cos(x, z) = \frac{||x||\cdot||z||}{x \cdot z}$
- **Manhattan distance:** $(\sum_{i=1}^{n} |x_i - z_i|)$

When faced with text data, or general non-numeric data, this first have to be transformed to numerical ranges before distance metrics can be applied (Bholat et al. (2015); Huang (2008)). The results of HCA are often presented as a tree diagram, known as a *dendrogram*. We will see an example in case study 3 where we use a dendrogram to demonstrate the relation between HCA and k-means clustering. (see Figure 20).

3.2.3 Recommender systems

Recommender system is not a machine learning algorithm in the original sense where a clearly defined problem type is provided, such as in supervised or unsupervised learning. They usually consist of blends of several methodologies.
As an example, consider online retail where recommender systems changed the way we interact with websites. There is an extensive set of web applications, especially in the context of online retail, social media and advertising, that involves predicting users' preferences and evaluating their choices. These systems can be grouped into two broad categories:

- **Collaborative filtering systems** recommend items based on similarity between users. Recommended items are those preferred by similar users.
- **Content-based systems** match user and product profiles. Users are recommended items with the highest overlap with their profile.

On the microeconomic level, recommender systems may be used to study individual preferences. On the macro level, they may provide new avenues for the evaluation of policy choices. For instance, one may gauge the reactions to and aptitude of monetary policy options, or changes in the regulatory framework of different financial institutions or market participants in general. Setting up collaborative filtering and content-based recommender systems based on the evaluations of consumer preferences could look as follows. Imagine that there are k consumers, m products and n product characteristics (features). The product data can be represented by a *utility matrix* $U \in \mathbb{R}^{k \times m}$. Each element represents a consumer-product pair and the degree of preference of a user for an item. One often assumes that the matrix is sparse, meaning that most entries are unknown. This means one does not know the preferences of consumers for most products.

The utility matrix is the basis for collaborative filtering. The similarity between consumers is given by the similarity between rows. This similarity can again be evaluated through various difference measures, such as Euclidean distance, cosine distance or the Jaccard index (Rajaraman and Ullman (2011)). Recommendations for consumer U_i are made by determining peer groups of consumers who are most similar to i, but where consumer i did not yet purchase/rate a product. This offers itself to a clustering analysis.

This framework can be modified to include consumer and product characteristics. Consumer profiles and product characteristics are represented by matrices $C \in \mathbb{R}^{k \times n}$ and $P \in \mathbb{R}^{m \times n}$, respectively. Each consumer or product is characterised by the same n features. By taking the matrix product $U = C \times P^T$, we obtain a new utility matrix, where each entry indicates the preference of a consumer for a certain product. In content-based systems, consumers are recommended products with the highest estimated preferences.

Having a time series of the utility matrices where subsequent purchases can be used as a target variables, it is possible to combine recommender systems with previously discussed learning methods, such as decisions trees. Recommender systems could be used in a central banking and regulatory settings in several contexts. For example, one may want to know whether mortgage borrowers are well informed before they purchase a certain product. To do that, one could merge mortgage flow data with buyers' profiles and see if they receive suitable products most of the time. If one detects systematic discrepancies from certain norms or observes trends within the market in this way, this may point to miss-selling, a market malfunction or be associated with unsustainable macroeconomic trends. The results may support regulatory action on the micro or macro-prudential level, depending on the outcome of diagnostics.

4 Model validation

Machine learning systems, like econometric models, rarely give the best possible result from the start. Adjustments and robustness checks are needed in most cases. We focus on evaluating and tuning machine learning approaches in this section. The presented techniques are also seen as useful for general statistical analysis. Further discussions of many of the presented concepts can be found in Friedman et al. (2009); Abu-Mostafa et al. (2012).

4.1 Performance metrics

The steps to build a machine learning system include training, validation and testing. In the first two stages, one estimates and calibrates a model to have the best generalisation properties. This means one tries to find the model within the domain of the chosen model class which

maximises test performance. By testing we mean out-of-sample prediction (see Figure 1). Direct performance evaluation is only possible for supervised learning by comparing the model output with unseen target data Y.

Depending on the problem, different performance metrics may be appropriate. For regression problems, squared errors are commonly used and the error rate, i.e. the fraction of misclassified observations, for classification problems.

An additional set of performance metrics to evaluate binary classification problems (Powers (2011)) is *accuracy, precision, recall* and *F-score*. These are summarised in Eq. 22. The error rate is one minus accuracy. Precision and recall are particularly interesting in the presence of skewed classes, i.e. where positive examples are relatively rare compared to negative ones[17].

$$\begin{aligned}
\text{accuracy} &= \frac{true\ positive + true\ negatives}{number of observations} && \in [0,1] \\
\text{precision} &= \frac{true\ positives}{true\ positive + false\ positives} && \in [0,1] \\
\text{recall} &= \frac{true\ positives}{true\ positive + false\ negatives} && \in [0,1] \\
\text{F}-\text{score} &= 2 \times \frac{precision \cdot recall}{precision + recall} && \in [0,1]
\end{aligned} \tag{22}$$

More generally, the above statistics describe different elements of a $2-by-2$ *confusion matrix*, where precision and recall measure the number of true positives relative to false positives (type-I error) and false negatives (type-II error), respectively. Possible interpretations of these two measures, respectively, are the fraction of relevant elements among the selected items and the fraction of selected relevant items overall.

4.2 Bias-variance trade-off

There are trade-offs between simple and complex models with regard to performance and cost. Simple models with few parameters or degrees of freedom are usually easier to compute but may lead to poorer fits, i.e., they are said to have a high *bias* or are *under-fitting* the data. On the contrary, complex models may provide more accurate fits, but risk being computationally expensive. Furthermore, they may *over-fit* the data, or have a high *variance*, resulting in equally large test errors. Optimal model complexity is determined using the bias-variance trade-off.

This trade-off is illustrated in Figure 8. The true function is a 4^{th}-order polynomial (green line).

[17] Additional performance metrics not discussed here are likelihood scores or receiver operating characteristics (ROC) curves and area-under-curve (AUC; one number) which can be visually represented.

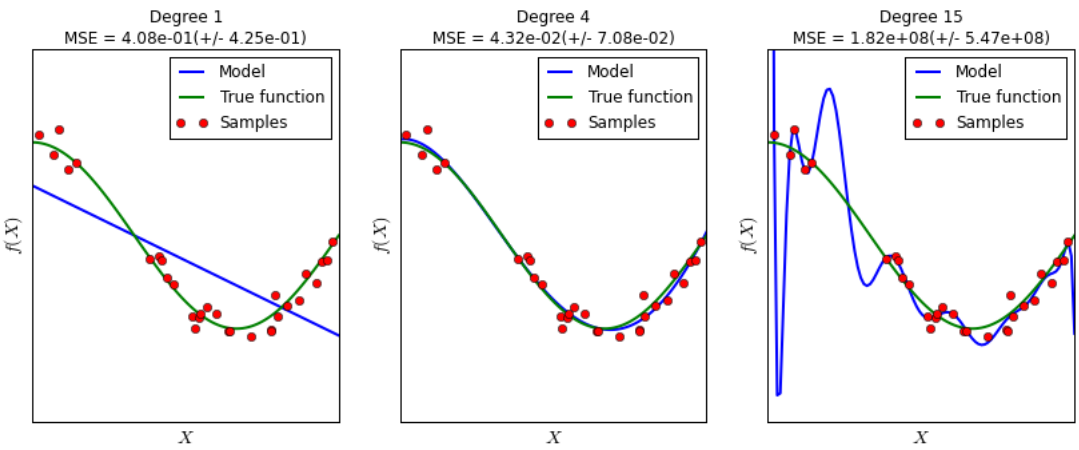

Figure 8: Under and over-fitting data. The true data generating process (DGP) stems from a fourth order polynomial and a bit of noise, shown by the green lines and red dot, respectively. Left: A straight line (degree-1 polynomial) clearly under-fits the data. Middle: A model with the appropriate complexity (degree-4 polynomial) achieves an almost perfect fit. Right: A too complex model (degree-15 polynomial) clearly over-fits the data, leading to an explosive mean squared error (MSE). Source: Pedregosa (2011) and authors' calculations.

Observation (red dots) from the true data generating process (DGP) will always deviate from the true function due to uncontrollable noise and measurement errors, to the point that it is hard to decide on the correct functional form and its complexity. The left and right panels show fits of the true DGP using a first and 15^{th}-order polynomial, respectively. They under and over-fit the data suffering from a high bias and high variance, respectively.

The expected squared error of a model $f(X)$ can be decomposed as

$$E[(f(X)-Y)^2] \quad = \quad \underbrace{E[f(X)-Y]^2}_{\text{bias}^2} \quad + \quad \underbrace{E[(f(X)-E[f(X)])^2]}_{\text{variance}} \quad + \quad \underbrace{\sigma^2}_{\text{noise}}, \qquad (23)$$

where the last term represents zero-mean stochastic noise. We see that the general error decomposition always has two components. Firstly, bias captures our model's inability to grasp the full complexity of the world. This is also called deterministic noise in the sense that it is determined at the moment we fix our model complexity in relation the complexity of the DGP. Secondly, variance is the result of fitting the random component of the error. Hence, high variance model is too complex given the available data. The best model usually lies at medium complexity, as illustrated in Figure 9. Deviations from the optimal model complexity in either direction will reduce test performance. Good practice is to follow the principle of Occam's razor, i.e. to start with a simple model and incrementally increase its complexity as required.

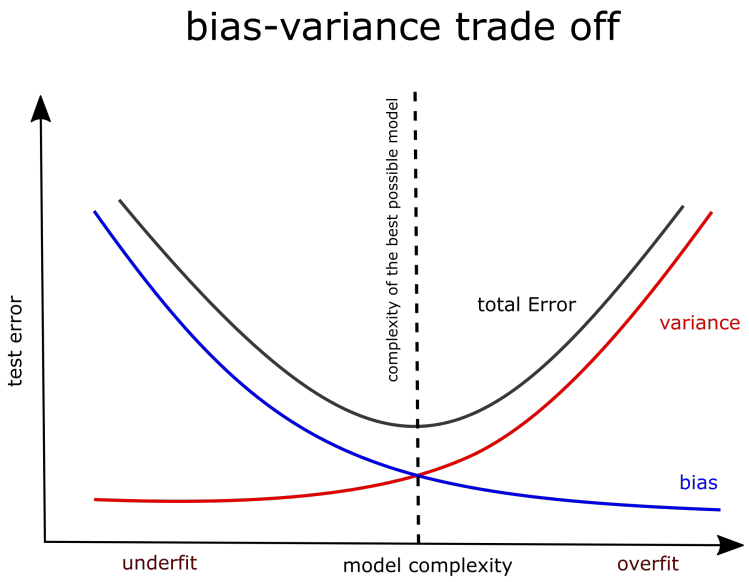

Figure 9: The bias-variance trade-off can be highlighted by plotting a model's bias and variance versus its complexity (degrees of freedom). The sum of the two (total test error) will have a minimum at the optimal complexity.

4.3 Cross-validation

Validation is the process of calibrating a model's *effective complexity* to maximise out-of-sample test performance. This may either mean the variation of a model's degrees of freedom, like the number of nodes and layers in a neural network or the number and order of features to consider, or tuning a hyper-parameter using a meta-algorithm (see next subsection).

One way to improve a model is cross-validation, where one compares the test performances of different model realisations with different sets or values of parameters. An efficient and popular form of cross-validation is *k-fold cross-validation*. Its idea is depicted in Figure 10. The training data (leaving the test data aside) are randomly sub-divided into k equal folds, where k is usually set between five and ten. For k different model realisations, use $k-1$ different folds for training and the remaining fold for testing (out-of-bag error). Choose the best such validated model and proceed to test it on the test dataset (out-of-sample error). If several parameters need to be tuned, repeat this process using the training set. The test set should be reserved for the final model evaluation to provide the best possible estimate of the model's generalisation properties. Note that this is a refinement of the initial framework laid out in Figure 1. Also, while there are asymptotic results showing that k-fold cross-validation can provide near-optimal model complexity (van der Vaart et al. (2006)), ad-hoc approaches for validation may be needed for small and medium sized samples, as discussed in cases 1 and 2 in Section 5.

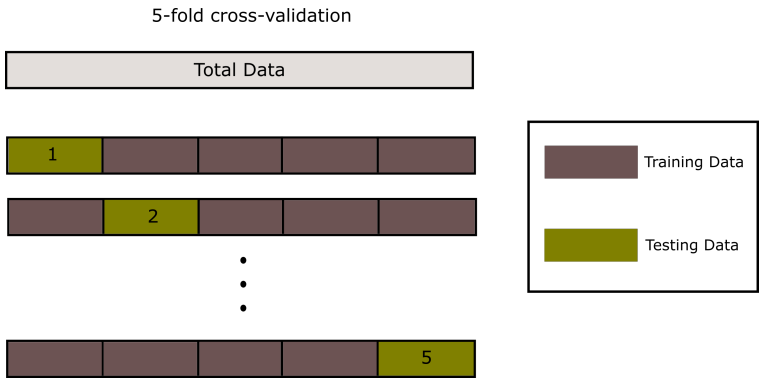

Figure 10: Schematic representation of 5-fold cross-validation. For five different values of a model hyper-parameter, one trains the model on 80% of the data (dark shaded areas) and tests this model on the remaining 20% (green shaded area). One chooses the parameter value with the lowest test error. All folds should be sampled randomly and testing performed out-of-bag.

4.4 Meta-algorithms

Meta-algorithms are methods for the tuning of model hyper-parameters or the adjustment of a model's structure in order to optimise test performance (model calibration). Hyper-parameters are what we collectively labelled λ in Eq. 2. They are usually set *ex-ante* and not determined through optimisation of the objective function. Meta-algorithms often involve the smoothing of model output or the use of ensembles, where cross-validation is a popular approach.

4.4.1 Regularisation

Regularisation is a smoothing method, which introduces a penalty term to the error function acting on non-zero, especially large, parameter values. Its effect is the reduction of the complexity, or the smoothening, of a model. This is useful when faced with an over-fitting model but also for feature selection. Note that regularization is not scale invariant. This means that variables should be appropriately normalised to comparable or unit scale. The penalty term to the objective function is called regularisation term. A generic form is

$$ERR(X, Y, \beta) \; + \; \lambda \sum_{j=1}^{n} \left[(1-\alpha) \left|\beta_j\right| + \alpha \left|\beta_j\right|^2 \right] \quad \alpha \in [0,1] \,, \tag{24}$$

where the positive hyper-parameter λ represents the weight we put on regularisation, where zero means no regularisation. α is an additional tuning parameter, characterising the type of regulariser. The general case, $\alpha \in (0,1)$, is called *elastic net*. $\alpha = 0$ leads to *Least absolute shrinkage and section operator* (LASSO) and $\alpha = 1$ to *Ridge regression*. Both LASSO and Ridge regularisation lead to parameter shrinkage, but LASSO tends to shrink parameters to zero and can thus be used for feature selection (Friedman et al. (2009)). Regularisation, either

in the above form or variations of it, can be applied to linear or logistic regressions, artificial neural networks and support vector machines. For the three former models, LASSO approaches may be used to reduce the dimension of the parameter space by setting certain variables to zero (feature selection; see Section 2.5.2). Penalised regressions are also a major field of application and research in econometrics and, as such, a clear point of contact with machine learning.
For example, gains from cross-validation, regularisation and testing may also provide useful extensions for multi-linear regressions. Under the assumptions of the Gauss-Markov theorem, the OLS estimator provides the best linear unbiased estimator (BLUE) for the parameters β if the model is linear in its parameters (Hayashi (2011)). There are several points where approaches from machine learning may improve on the OLS estimator depending on the *purpose* of the model. Even a complex model specification which is linear in parameters may not fully grasp reality, e.g. we are explicitly introducing deterministic noise into the model. By assumption, the resulting model would be unlikely to strike the best balance between bias and variance. Though offering the best linear approximation, it may perform poorly in describing the population in out-of-sample predictions. Adding a validated Ridge regulariser to the original model still provides an analytical solution and may improve on this situation (Raskutti et al. (2014)).

4.4.2 Bootstrapping, bagging, boosting and pruning trees

Bootstrapping is a technique which derives point estimates and confidence intervals for sample statistics based on re-sampling with replacement. It allows one to obtain approximate distributional functions over the population. It also provides a basis for other meta-algorithms (see next paragraph). An application of bootstrapping within an econometrics analysis is the approximation of the sampling distribution of model coefficients, which can be compared to asymptotic results (Freedman and Peters (1984); Li and Wang (1998)).

Bagging stands for bootstrap aggregation and is a popular method for improving an estimator's properties. The idea is that, when aggregating (averaging) the outputs of many estimators, one reduces the variance of the full model. When over-fitting is an issue, as for tree models, bagging is likely to improve performance. Bagging is a realisation of *ensemble learning*, where the final model consists of a collection of models. Ensemble techniques have also the advantage of generally improving accuracy compared to a single model due to the reduction in model variance (Opitz and Maclin (1999)). Bagging lies at the heart of case studies 1 and 2 in Section 5, where it also allows for the construction of prediction intervals around various model outputs. Note that bagged estimators may be biased if the underlying model is not linear in the data, which can again be addressed by cross-validation.

Boosting describes a set of methods which transform, or boost, a *weak* learner into a *strong* one (model). This means that one iteratively applies models to the data which individually have weak generalisation properties, but the final ensemble of models generalises well. Two popular boosting techniques are *gradient boosting* and *adaptive boosting*, or *adaBoost*. In gradient boosting, one sequentially updates the estimator in the direction of change of the loss function, i.e. a form of error correction. For adaBoost, one gives larger weights to misclassified or badly fitted observations at each consecutive step until one reaches a certain stopping criterion. Stopping criteria can be linked to target test or a flattening out-of-bag performance. AdaBoost requires a potentially large amount of data to converge and is susceptible to noise in the data. Boosting is often applied to tree models.

Pruning tree model is another very popular technique. As stated previously, over-fitting (high variance) is a serious problem for tree models. Therefore, one commonly restricts the maximal depth of a tree. Another technique to decide on an appropriate size for a tree model is pruning. One of the most common strategies is to stop splitting a node if it creates a new node of a size less than a certain value. This has already been described as a stopping rule.
An alternative to stopping rules, or *a priori* pruning, is to shrink a tree *a posteriori*, called reduced error pruning[18]. The full tree is grown first. Subsequently, one removes branches starting from the leaf nodes progressing towards the root. An inner (non-root, non-leaf) node is pruned by the deletion of all successor nodes. The resulting leaf node value is assigned by the majority vote or average value of the observations downstream. The following criteria may be applied for pruning:

1. A node is small, i.e. there are not too many down-stream observations in this branch.
2. A node is pure or almost pure, i.e. having a large class majority or a small variance in its target values.
3. Corresponding observations in a node are very similar in a pre-defined sense, e.g. by using a distance metric.
4. Performance in the training set is not changed significantly by the pruning process.

Finding a well-pruned candidate (e.g. via cross-validation) can be challenging. Promising candidates may not result from locally optimum pruning decisions (greedy strategies). That is why random forests may be a more appealing option than pruning a single tree.
In our later case studies, we applied *a priori* pruning by determining optimal tree depths (or

[18] Another popular method is cost-complexity pruning.

heights) of tree or forest models in a cross-validation exercise. This led to an increase in bias and a reduction in variance of the final model, hence, striking a balance between the two.

4.5 Learning curves

Learning curves plot a model's training and test errors, or the chosen performance metric, depending on the training set size (see Figure 11). They provide a set of useful model diagnostics. Firstly, they can be used to measure the amount of bias and variance in the model. The curves for in-sample (E_{in}) and out-of-sample (E_{out}) error converge to the amount of bias. Variance is then given by the difference between E_{out} and this level. Secondly, the height of the bias level and the relative shape of the learning curves help to tell if the model is too simple or too complex given a number of training data and performance goals. The LHS of Figure 11 shows a simple model where both learning curves rapidly converge to the achievable level of bias. This level may be too high and the corresponding model too simple. The right part shows a more complex model where a larger number of observations is needed to reach a lower bias level (as we move the dotted line from left to right). Having too few training examples will lead to a small E_{in}, but a large E_{out} (left dotted line in the RHS of Figure 11). Thus, the third insight we can gain from learning curves is on a model's generalisation error E_{gen}, defined as the difference between E_{out} and E_{in}. The lower it is, the more confident we can be about a model's out-of-sample test performance. The following relations summarise the above points

$$E_{out} = E_{in} + E_{gen} = bias + variance\,. \tag{25}$$

Fourth, given data constraints and performance goals, learning curves can help to calibrate the best combination of model complexity and the amount of data needed. This is important since it is often not feasible to collect more data. On the other hand, collecting more data is ineffective when having a simplistic high-bias model. However, it may be appropriate faced with a high-variance and low-bias model. As a result, the learning curve could suggest to increase model complexity or the collection of more data.

Practically, learning curves and confidence bounds for them can be obtained by randomly sampling training sets of different sizes. For each sample one fully fits the chosen model and evaluates E_{in} and E_{out}.

4.6 Time series modelling

Machine learning approaches are mostly cross-sectional in nature, i.e. time is not explicitly accounted for. However, the random sampling of observations for training and testing a model

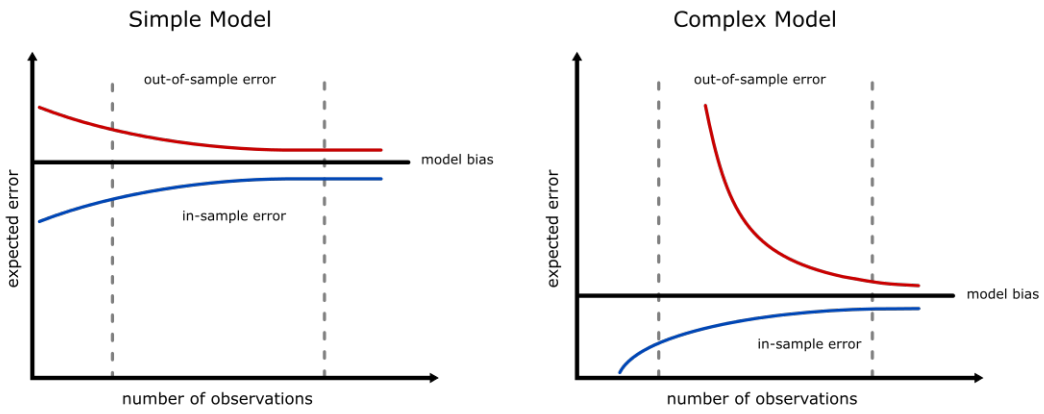

Figure 11: Left: Learning curves for a (too) simple model. In and out-of-sample errors both converge on a high level (y-axis) and for few training observations (x-axis). Right: Learning curves for a more complex model. Both learning curves may eventually converge at a low error level provided a large number of observations. Depending on the test performance and the feasibility of collecting more data, one may adjust model complexity or train one's model on a larger dataset if needed.

can be an issue when faced with a times dimension. Random sampling may either destroy serial correlation properties in the data which we would like to exploit, such as seasonality, or lead to biases. For example when training a model on future sampled data and using it for forecasting one may come to over-confident conclusions about a model's performance (*look-ahead bias*). Another potential issue is the missing of shifts or trends in the underlying data (non-stationarity), which may affect a models generalisation properties. On the other hand, plugging time series data into a machine learning model, such as neural network, may work just fine but one should be aware of the above.

Addressing the issue of training inconsistency and look-ahead bias, we present a simple training-testing framework in the context of projections. The basic idea is depicted in Figure 12. Assuming one has some form of projection model, an expanding horizon or sliding window approach can be used to train and test the data. At each time step, the training dataset progressively expands in time, while testing is performed on the projection horizon in the test dataset. The model is fully trained and tested at the sample end, where it can be used for the final projection. This framework can also be used to assess the impact of breaks to the input time series. Model performance or the relations between variables may change suddenly after a certain point in time (red dashed line). From a machine learning perspective, it would then be interesting to see how quickly a model is capable of re-learning important relations between features. This framework will be used in the case study on inflation forecasting in Section 5.2. We will see that the global financial crisis 2008/09 had an profound impact on the models' performance. More generally, a

sliding window can be used in this setting to detect changing relations (correlations) between variables. These may point to underlying structural changes. There exist additional ways to

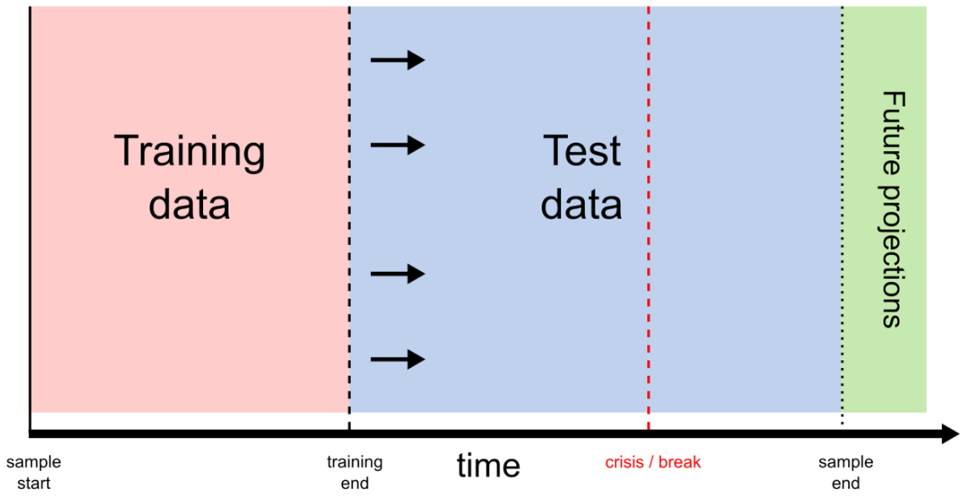

Figure 12: Schematic representation of a training-test framework for time-series projections. One trains a model on an initial training set. Model performance is evaluated projecting to the right and by using a window that either expands or slides to the right (repeated training). The model may be re-estimated or cross-validated at each time step depending on the application.

account for time, either implicitly or explicitly. Implicit methods are the including of lead-lag relations, auto-regressive features or features derived from time series analyses, e.g. spectral properties, time-series clustering, seasonalities or trends (Liao (2005)). Another option is to transform the problem. For instance, instead of modelling a continuous time series, one may bin the target variable and model transition probabilities between different states over varying time horizons (Giesecke et al. (2016)). Additional applications of machine learning techniques to time series analysis are the use of cross-validation to emprically determine optimal parameter values for techniques like exponential smoothing and weighted forecast combinations (Timmermann (2006)).

Methods which can take time explicitly into account include online learning techniques, recurrent neural networks and Bayesian approaches. *Online learning* systems, which are common for applications where constantly large amounts of new data arrive, such as online social networks and online retail, keep on training models continuously. Changes over time can be inferred from the change of the model's parameters over time. *Recurrent neural networks* include directed cycles of links and allow the interference of earlier and later inputs (short-term memory). Such a model may explicitly take seasonality in data into account after eventually learning it from the data itself. *Bayesian methods* allow the differentiation between past and present through the prior-posterior relation. A posterior distribution can be made prior and then updated by

newly arriving data. References for further reading are Friedman et al. (2009); Goodfellow et al. (2016); Bottou (1998); Barber (2012).

5 Case studies

We present a set of three case studies chosen to demonstrate the applicability of machine learning approaches in a central banking and policy context: (1) Predicting regulatory alerts on the balance sheet of financial institutions in an environment of incomplete information. (2) Forecasting UK CPI inflation on a medium-term horizon of two-year using a set of quarterly macroeconomic variables. (3) Investigation of the global funding network of investor-firm relations for tech start-ups with a focus on financial technology[19].

5.1 Case 1: Banking supervision under imperfect information

5.1.1 Description

Regular close scrutiny of banks' balance sheets has become a standard for financial supervisors following the financial crises. However, the manual inspection of hundreds or thousands of firms records' can be inefficient. Most firms will be sound and spotting complex relations between items for firms which are not, can be difficult. One approach which could make this process more efficient, but also more accurate, is to train a machine learning model on a set of validated supervisory alerts which indicate the need for closer scrutiny of a particular firm. Our first case study for supervised learning is the prediction of alerts associated with balance sheet items of financial institutions which could be reason for concerns. An example for a simple alert may be under or overshooting of certain threshold values of the quarterly change in a firm's assets. Large losses may raise concerns about the equity position of a firm, while large gains may raise doubts about the sustainability of these developments.

Here we present a stylised setting for such a framework. We start with a set of six well-populated items from firms' quarterly regulatory returns. We take the 20%-tile outlying part of the distribution of each measure across all firms and mark it as a single alert (outlier). We define a firm alert by having three or more single alerts within a given quarter. This will be our target

[19]Some of the used data sources contain sensitive information, such that not all details of the analyses and results are stated to respect this sensitivity. These include the precise composition of firms and the used period of time in case study 1, as well as the detailed composition of clusters and the identification of individual firms in case study 3. However, we elaborated on a clear presentation which should make all results transparent and reproducible on comparable datasets.
Furthermore, we would like to emphasis that the presented cases studies showcase prototype applications. They do currently not feed into regular supervision or projection processes.

name	has-3-alerts (Y)	leverage	capital	profitability	assets	CP exp. 1	CP exp. 2
unit	Boolean	log-ratio	%-change	log value	%-change	log value	log value
1-alerts	- -	L	L/R	L/R	L/R	R	R
count	840	840	840	840	840	840	840
mean	0.18	-1.07	0.01	3.74	0.02	9.36	9.30
std	0.39	0.44	0.17	5.42	0.12	0.98	0.96
min	0.00	-3.30	-0.96	-9.45	-0.49	6.34	6.34
25%	0.00	-1.28	-0.01	4.95	-0.02	8.61	8.58
50%	0.00	-1.13	0.00	6.00	0.01	9.21	9.14
75%	0.00	-0.95	0.01	6.97	0.05	9.94	9.86
max	1.00	2.38	2.90	9.87	1.47	12.03	11.87

Table 2: Summary statistics of supervisory data from financial institutions' balance sheets as model inputs for supervised learning. The target variable "has-3-alerts" indicates if a firm had three or more single alerts within the 20% outer most part of a measure's values. These single alerts are obtained by taking the observations in the left (L), right (R) or both (L/R) side of each indicator's distribution, respectively.

variable. The rational between this approach is as follows. Balance sheet items are intrinsically linked via accounting relations, although there may be misalignments over short time horizons, accounting practices and standards. The presented approach aims at spotting potentially unsustainable development in an institution's accounts when several items collectively move towards the fringes of the distributions defined by its peer group. To simulate an environment of incomplete information or uncertainty surrounding various measures, we remove two of the six input variables (features) used to create the target variable.

A summary of the used balance sheet measures[20] are given in Table 2, covering 140 national and international financial institutions of different sizes during six quarters. The target variable "has-3-alerts" is Boolean (classification problem) and has 18% *True* values. Assuming one knows the alert rate of 18%, this is the inverse error rate one would obtain by always guessing the majority class (*False*, i.e. no alert). This is the trivial benchmark model.

The second row of Table 2 shows the units and any eventual transformation of the data, which have been taken. The third row indicates the side of a single measure's distribution which has been used to generate single alerts. They have been chosen according to where one would expect eventual firm problems to materialise.

Table 3 shows the Pearson cross-correlations between all balance sheet items considered, where coloured fields indicate particularly strong correlations with p-values below 0.001 (green: positive values, red: negative values). All of these relations are as one would expect from the

[20] Assets: quarter-on-quarter change in total assets, capital: quarter-on-quarter change in CET1 ratio (common equity tier 1 capital over risk weighted assets), profitability: profit after tax earned in quarter, leverage ratio: capital over exposure, counterparty (CP) exposure 1 and 2 are pre and post risk conversion of off-balance sheet items. See Bank of England (2017) for more details.

name	has-3-alerts	leverage	capital	profitability	assets	CP exp. 1	CP exp. 2
has-3-alert	- -						
leverage	-0.310	- -					
capital	0.016	-0.024	- -				
profitability	-0.049	-0.115	0.035	- -			
assets	0.010	0.139	-0.165	-0.004	- -		
CP exposure 1	0.679	-0.452	-0.023	0.118	-0.040	- -	
CP exposure 2	0.701	-0.458	-0.023	0.122	-0.005	0.996	- -

Table 3: Pearson pairwise correlation structure of supervisory data. Coloured fields indicate strong correlations with a p-value below 0.001 (green: positive values, red: negative values).

underlying mechanical relations between balance sheet items. However, they are also far from perfect leaving substantial room for judgment and models. Both counterparty (CP) exposure measures are removed for modelling, leaving the final set of features: leverage, capital, profitability and assets. Note that this constitutes a substantial amount of information being removed as exposure measures are indicators showing substantial correlation with the target variable and among themselves. This leads the leverage ratio as the only measure substantially correlated with the target variable. We will, however, see that this information alone can be misleading when judging a variable's importance for a model's performance.

5.1.2 Model comparison

Training and test framework For the comparison of different models we adopt an approach of training, validation and testing suitable to the given dataset. The dataset is relatively small and the two target classes of having an alert or not are skewed towards the latter. This means that separating a considerable proportion of the data for testing or calibrating may lead to biased results. To address this issue, we train, calibrate and test every model four times where a randomly selected 75% of the observations are used for training and calibration. The remaining 25% are reserved and testing. We apply bagging and calculate out-of-bag errors for training and validation respectively on the larger portion of the data. That is, for 1000 bootstraps, we train a model on 430 observations and calibrate model hyper-parameters (cross-validation) on the remaining 200 observations (out-of-bag error), which corresponds to an approximate 70/30 split.The bootstrapped models are then averaged. Bagging has the advantage that it reduces model variance and allows for the measurement of between-model variation. This procedure is still prone to variations due to the initial four-fold partition of the dataset, which we address by repeating this procedure ten times over shuffled datasets.

Table 4 summarises model performances for the detection of supervisory alerts. The second col-

umn shows the values of cross-validated model hyper-parameters[21]. Advanced machine learning approaches are seen to generally outperform conventional approaches. For instance, the logic model does not perform considerably better in terms of accuracy than the trivial benchmark of never raising an alert. On the other hand, most models' test performance plateaus at around 92% accuracy, which is an example of the flat maximum effect. It states that there is no substantially best model in many situations but many different models may show similar performance. A small deviation from the flat-maximum effect is the slightly better performance of the random forest classifier. This is an example of an appropriate model choice as the intrinsic working of random forests matches well the DGP. Namely, by a combination of thresholding, a non-trivial rule of combining three or more thresholds and noise inductions through the removal of variables. The relatively poor performance of the Naïve Bayes classifier is not surprising in this case. Individual balance sheet items are not independent from each other invalidating the "naïve" base assumption of this model.

Precision and recall measure the number of correct alerts relative to false and missed alarms, respectively. Recall scores are lower overall, which would be worrying in a supervision context. Note, however, that this is related to the way we constructed this exercise, i.e. by removing much of the original input signal. Also note that this example ignored complimentary information which may be useful for the problem at hand. For example, we did not account for institution type and size, the overall macroeconomic environment, temporal relations between different quarters nor institution-specific information accumulated by supervisors over time.

We next compare the various model performances to a simple look-up strategy assuming that the cut-off thresholds for outliers are known. This is the type of approach in which these alerts would be flagged in a conventional analysis. Three different such approaches would raise an alert if either one, two or three or more outliers are detected within the four model features, respectively. The results of this exercise are shown in the lower three rows of Table 4, respectively. Unsurprisingly, the most conservative approach (1-alert) has the highest [H]. What is more interesting is that the recall score in the 2-alert case is comparable to most machine learning models. This means that this approach is equally powerful in identifying real alerts. However, it also raises substantially more false alarms and requires more information as inputs (the outlier cutoffs).

Machine learning models, like decision trees, learn these thresholds themselves, allowing them

[21] Naïve Bayes: Kernel function. k-NN: number of nearest neighbours and exponent of p-norm. decision tree: maximal tree depth. random forest regressor: number of trees (not validated) and maximal depth of single trees. Feed-forward artificial neural network (FFANN): regularisation parameter and number of hidden layers. Support vector machine (SVM): regularisation parameter and width of Gaussian kernel function. Logit: Regularisation parameter.

method	$cross-validation$	acc_{train}	acc_{test}	$precision$	$recall$	F_1
naïve Bayes	Gaussian kernel	75.8	74.8	38.5	64.4	48.2
k-NN	neighbours/5, p/1	94.8	91.4	81.4	68.4	74.3
decision tree	max. depth/6	98.2	90.4	76.3	68.9	72.4
random forest	trees/200, max. depth/9	100	92.3	84.6	70.5	76.9
FFANN	α/0.1, hidden/2	93.0	91.7	82.5	69.8	75.4
SVM	C/100, γ/1	96.0	92.0	83.1	70.2	76.1
Logit	C/0.1	81.5	81.2	36.1	4.2	7.6
1-alert	- -	- -	64.5	33.9	100	50.7
2-alert	- -	- -	84.0	55.0	68.0	60.8
3-alert	- -	- -	86.2	100	24.2	38.9

Table 4: Comparative performance statistics of various machine learning models for predicting supervisory alerts on the balance sheets of financial institutions. The second column describes cross-validated model characteristics. The last three rows show the results for look-up strategies assuming that the outlier thresholds are known.

to dynamically adjust their "believes" with newly arriving data. Such model output could be useful for the construction of simple models, such as "supervision heuristics", in an environment of uncertainty (Aikman et al. (2014)).

Conditional predictions for the random forest classifier are shown in Figure 13 (left). Profitability and changes in assets are varied within reasonable ranges observed in the data, while all other variables are being held constant on the value of the median observation. The white circle on the left of Figure 13 indicates the values for the two shown features with the conditioned value of the target variable inside the circle. We draw two conclusions from this. First, the model's reaction is highly non-linear (linearity would be represented by a straight colour gradient). The rectangular boundaries are typical for a tree-based model (see Figure 4). Varying colour shades and the fuzziness of some of the boundaries are related to the bagging procedure. Second, the model describes the median point in the sample well. The value inside the white circle matches its surroundings quite adequately as well.

This type of heat map can used to assess the conditions under which certain values of the output variable are attained. For instance, one clearly sees the important role played by profitability to indicate alerts from the abrupt transitions of high alert probabilities at the bottom and particularly at the top of the spectrum. At the upper end, the model output can intuitively be interpreted as saying that too high a profit rate may be unsustainable and therefore deserve closer scrutiny. More generally, it turns out that profitability is the most important variable in the current setting. This can be seen from the right of Figure 13 which shows the max-normed feature importance from the random forest classifier. Feature importance scores are obtained by measuring the decrease of the forest's error function across all trees when selecting a variable

at each split point of a single tree (see Eq. 9). The score of the most important feature is set to 100 for each bootstrap individually. The error bars represent the one standard deviation (SD) variation between bootstrapped models. The quotient of feature importance scores and its variation (signal-to-noise ratio) can be interpreted as the significance of this variable across bootstraps. More generally, variance across bootstrapped samples can be used to approximate sample statistics, which are not directly available for most machine learning models, such as confidence intervals and standard errors (Efron and Tibshirani (1986); DiCiccio and Efron (1996)). A summary of feature importance and signal-to-noise ratios is shown in Table 5. Model specification (1) is the main specification where counterparty exposures have been removed. The results for a full specification (2) with all variables which have been used to create the target alert variable is shown at the bottom. The variable rankings of both specifications are compatible with each other. Removing counterparty exposure measures constituted a major reduction in the amount of information available to the main specification, as seen from their large importance scores and signal-to-noise ratios. Note that this is directly related to the strong correlation between the two measures of CP exposures and how the target variable has been created.

name	leverage	CET1	profitability	assets	CP exp. 1	CP exp. 2
importance$^{(1)}$	63	19	100	27	- -	- -
signal-to-noise ratio$^{(1)}$	7.4	8.1	∞	8.6	- -	- -
importance$^{(2)}$	34	12	45	12	96	94
signal-to-noise ratio$^{(2)}$	6.8	6.1	6.7	5.7	14.0	12.0

Table 5: Feature importance and signal-to-noise ratios for two different model specifications: The partial model without counterparty (CP) exposures $^{(1)}$ used throughout and the full model $^{(2)}$.

5.1.3 Anomaly detection

The current case study is an example of anomaly detection. Anomaly detection refers to a collection of techniques aimed at the identification of observations which do not conform to an expected pattern or differ significantly compared to other observations. Practical examples where anomaly detection can be applied include fraudulent transactions, structural defects in goods, medical diagnostics, errors detection in texts or the cleaning of data (outlier detection; Chandola et al. (2009)).

There are two main categories of anomaly detection techniques. Unsupervised anomaly detection techniques look for anomalies in an unlabeled test data set. The main assumption of this type of technique is that the majority of the instances in the data set are similar. Supervised anomaly detection techniques require observations where entries have been labeled as “normal”

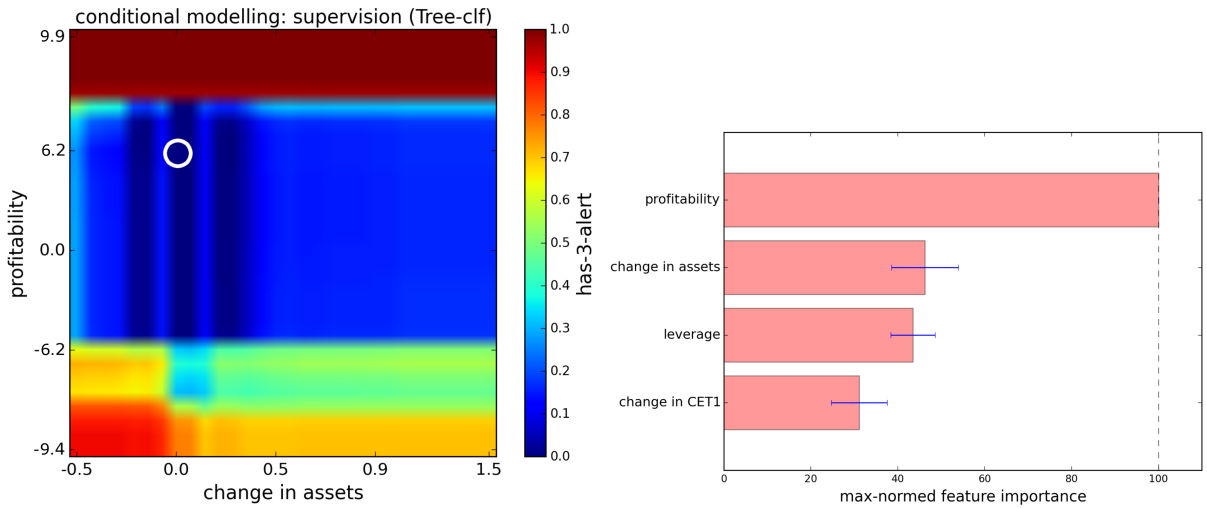

Figure 13: Left: Conditional model predictions for the bagged decision tree across the observed ranges for the change in profitability and the change in assets. The encircled point represents the median observation on which all other input variables are held with the actual value represented by the colour within. Right: Max-100-normed feature importance of random decision forest. The error bars show 1-SD variations across bootstrapped models. Profitability is seen as the most important feature with no variation across model realisations.

or "abnormal" and involves training a classifier. This is a two-class classification problem. The key difference from many other statistical classification problems is the inherent unbalanced nature of outliers vs non-outliers (*skewed classes*). This renders some error metrics, like error rates, impractical. Measures like precision and recall can be useful in this situation.

Several anomaly detection techniques have been proposed. Popular choices, which will not be discussed in more detail, are:

- Density-based techniques (k-nearest neighbor, local outlier factor, and variations thereof (Ramaswamy et al. (2000)).
- Subspace and correlation-based outlier detection for high-dimensional data (Zimek et al. (2012))
- One class support vector machines (Scholkopf et al. (2001)).
- Replicator neural networks (Hawkins et al. (2002)).
- Cluster analysis-based outlier detection
- Deviations from association rules and frequent item sets.
- Fuzzy logic based outlier detection.
- Ensemble techniques, using feature bagging, score normalisation and different sources of diversity (Lazarevic and Kumar (2005); Schubert et al. (2012)).

As for many machine learning algorithms, the most suitable method depends on the problem at hand and several options should be compared.

5.2 Case 2: UK CPI inflation forecasting

5.2.1 Description

We use a simple framework for consumer price index (CPI) inflation forecasting on a medium-term horizon of two years. One rational behind this example is that this is also the horizon on which monetary policy is usually thought to focus. We will also see that machine learning techniques can be applied to a traditional "small data world". We consider a set of quarterly macroeconomic variables for the UK between 1988 Q1 and 2015 Q4 which are publicly available[22]. To maximise reproducibility, the data and `Python` code for this case study are made available alongside the paper[23].

All approaches will be based on a lead-lag model where the target variable CPI inflation leads changes or the level of other variables (features) by two years. Features have been constructed to have comparable numerical values, such that we do not standardise them further. This also enhances the interpretability of models by facilitating the understanding of input-output relations. Time series for all variables, except for changes in commodity prices and the effective exchange rate, are shown in Figure 14 and a data summary is given in Table 6. The onset of the global financial crisis at the end of 2008 is marked by the vertical dashed line and constitutes a break point for linear time series modelling[24] (Zeileis (2002)). We will see that the crisis has a profound impact on any model's performance, but also that there are marked differences between them.

To get a rough idea of relations between different variables, we again compute the Pearson correlation matrices between them for the contemporaneous and lead-lag series. These are shown in Table 7. Values of relatively strong correlations are highlighted in green (positive) or red (negative). Most variable relations are as one expects them to be. However, there are some informative differences between the cases where CPI inflation leads by two years (second columns)

[22]Sources and descriptions: Bank of England Statistical Interactive Database (BoE), Office for National Statistics (ONS), Bank for International Settlements (BIS), World Bank. CPI: consumer price index (ONS ID: D7BT), M4: broad money supply (BoE ID: LPQAUYN), private sector debt (BIS ID: Q:GB:P:A:M:XDC:A, Bank for International Settlement (2016)), employment rate (ONS ID: LF24), unemployment rate (ONS ID: MGSX), GDP: gross domestic product (ONS ID: ABMI), labour productivity (ONS ID: A4YM), Bank rate: policy rate of the Bank of England (BoE ID: IUQLBEDR), 5YrBoYlImplInfl: implied inflation from the difference in yield curves of indexed and non-indexed 5-year gilts (BoE ID: IUQASIZC), ERI: real effective exchange rate of the pound sterling (BoE ID: XUQLBK82), Comm_idx: global commodity price index: average of energy and non-energy price indices, excluding precious metals (Work Bank, Pink Sheet (The World Bank (2016))), GDHI: gross disposable household income (ONS ID: QWND).

[23]Please see `https://github.com/andi-jo/ML_projection_toolbox`.

[24]Related to this, non-stationarity cannot be rejected for some of the series, particularly CPI inflation, which would have implications for the inference properties of linear models.

and the contemporaneous case (third column). First, lagged changes in the money supply and private debt are strongly correlated with changes in CPI, while contemporaneous relations seem to be weak. This is interesting, because they turn out to be important predictors for inflation but are largely absent in state-of-the-art macroeconomic models. Similar switches in the strength or even the sign of correlations hold for changes to employment, the level of the unemployment rate and GDP. Surprisingly, changes in CPI seem to be largely decoupled from externally facing variables on this time horizon, like the effective exchange rate and global commodity prices. The reason for this is the difference in times scales between changes in feature and response variable for the chosen horizon. All of these observations will be exposed in the machine learning models.

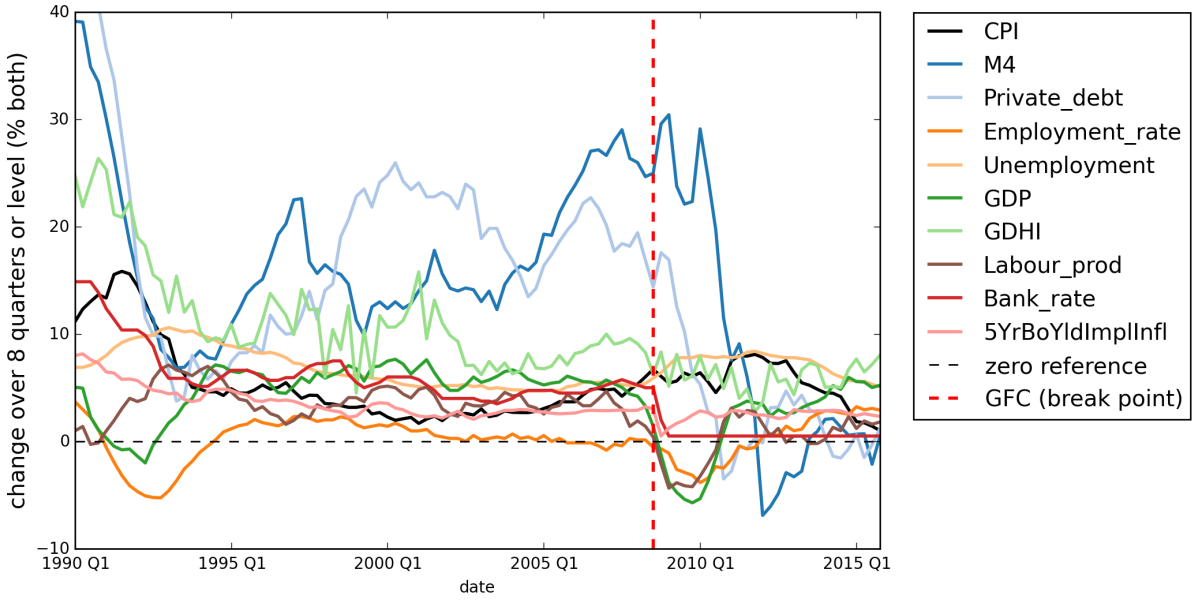

Figure 14: Time series of macroeconomic variables used for UK CPI inflation projections. Variables show two-year changes, except for the Bank rate, implied inflation and the unemployment rate, which are in levels. Changes in commodity prices and the effective exchange rate are not shown because of their large changes. The vertical dashed line indicates the onset of GFC in 2008 Q3.

name	CPI (target Y)	M4	Priv.debt	Empl. rate	Unempl.	GDP	GDHI	Lab. prod.	Bank rate	Impl. Infl.	ERI	Comm_idx
type	prices	debt		employment		income			rates		external	
change/level	change	change	change	change	level	change	change	change	level	level	change	change
count	104	104	104	104	104	104	104	104	104	104	104	104
mean	5.39	14.83	14.16	0.19	6.90	3.98	9.97	2.65	4.74	3.32	0.50	9.55
std	3.38	10.00	11.01	2.06	1.69	3.15	4.98	2.42	3.46	1.36	9.98	27.47
min	1.01	-6.89	-3.50	-5.26	4.70	-5.72	2.76	-4.36	0.50	0.50	-28.86	-45.53
25%	3.01	8.52	5.67	-0.55	5.30	3.05	6.75	1.40	0.50	2.55	-3.43	-9.88
50%	4.65	14.58	14.16	0.41	6.65	5.23	8.84	3.07	5.00	2.87	1.26	3.46
75%	6.29	22.35	21.69	1.60	8.00	6.06	11.72	4.05	6.00	3.75	4.75	27.89
max	15.84	39.14	52.25	3.70	10.60	7.65	26.36	7.11	14.88	8.16	31.73	84.99

Table 6: Summary statistics of UK macroeconomic time series data grouped by categories. All variables are chosen such that the bulk of their distributions roughly lies within comparable numerical ranges. Sources: BoE, ONS, BIS, World Bank and authors' calculations.

name	CPI_{+2y}	CPI	M4	Priv. debt	Empl. rate	Unempl.	GDP	GDHI	Lab. prod	Bank rate	Impl. infl.	ERI	Comm. idx
M4	.75	.18	- -										
Priv. debt	.64	.17	.77	- -									
Empl. rate	.28	-.45	.31	.54	- -								
Unempl.	.22	.61	-.18	-.26	-.31	- -							
GDP	.03	-.58	.10	.40	.77	-.31	- -						
GDHI	.61	.65	.58	.73	.23	.30	.12	- -					
Lab. prod	-.18	-.19	-.18	.10	.08	.10	.66	.15	- -				
Bank rate	.58	.48	.63	.80	.37	.14	.34	.90	.30	- -			
Impl. Infl.	.70	.70	.51	.64	.30	.44	.21	.87	.22	.88	- -		
ERI	-.27	-.19	-.12	.09	.44	-.23	.52	.04	.33	.21	.04	- -	
Comm. idx	.13	-.15	.10	.01	.12	-.33	.31	-.17	.19	-.08	-.07	.03	- -

Table 7: Contemporaneous pairwise Pearson correlation structure for macroeconomic time series up to 2013 Q4. CPI_{+2y} leads the other variables by two years, i.e. goes up to 2015 Q4. Two-year changes have been used for all variables except the unemployment rate, the Bank rate and implied inflation, which are in levels. Coloured fields indicate strong correlations with an p-value below 0.001 (green: positive values, red: negative values). Sources: BoE, ONS, BIS, World Bank and authors' calculations.

model description		training			testing (full period)			testing (pre-crisis)			testing (post-crisis)		
method	$cross-validation$	error	SD	corr.	error	SD	corr.	error	SD	corr.	error	SD	corr.
k-NN	neighbours/2, p/1	0.26	0.25	0.99	0.95	1.11	0.23	1.00	1.07	0.60	0.92	1.23	0.02
regression tree	max. depth/5	0.25	0.22	0.99	0.90	1.20	0.16	0.97	1.03	0.60	0.84	1.44	-0.18
random forest	trees/200, max. depth/11	0.28	0.24	0.99	0.86	1.16	0.21	0.93	0.97	0.65	0.82	1.39	-0.12
dFFANN	α/10, hidden/2, AF/ReLU	0.34	0.26	0.98	0.97	1.76	0.39	0.59	0.76	0.87	1.23	1.80	0.44
SVM	C/50, γ/0.001, ϵ/0.2	0.31	0.25	0.98	0.78	1.33	0.16	0.47	0.84	0.73	1.01	1.17	-0.26
SVM-dFFANN	as above	0.30	0.23	0.99	0.67	1.27	0.41	0.48	0.73	0.84	0.83	1.39	0.31
Ridge	α/10	0.44	0.27	0.97	1.38	2.43	0.30	0.88	1.43	0.82	1.73	2.42	0.29
VAR_1	- -	0.48	0.32	0.93	1.15	1.49	0.21	1.41	2.09	0.37	0.96	1.40	0.12
AR_p	p/BIC	0.16	0.11	0.98	0.95	1.05	0.16	1.00	0.96	0.61	0.91	1.18	-0.17
$AR_1^\star$	- -	1.71	2.03	0.70	2.02	0.95	0.15	1.61	0.70	0.60	2.47	1.00	-0.11

Table 8: Comparative performance statistics of various machine learning models for projecting UK CPI inflation on a two-year horizon. The second column describes cross-validated model characteristics. The combination of a support vector machine and a deep neural network outperforms all other single models. Error and SD columns show the mean absolute error and the one standard deviation of the error as multiples of the $AR_1^\star$ benchmark, for which absolute values are shown. Values below/above one indicate a better/worse performance relative to the $AR_1^\star$. Sources: BoE, ONS, BIS, World Bank and authors' calculations.

5.2.2 Model comparison

Training and test framework The lead-lag model above is trained and tested using the time series framework described in Section 4.6. We start with an initial training period of 15 years (1990 Q1 -2004 Q4) from which we draw 1000 bootstraps, where 30% of observations are reserved for out-of-bag testing. The out-of-sample test is performed forward looking, i.e. using the model for projections of CPI inflation two years ahead and compare it to the actual value. We repeat this procedure applying an expanding horizon. That is, we consecutively expand the training set by one quarter till reaching 2013 Q4 and evaluate the forecast two years ahead. Note that there is one major simplification in this approach. The current analysis is based on historical data, where some of the series may see substantial revisions which are not available at the moment of an actual forecast. This may introduce bias to the analysis, which can be addressed by training the model on various vintages of the data, especially when nearing the end of the current training period.

All model performances are significantly different for the periods before and after the GFC (2008 Q3). We therefore look at training and test statistics for four different periods. We state the out-of-bag error for the initial training period (1990 Q1 - 2004 Q4) and projection test errors distinguishing between the pre-crisis (2005 Q1 - 2010 Q3)[25] and the post-crisis (2010 Q3 - 2015 Q4) period, as well as the full test period (1990 Q1 - 2015 Q4).

The results from this exercise are shown in Table 8. The second column shows again the values of cross-validation parameter for each model, based on the full test period[26]. The mean absolute error, its standard deviation (SD) and the prediction-target correlation (*corr*) are given for the initial training and all test periods. The latter four models are given for reference: Ridge regression, vector autoregressive model with one lag (VAR_1), autoregressive models with one lag (AR_1) and autoregressive model with p lags (AR_p). The number of lags of the VAR model is restricted to one because of the relatively large number of parameters relative observations for longer lags. The number of lags p for the second AR model is set dynamically using Bayes information criterion. Dynamically means that p is adjusted at every step of our expanding horizon approach. Errors and SD are given relative to the $\text{AR}_1^\star$ benchmark, for which absolute values are shown. This makes model comparison easier. Values below/above one indicate a better/worse performance relative to the $\text{AR}_1^\star$.

[25] Using a lag of two years, crisis effects only start entering in the end of 2010.

[26] k-NN: number of nearest neighbours and exponent of p-norm. Regression tree: maximal tree depth. Random forest regressor: number of trees (not validated but chosen according to computational feasibility) and maximal depth of single trees. Feed-forward artificial neural network (FFANN): regularisation parameter, number of hidden layers and activation function. Support vector machine (SVM): regularisation parameter, width of Gaussian kernel function and penalty-free margin. Ridge regression: regularisation parameter. Autoregressive model (AR): number of lags by Bayesian information criterion.

We see that all models fit the data well during the training period where the out-of-bag error is used. In-sample forecasts on the two-year horizon are shown for the reference models[27]. We start to see model differences in the pre-crises test period. Particularly, the neural network and SVM perform well according to all measures. k-NN and tree-based models seem to show low levels of variance compared to the target variable resulting in a decreasing performance. One reason for this is the partly large variation seen between bootstrapped models results in low variability of the aggregated model. The performance of all models deteriorates markedly after the effects of the crisis kick in. This can be seen in the last section of Table 8. Absolute errors and variability (SD) double or even triple, together with significantly reduced model-target correlations which even turn negative in some cases. Interestingly, an unweighted combination of the previously best performing models (dFFANN and SVM) perform relatively well after the crisis. The reason for this is their opposite and mutually offsetting reactions to the shock. A fact which we investigate in more detail below and which will allow us to gain some insight into the working of these models.

The performance of all reference models is generally worse than that of most machine learning approaches. The Ridge regression achieves a decent fit before the GFC, but does not manage well thereafter. Here the AR models are doing slightly better in terms of variability. Their performance is close to those of tree-based models and k-NN.

Technical implementation Random forests and deep neural network can be computationally challenging. This is particularly the case in the current training-test framework for time series where a set of bootstrapped models has to be fitted for each observation in the test period. This led to some limitations (particularly on working memory) for the random forest regressor when using a standard laptop (4-core CPU with 2.60GHz, 16GB RAM, 64-bit operating system), where the number of trees in each forest was limited to 250.

Artificial neural networks are likely to pose technical challenges as they grow in size (Giesecke et al. (2016)). Parallellisation of model training, e.g. through the use of graphical processor units (GPUs), can be recommended in such cases. In our case, we faced the problem of having more parameters (144) in our model than observations across the full training period (112). We use the ReLU function $f(x) = max(0, x)$ for activation to address this. It automatically sets negative combinations of input features to zero, while positive inputs case a simple linear response. This reduces the final number of parameters in the model as well as the computational

[27]The $AR_1^\star$ benchmark performance is relatively poor within this period because of the sharp turn and drop-off in CPI inflation at the beginning of the period.

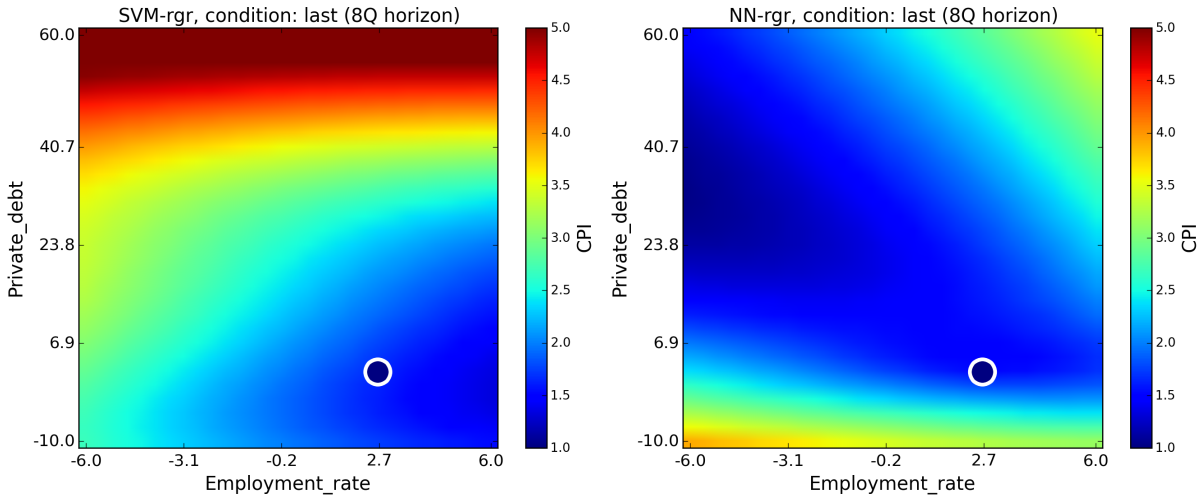

Figure 15: Conditional model predictions for a variable change in employment and private debt. All other variables are held at their value of last observation (2015 Q4). The white circle is set at the conditioning value with the actual level of change in CPI indicated by the colour within. Left: SVM. Right: FFANN. Both models approximately describe the conditioning point, but shows different structural output over the compared variables ranges. This can partly explain their different reaction to the GFC. Sources: BoE, ONS, BIS, World Bank and authors' calculations.

costs of training, while improving model performance. These properties make it one of the most widely used activation functions for deep learning (LeCun et al. (2015)).

Conditional model outputs For the support vector regressor (left) and artificial neural network (right) are shown in Figure 15. The change in the private debt and employment are varied within fixed ranges roughly observed over the whole period, while all other variables are being held constant on the value of the last observation (white circle). Two conclusions can be drawn from this. First, both models are highly non-linear. This also reflects the choice of the Gaussian kernel function for the SVM. Second, both models predict slightly too high a value for two-year changes in CPI as was actually recorded. The last point in the sample (if fully trained to the end of the observation period 2015 Q4) is represented by the white circle where the inside colour is taken from the data. This indicates that there is something within the macroeconomic development at that time which is not fully reflected in both model types. Third, the structural model reactions for this scenario vary significantly. This is related to both model types picking up different signals within the data, which we will investigate below. Note, however, that more observations would be needed for a full evaluation of the neural network.

This type of heat map may be used to assess the conditions under which certain values of the output variable are attained or how feasible it is to get to those conditions from the conditioning point. Looking at the SVM (LHS), target values of CPI inflation of about 4% on a two-year

horizon are associated with changes in private debt of about +40%, which are roughly independent from changes in unemployment. Given that this is inconsistent with most of the historic experience in Figure 14, one can conclude that this is inconceivable from the conditioning point and that other variables have to be aligned with such a development. This relates to the general point that certain regions in the model output space may be unattainable in reality. Intrinsic structural relations between input features may make certain variable combination unlikely. Finally, note that both models indicate higher levels of inflation for drops in employment when debt levels are falling. Partly counterintuitive, this feature is associated which observations from the early 1990s which both models have incorporated (Figure 14), hence learned.

Machine learning forecast combinations We have seen that any model's forecasting performance deteriorates substantially after the effects of the global financial 2008/09 crisis kick in at the end of 2010 (given a two-year lag). It turned out, however, that a simple unweighted combination of the two best performing machine learning models, the support vector regressor (SVM) and a deep neural network (dFFANN), improves modelling performance after the crises. Moreover, this combination provides the best performance across all investigated single models measured by overall test performance. The SVM-dFFANN output is shown in Figure 16, which also demonstrates the working of the presented time series framework. The initial training period is 15 year which is gradually expanded while consecutively training and testing the model. We see that the combined model (green line) well describes UK CPI inflation (blue line) during the training period (0.52% out-of-bag error) and the pre-crises projection period (0.73% projection error). The shift in performance is still marked for the post-crisis (2.05% projection error), while overall patterns are matched better than in all other models. As reference, a vector autoregressive model with one lag (VAR_1) is shown in gray. The VAR_1 is highly volatile in its output and does not well match inflation during the test period, neither pre nor post crisis.
The prediction intervals (red shaded area) are obtained through the bootstrapped training-testing setup. The increase in the size of prediction intervals in the test period is largely due to the neural network, which shows relatively high variance in its output. This is not surprising given its large number of parameters compared to the number of observations. Last but not least, inflation is seen to pick up during the final projection period. This signal is also obtain from the combined model which has been trained up to the end of the dataset (2015 Q4). This indicates that the two underlying machine learning models have partly learned new relations between variables after the crisis, which is expected to improve with time as new training data become available[28].

[28]This result is largely driven by the SVM.

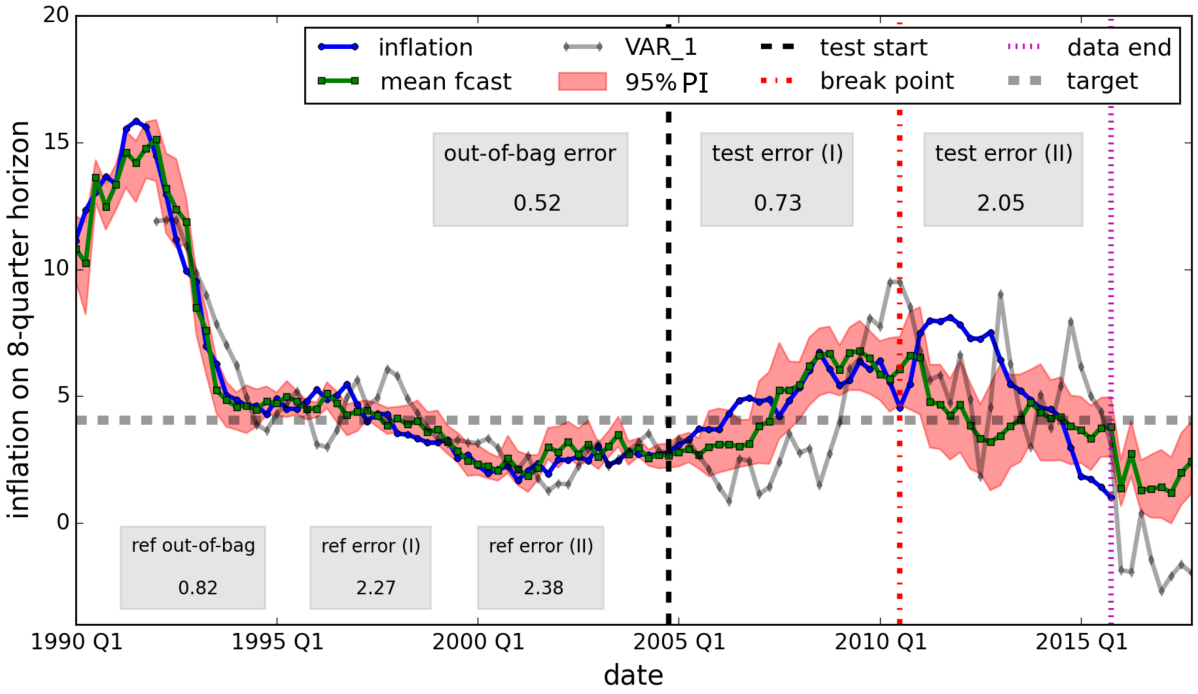

Figure 16: Averaged bootstrapped SVM-dFFANN projections (green line) for two-year changes in CPI (blue line). The shaded band indicates the 95% prediction interval (PI) across bootstrapped models. The vertical dashed lines separates the initial training, pre-crisis, post-crisis and post-data periods, respectively. Model and VAR_1 reference errors are given in the boxes. Sources: BoE, ONS, BIS, World Bank and authors' calculations.

5.2.3 Feature importance and "model forensics"

Understanding the performance and crisis reaction of the SVM-dFFANN combined model is not a straightforward task given the internal complexity of the two constituent models. Although challenging, looking at the building blocks of each model, one can learn about the model types in general and the problem in particular. Given the non-parametric nature of the SVM-dFFANN and complex interactions between a model's parameters and input data, one cannot expect to derive general rules or conditions under which one model or a combination of models will perform better or worse. Rather, one has to introspect the internal workings of each model and its interaction with the data. We do this by projecting out the dominant contributions of each models outputs from its inputs for different time periods. This will provide us with information about the relative importance of individual features in each model and if and how these change over time.

The dominant feature contributions $\vec{D}(\beta|X)$ for the SVM (17) with Gaussian kernel (16) and a FFANN with two hidden layers (dFFANN) with a rectified linear unit activation function,

$f(x) = max(0, x)$, given their respective sets of parameters β and conditioned on input data subsets X_s is given by

$$\vec{D}(\beta|X_s) = \begin{cases} \frac{1}{m_s}\sum_{i=1}^{m_s}\sum_{j\in\{SV\}}\alpha_j|\vec{SV}_j - \vec{x}_i| & \text{SVM} \\ \\ \frac{1}{m_s}\sum_{i=1}^{m_s}\sum_{j\in\{HL_1\}}\vec{x}_i \cdot W^T_{1;\cdot,j} & \text{dFFANN}\,. \end{cases} \tag{26}$$

$\vec{D}$ is a vector of length n (number of features) averaged over bootstrapped model realisations according to our training-test setup described in Section 3.1. The first sum in each expression averages across the subsample X_s with m_s observations. The explicit inclusion of the input data X in (26) is due to two particularities of our setup. First, inputs and subsequent model output are likely to vary within a time series setting due to trends or changes in levels of variables. By including the inputs one accounts for such changes. Second, the inclusion of inputs corrects for non-standardisation of features. The numerical magnitude of inputs is partly reflected in the numerical values of model parameters, especially for the case of the dFFANN. However, the combination of parameters and inputs neutralises this effect.

The following rationales are behind the model specific parts of Eq. 26. For the SVM, the absolute values of the differences between the support vectors (SV) and input data x_i provide the main feature contributions entering the kernel function (17). The coefficients α_j are weights for the SV which are the same for each feature. For the dFFANN, we consider the weighted sum over inputs to the first hidden layer (HL_1) of the product of feature inputs and transfer weights W_1. This simplification, especially the omission of the second hidden layer, is justified on the ground of two observations. Firstly, the product $\vec{x} \cdot W_1^T$ exhibits pronounced patterns between the input layer and the first hidden layer, indicating that the inputs from some features weigh heavier than those of others and that this structure is persistent across the network. Secondly, the second hidden layer only serves as model refinement. Including the second hidden layer improves model performance only marginally. After receiving dominant inputs from just a few input nodes, the entries of the second weight matrix carrying signals from the activated derived features in the first hidden layer to the second, are relatively homogeneous (no pronounced patterns) and the resulting signals of the second hidden layer show patterns similar to those of the first.

period	model	M4	Priv.debt	Empl. rate	Unempl.	GDP	GDHI	Lab. prod.	Bank rate	Impl. Infl.	ERI	Comm_idx
		debt		employment		income			rates		external	
pre-crisis	SVM	100	85	35	11	19	29	14	33	1	58	64
	dFFANN	7	100	6	21	13	90	8	58	35	13	24
post-crisis	SVM	88	100	3	6	25	12	2	36	15	38	17
	dFFANN	25	50	5	100	1	73	2	1	50	22	10

Table 9: Absolute values of max-100-normed feature importances for the SVM and dFFANN models, averaged across bootstrapped models for pre-crisis (1990 Q1 - 2009 Q4, two-year lag in inputs) and post-crisis (2011 Q1 - 2015 Q4). The largest two values are highlighted in green for each model. While scores are about stable for the SVM, the neural network adjusts its internal structure giving more weight to the unemployment rate after the GFC instead of changes in private debt. Sources: BoE, ONS, BIS, World Bank and authors' calculations.

We are now in a situation to evaluate the two models' reactions at different points in time. We split the full modelling period in two parts (X_s), a pre-crises period (1990 Q1 - 2009 Q4, note the two-year lag of inputs with respect to this period) and a post-crisis period (2011 Q1 - 2015 Q4). We leave out the year 2010 when the effects of the crisis phases in. The absolute values of max-100-normed feature importance are averaged over the respective periods and model bootstraps are shown in Table 9. The hierarchy between features, especially groups of features (debt, employment, income, rates, external), stays constant between bootstraps, while there is some variation in numerical values.

We observe two dominant features for each model and period which are highlighted in green. Concentrating on these we see some shared patterns but also discrepancies between the two models which explains their different reactions to the crisis. Before the crisis, the main inputs to both models came from changes in private debt. Changes in the money supply (M4) also factor into the SVM and changes to household' disposable income (GDHI) into the dFFANN. After the crises, the SVM did not adjust much. On the one hand, this can explain its largely deteriorated performance during this period. Changes to the money supply and private debt collapsed at some point while the input signals remained largely unchanged. Intuitively this can be understood from an SVM's construction around the input data. Previous episodes still need to be "sustained" by their support vectors, while new episodes require an adjustment of the model and as such more inputs from more recent periods. A sliding window approach may alleviate this situation but suffers from data scarcity in this example. On the other hand, the fully trained SVM captures the renewed size in CPI inflation during 2016 the best (see Figure 16). This could indicate that relations between variables actually did not change substantially, but were disturbed temporarily through the crisis and may have returned to normality.

On the contrary, the reaction of the dFFANN changed markedly in the post-crisis period partly offsetting that of the SVM. Its dominant signals in the post-crisis period stem from increased levels of unemployment and changes in GDHI. While roughly being correct with respect to the direction of travel, the dFFANN is overly pessimistic at the outbreak of the crisis, which saw inflation actually rise before falling off. This more flexible response is attributed to the more complex model structure. It is interesting to note that the dFFANN generally does not react much to the money supply but reacts to private debt and also does not treat these two variables as substitutes, which puts it apart from the SVM.

The differences in input patterns for the pre and post-crisis periods entering the first hidden layer of the dFFANN are shown in Figure 17. Each square shows the relative strength of the average signal originating from a feature entering each hidden neuron (NH), where darker shades

indicate stronger inputs. One sees that signals from only a subset of features enter dominantly and that this pattern changes between the two periods. The patterns reflect the numerical results shown in Table 9, but also provide some intuition behind the workings of artificial neural networks.

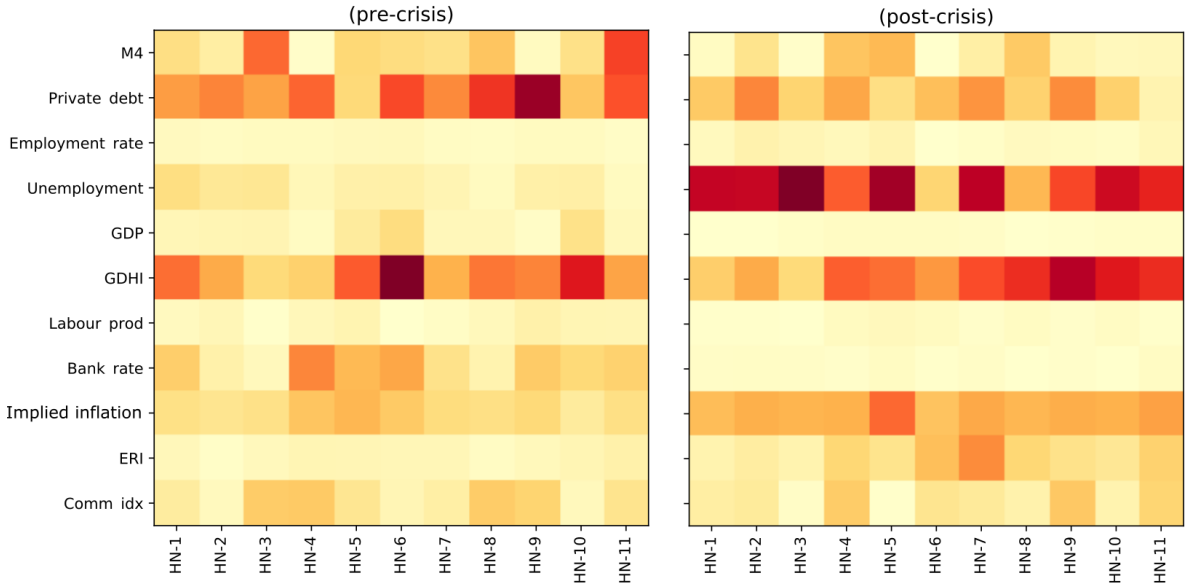

Figure 17: Input patterns to the first hidden layer of the dFFANN for the pre and post-crisis periods. Each square shows the relative strength of the average signal originating from each feature entering each hidden-layer neuron (HN) according to Eq. 26. A darker shading indicates a stronger signal. The observed patterns are in line with the values shown in Table 9. Sources: BoE, ONS, BIS, World Bank and authors' calculations.

Feature importance revisited The relative importance of individual features can be a crucial aspect to understand a model's output. A general idea to gauge the importance of an variable is the amount by which the inclusion or exclusion of a variable improves or deteriorates a model's performance. As shown in the previous case study, tree models, and particularly random forests, allow us to directly measure the relative importance of a variable and its fluctuation between different bootstraps. This can be done by measuring the reduction of the error functions across all splits (see Eq. 9). The most important feature is the one which reduces the overall error most.

Figure 18 summarises variable importances for different time horizons, that is, for different lead-lag relations between features and CPI inflation. The highest line shows the most important feature at each horizon normed to a value of 100. The shaded bands indicate 1-SD bands and the vertical dashed line marks the two-year horizon used throughout the current analysis. One can see that disposable income, Bank rate and implied inflation are the most important

features on short horizons. Changes in private debt turn out to be most important on the medium turn horizon, while changes in the money supply and employment are most influential on the long horizon. Note that these results are in line with the results from the analysis of the SVM-dFFANN model and the correlation patterns shown in Table 7. Also, some of the used variables are strongly correlated, while different relations may hold at different time horizons. In our example, this is the case for changes in private debt and the money supply. This means that they may mutually lower each other's importance[29]. This is indeed the case here. When excluding the money supply, its importance score is taken over by the change in private debt at this horizon.

An alternative and model-independent method to investigate variable importance is a counterfactual analysis. That is to measure the changes in model performance conditioned on leaving out one or several variables. We do several such counterfactual tests using different (sub)sets of features and models. Signals on feature importance from this exercise are generally in line with those from tree models. They are, however, substantially noisier in many cases and confidence intervals are often overlapping, making it difficult to draw clear conclusions in this example.

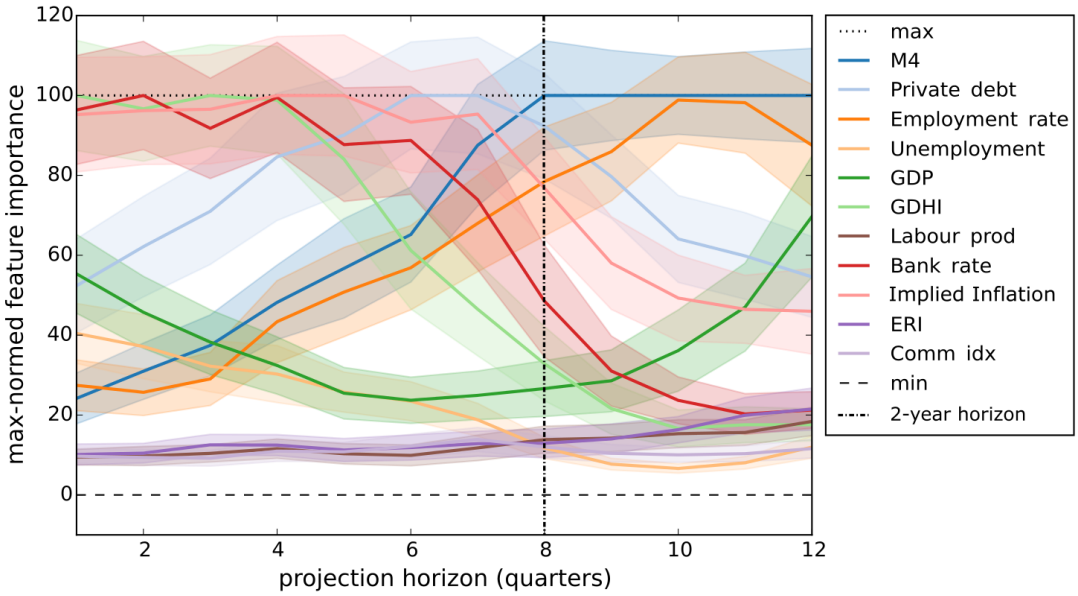

Figure 18: Max-100-normed feature importance for random forest model at different lead-lag relations between CPI inflation and the feature variables of 1 to 12 quarters. 1-SD variation bands are given for 250 bootstraps. The vertical dashed line marks the two-year horizon. Sources: BoE, ONS, BIS, World Bank and authors' calculations.

[29]This is similar to a linear regression model when using highly correlated covariates.

5.3 Case 3: Unicorns in financial technology

5.3.1 Description

We perform a clustering analysis of the universe of investor-firm funding relations for technology firms with the aim to shortlist potentially high-success firms in financial technology (fintech). This work can be useful for central banks in many ways (Bholat and Chakraborty (2017)). Technological changes have the potential to disrupt established industries. For instance, recent technology-driven firms like Uber and Airbnb are disrupting the taxi and hotel industries, respectively, partly changing the way those sectors operate. Similar changes may occur in the financial industry (if not already happening (The Economist (2017))). Given the centrality of the financial sector in the modern economy, central banks should try to anticipate and be prepared for the economic shocks such changes may bring. Hence, this work can help to understand how fintech firms are evolving and make sure that their contributions to the economy are valuable and not destabilising. The idea is to detect firms in their early stages and decide if they may become disruptive innovators in the financial industry. The underlying idea is that potentially successful firms are identified by being close to those firms in the feature space which have already been marked as such. Already successful firms are those firms marked as "unicorns" with a valuation of at least one billion USD (TechCrunch (2015)).

We use the CrunchBase database (TechCrunch (2015)), which is a comprehensive database of funding relations for technology start-ups. It contains data on company-investor relations up to a company's death, IPO, mergers or acquisitions relations between 1960 and June-2015. The full database consists of over 10 million funding relations, most of which are past 2010. A network representation of the dominant core component of investor-firm relations for the year 2014 is given in Figure 19. Nodes show either investors or firms. The network is largely bipartite, meaning that firms are either investors or recipients of funds, with the cardinality of both groups being approximately equal. Node sizes are proportional to the number of in-going investment relations, i.e. firms having many investors are larger. The funding network contains about 15,000 nodes (firms and investors) intermediated by 23,000 aggregated funding relations. It exhibits a rich structure. Besides a core-periphery arrangement, there are no dominant hubs and the overall network is very sparse. Many large investors are actually found in the periphery and the core is homogeneously connected. However, firms with many investors (larger nodes) are found towards the center of the network as are some dominant investors.

The node colours represent a possible community assignment[30]. Apart from an intuitive repre-

[30]Community detection tries to find a partition of a network into groups such that the number of intra-community links is maximised relative to the number of inter-community links. Intuitively, communities consist of nodes that have closer links among them than with members of other communities. It can provide important

sentation of this complex economic ecosystem, it will turn out that results from machine learning approaches are broadly in line with the shown assignment, however, drawing a more granular picture.

We select a set of four features to achieve a dense *a priori* clustering of unicorn firms. The amount of money raised per year, the average number of fundings received per year (rounds per year), the average number of individual investors per year and the average score of investors. The average score of investors is the fraction of investors who previously invested in highly-successful firm, i.e. so-called "king makers". Time averages are taken between the operation started or funding was first obtained (whichever is earlier) and firm exit or the end of the reporting period. The final sample contains about 33,000 firms. A descriptive summary of all features is given in Table 10. The investor score has been normed to lie between one and two. We can see that the data are highly right-skewed apart from the investor score. To partly counter this, but still keep the original heterogeneity (as it may point to the most successful firms), we standardise the data using z-scores. Extreme values in this data are caused by firms being in the database for either a very short or long periods of time.

name	funding / year	rounds / year	investors / year	investor score
count	32817	32817	32817	32817
mean	4.28e+06	0.55	1.03	1.22
std	1.26e+08	4.03	9.53	0.34
min	4.00e-02	0.02	0.02	1.00
25%	7.59e+04	0.07	0.10	1.00
50%	4.43e+05	0.28	0.32	1.00
75%	1.91e+06	0.63	0.84	1.40
max	2.08e+10	365.00	1095.00	2.00

Table 10: Summary statistics of constructed investment features for technology start-ups. Features are normalised using z-scores for the clustering analyses. Source: CrunchBase (TechCrunch (2015)) and authors' calculations.

5.3.2 Results

We use k-means clustering (see Section 3.2.1) to look for natural segmentations within firms' funding patterns. The number of clusters is determined using the elbow method. The green line on the LHS of Figure 20 shows the max-normed sum of distances of all observations to their respective cluster centres. It substantially flattens between four and six clusters. The signal

information on the organisational structure of a networked system. Network community detection can itself be classified as an unsupervised learning method. The current assignment is based on modularity maximisation. For a comprehensive introduction and review see Barabasi (2016) and Fortunato (2010)

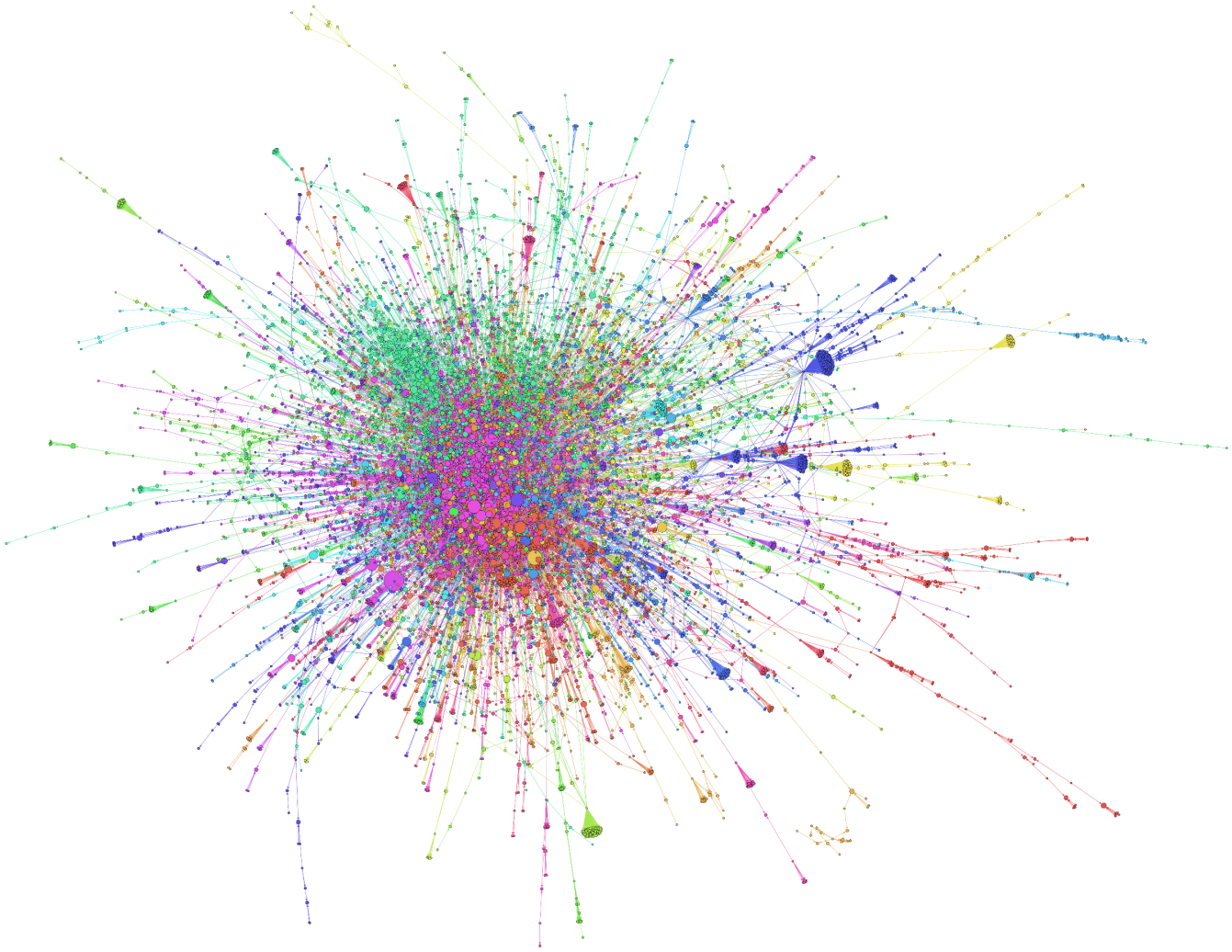

Figure 19: Visualisation of the funding network of global technology start-ups in 2014. Nodes represent either investors or firms. Node size is proportional to the number of investors a firm has (in-degree). Node colour has been assigned using a community detection algorithm via modularity maximisation (Barabasi (2016)). Source: CrunchBase (TechCrunch (2015)) and authors' calculations.

from the silhouette coefficient (blue line) is less clear in this case, but it also flattens out at about the same number of clusters. Based on this, the number of clusters is set to $k = 5$.

For comparison and for an initial evaluation of the obtained clustering, we also perform an agglomerative hierarchical clustering analysis (HCA) using average linkage and Euclidean distance (see Section 3.2.2). The resulting dendrogram is shown on the RHS of Figure 20. It has been truncated to 20 clusters (number of vertical end lines on the horizontal axis). The vertical axis indicates the average linkage distance between clusters and can be interpreted as a resolution parameter. The numbers on the left are the number of clusters of the corresponding clustering. The distances between cluster split points (blue dots) show a similar pattern to the green elbow line on the LHS, flattening at about $k = 5$ or 6 when read from the top to the bottom. The

split with $k = 5$ is highlighted by the horizontal black line. Noting that the precise structure is likely to be sensitive to the details of the clustering procedure, these observations are in line with those from the above exercise of determining k. The subsequent results are based on a k-means clustering with $k = 5$.

All previously identified unicorns are found in a single cluster of about $M_u = 3300$ firms. This cluster, its composition and size, are largely robust to the number of clusters set in a wide range. Note that this is expected from the procedure of feature selection which was based on minimising the volume of feature space covered by unicorns. They are clustered in only about 9% of the four-dimensional vector space covered by all firms. The remaining clusters consist of one dominant cluster of about 20.000 firms and three clusters of two, three and four thousand firms each.

This result is in line with the observation that the largest pink cluster at the centre of the network representation of firm-investor relations (Figure 19) contains most unicorn firms, although in a larger community than our unicorn cluster. This finding may be interpreted as highlighting the importance of the social component of investor-firm relations, in the sense that being part of a large and connected central community may increase the chances of successful funding and success overall.

We cannot visualise the overall clustering in a simple scatter plot, because of the 4-dimensional feature space. To get around this, we use a dimensionality reduction technique called t-distributed stochastic neighbour embedding (t-SNE; van der Maaten and Hinton (2008)) for visualising higher dimensional data in two or three dimensions[31]. The overall clustering is shown in Figure 21, where the unicorn cluster is marked in red with actual unicorns highlighted in green (about 200). This cluster is mostly well separated from the remaining firms, which provides us with confidence that the features we have chosen for this clustering are useful for our purposes. Next, we look out for firms which are close to unicorn firms in the feature space and which may be recommended to investors using a content-based recommendation engine (see subsection 3.2.3). Focusing on the unicorn cluster, we calculate a $(M_u \times M_u)$ utility matrix U. Element u_{ij} is calculated as the cosine distance (see Section 3.2.2) between the feature vector of firms i and j, indicating similarity. For all known unicorns, i.e. about 200 rows or columns, we record the ten most similar firms, i.e. utility values, to shortlist a subset of firms for this case study. This procedure leads to about 700 technology start-ups of which around 60 are financial technology firms[32].

[31] PCA (Principle Component Analysis) would be a common choice for this problem, but did not prove apt for the current application.

[32] Because of a high degree of overlaps among the recommended firms, these numbers are not too sensitive to the actually chosen cutoffs. That is, shortlisting slightly more or less than ten firms does not change the result

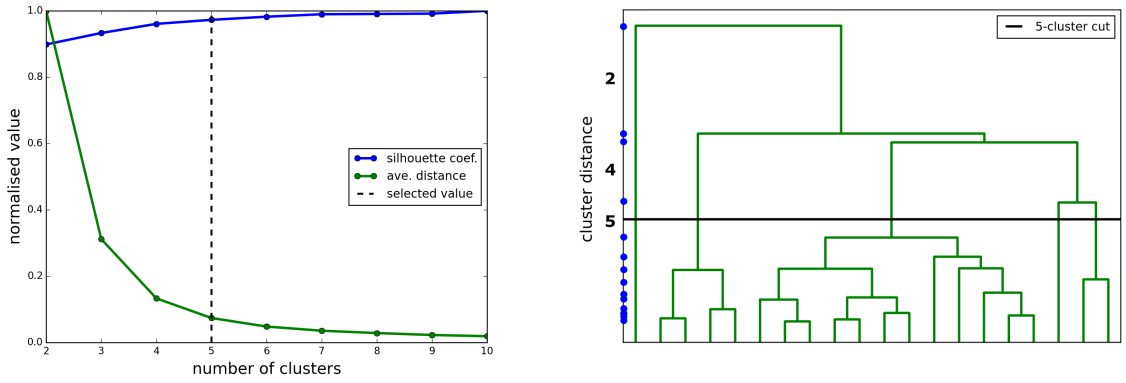

Figure 20: Comparative clustering analyses of technology funding data. Left: Elbow and silhouette coefficients for k-means clustering and k ranging from 2-10. The vertical dashed line indicates the selected number of clusters $k = 5$. Right: Dendrogram for HCA. The blue dots on the vertical axis indicate the relative values of the resolution parameter for cluster split points. The corresponding number of clusters is shown for low k-values. The horizontal line indicates the selected value $k = 5$ corresponding to the dashed line on the LHS. Source: CrunchBase (TechCrunch (2015)) and authors' calculations.

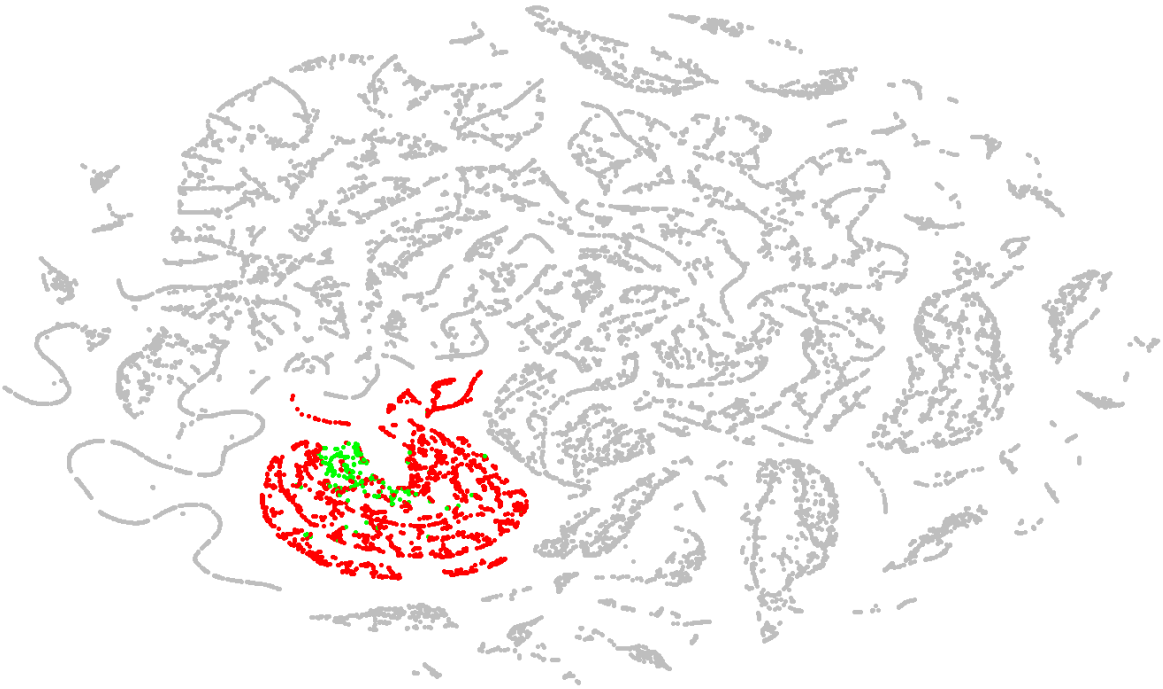

Figure 21: k-means clustering visualised in two dimensions based using t-SNE (van der Maaten and Hinton (2008)). The unicorn cluster is highlighted in red and actual unicorns are shown in green. The majority of firms in the unicorn cluster are well separated from other clusters. Axis dimensions do not have any particular meaning and are not shown. Source: CrunchBase (TechCrunch (2015)) and authors' calculations.

rank	2015	potential
1	USA	USA
2	UK	China
3	China	UK
4	France	India
5	Netherlands	Canada

Table 11: Country ranking by counts of unicorn firms in financial technology (fintech) as of 2015 and for identified high potential firms. Source: CrunchBase (TechCrunch (2015)) and authors' calculations.

Next we look at the business sectors and geographical distribution of the identified firms. Table 11 takes a look at the (potentially) changing geography of fintech firms. It shows a country ranking by country of residence counting current fintech unicorns and potential future unicorns based on our recommendation approach. The top of this ranking is fairly stable, while countries like India and Canada may produce more successful firms in the future. We have also noticed significant potential among Chinese FinTech firms (The Economist (2017)). This follows a general pattern we observe in the data. Namely, when looking at the geographical distribution and the activities potentially successful fintech firms operate in, many of them are based in emerging markets taking on functions traditionally served by a conventional banking system, such as the provision of funds, credit scoring and payments. This indicates that these firms are filling gaps of the established financial system. On a broader horizon, this may point to new avenues for wider technology-driven economic development.

Last but not least, we want to demonstrate how results from unsupervised learning can be used in a supervised setting, what additional insights can be gained from it and what questions it can throw up. We use the assignment of the above k-means clustering as a target variable and try to recover it from the original feature space. We address this classification problem with a decision tree (see Section 3.1.3). Additionally, we time-order the dataset. We train and validate our classifier on earlier data and test it on later observations. Training and validation is done on the first 20,000 records. We use 10-fold cross-validation to find the optimal tree depth (see Section 4.3). For depths between one and ten, we randomly split the dataset into 18,000 training and 2000 test observations and choose the depth leading to the minimal error, or the depths from which the error is no longer improving. It turns out that four is the optimal tree depth.

The tree classifier almost perfectly recovers the clustering labels with above 99% accuracy. Interestingly, it almost completely relies on the investor score to do so. This means, on the one hand, that the cluster structure can be recovered which is a form of consistency check and also expected. On the other hand, the information on this clustering can be compressed in only one

substantially.

feature. However, one needs to be careful when interpreting this result, as the investor score has been partly computed based on future data.

This points to two wider caveats of the current analysis which leave scope for future work. The first is a selection or survival bias which is based on not taking time into account. Firms tagged as successful may not always have been part of the unicorn cluster within the feature space. Furthermore, the funding patterns of successful firms suggest that these firms raise large amounts of money in a relatively small number of rounds. However, we do not know the direction of causality that this relation holds. That is, we cannot know what comes first. Do large funding amounts make firms successful or already successful firms have this pattern of funding? This issue is related to the wider point of correlation versus causation, where machine learning approaches lean heavily on the former.

6 Summary and conclusion

This paper introduced the main ingredients of the machine learning toolbox framed in a central banking and policy context. We presented the concept of machine learning systems and their components: problem types, data aspects, model classes, optimisation algorithms, validation and testing. The model classes included artificial neural networks, random forests, support vector machines and different clustering techniques among others. Model calibration techniques related to validation and testing are useful in their own rights. Note that there is no fixed machine learning model but recipes and a list of ingredients. As for the analogy of cooking, the successful design of a learning system depends on the combination of the right ingredients as well as the experience of the chef.

We presented a set of case studies relevant to central banks and regulators, discussing economic forecasting, banking supervision and financial market structure. Alongside these lines, we would like to stress the importance of an interdisciplinary approach to the associated problems. The handling of large datasets and the efficient construction of learning systems requires domain knowledge on the one hand and technical expertise on the other. The right combination of different skill sets is expected to provide better advice to decision makers than a monolithic approach.

The basic components of a learning system are *not* fundamentally new as components of statistical analyses, but rather broaden the toolbox available to econometricians. The novelty lies in the type of relations that can be modelled. Additionally, an increasing quantity of granular micro or high-frequency data are becoming available to central banks and regulators. These include single asset holdings and transactions of financial institutions, individual household portfolios

and consumption behaviours, firms' investment and hiring strategies, payments data, etc. Such datasets pose new opportunities and challenges, where tools from machine learning offer solutions. They allow one to model complex non-linear relations and can be used to structure and clean data at the same time. Universal approximators, like artificial neural networks, which are at the heart of the artificial intelligence revolution, may be used to uncover nuanced relations between variables and account for heterogeneity among agents. Taken further, the use of these tools may herald a new modelling paradigm on the micro and macro-economic level. Economic modelling, especially for the macro-economy, traditionally takes a deductive approach. That is, one starts with a set of (often strong) assumptions and tries to arrive at the largest generality possible. This approach risks being over-simplistic by ignoring important details and neglecting the possibility of emergence, where the total is more than the sum of its parts (Kirman (2016)). Historically, one of the main reasons for this approach, if not the main reason, has been a lack of data. In such a situation, simple structural models can provide some guidance. On the contrary, data-driven tools, such as those from machine learning, suggest an inductive approach to modelling. Given the vast range of possible descriptions of the world, they allow one to detect and investigate patterns on which structural models can be built. For instance, the derived features of a neural network may point to an unknown interaction between variables, which can then be used to build and support a model from first principles. In this way, one may be able to make a consistent transition between a micro and macro view of the economy.

There are caveats to machine learning. The three largest may be the difference between causal inference and prediction, the interpretability of many techniques and the modelling of time. Traditionally, machine learning tools have been developed and used to make predictions in the market place, i.e. with well defined payoff functions and with an attitude of "what-works-is-fine". Even though the general policy problem involves a prediction and a causal inference component, prediction alone may not be enough. Concretely, machine learning tools often ignore the issue of endogeneity and its many possible causes. Additionally, there are few known asymptotic or small-sample properties of estimators to draw on. These are serious issues when making decisions and more research is needed.

The second issue is interpretability. It is not always clear how a model's parameters can be interpreted or how a model generates its outputs from a set of inputs. Advanced techniques, like artificial neural network, random forests and support vector machines, can particularly suffer from this drawback and are often labeled as "black boxes". The degree to which this plays a role depends on the problem at hand. We provided suggestions of how to interpret various model classes. Two important points are the following. First, one should limit model complexity at

the onset. For example, start with a small neural network and add nodes and layers as needed (principle of Occam's razor). Small models are easier to interpret and to handle than large ones. Tuning the bias-variance trade-off helps to determine optimal model complexity. Learning curves and regularisation can be used to judge the appropriate level of complexity. Second, an econometric analysis may be used to evaluate and validate intermediate or final outputs, like derived features in a neural network, and test their significance in the given context. Additionally, the methods for model introspections which we presented in our analysis can further help to understand a model's working and to draw more robust conclusions.

The third issue is the ignorance of many machine learning techniques to the flow of time, i.e. the cross-sectional nature of most approaches. This can lead to biases in the analysis or the negligence of important information. The severity of the problem will again depend on the specific application. We provided a training and test framework in a projection setting which addresses these issues partly. We also discussed a variety of alternative approaches, such as online learning, models with internal memory and the modelling of transition probabilities.

Given that modern decision making is more and more evidence-based and built around data and that their amounts are rapidly expending, we believe that there is a large number of possible applications for machine learning at central banks, governments, statistical offices and similar institutions. In the long-run, these techniques may change the way we see, model and decide on socio-economic problems.

Bibliography

Abadie, A. and Kasy, M. (2017). The risk of machine learning. Working Paper 383316, Harvard University OpenScholar.

Abu-Mostafa, Y., Magdon-Ismail, M., and Lin, H.-T. (2012). *Learning from Data: A Short Course*. AMLBook.com. ISBN: 9781600490064.

ADRN (2017). The Administrative Data Research Network.

Aikman, D., Galesic, M., Gigerenzer, G., Kapadia, S., Katsikopolous, K., Kothiyal, A., Murphy, E., and Neumann, T. (2014). Taking uncertainty seriously: Simplicity versus complexity in financial regulation. Financial Stability Paper No. 28, Bank of England.

Albert, A. and Rajagopal, R. (2013). Smart meter driven segmentation: What your consumption says about you. *IEEE Transactions on Power Systems*, 28(4):4019 – 4030.

Altman, E. I., Marco, G., and Varetto, F. (1994). Corporate distress diagnosis: Comparisons using linear discriminant analysis and neural networks (the Italian experience). *Journal of Banking & Finance*, 18(3):505–529.

Angrist, J. D. and Pischke, J.-S. (2008). *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton University Press. ISBN: 0691120358.

Anstead, N. and O'Loughlin, B. (2015). Social media analysis and public opinion: The 2010 UK general election. *Journal of Computer-Mediated Communication*, 20(2):204–220.

Athey, S. and Imbens, G. (2015). Recursive Partitioning for Heterogeneous Causal Effects. *ArXiv e-prints*. 1504.01132.

Atz, U. and Bholat, D. (2016). Peer-to-peer lending and financial innovation in the United Kingdom. Staff Working Paper No. 598, Bank of England.

Bajari, P., Nekipelov, D., Ryan, S. P., and Yang, M. (2015). Machine learning methods for demand estimation. *American Economic Review*, 105(5):481–85.

Bakshy, E., Rosenn, I., Marlow, C., and Adamic, L. (2012). The role of social networks in information diffusion. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 519–528, New York, USA. ACM.

Bank for International Settlement (2016). Credit to the non-financial sector. `http://www.bis.org/statistics/totcredit.htm/`, last accessed: 11 December 2016.

Bank of England (2015). One Bank Research Agenda. http://www.bankofengland.co.uk/research/Documents/onebank/discussion.pdf, last accessed: 31 July 2017.

Bank of England (2017). CRD firms - reporting requirements. http://www.bankofengland.co.uk/pra/Pages/regulatorydata/formscrdfirms.aspx/, last accessed: 24 April 2017.

Barabasi, A.-L. (2016). *Network Science*. Cambridge University Press. ISBN: 1107076269.

Barber, D. (2012). *Bayesian Reasoning and Machine Learning*. Cambridge University Press. ISBN: 978-0521518147.

Belloni, A., Chernozhukov, V., and Hansen, C. (2012). Inference for High-Dimensional Sparse Econometric Models. *ArXiv e-prints*. 1201.0220.

Belloni, A., Chernozhukov, V., and Wei, Y. (2013). Post-Selection Inference for Generalized Linear Models with Many Controls. *ArXiv e-prints*. 1304.3969.

Bernard, S., Heutte, L., and Adam, S. (2010). *A study of strength and correlation in random forests*, pages 186–191. International Conference on Intelligent Computing. Springer Berlin Heidelberg. ISBN: 9783642148316.

Bholat, D. (2015). Big data and central banks. *Bank of England Quarterly Bulletin*, 55(1):86–93.

Bholat, D. and Chakraborty, C. (2017). Identifying 'Uber Risk:' Central Banks' Analytics Aid Stability. *Global Public Investor*, pages 102–103. https://ssrn.com/abstract=2987720, last accessed: 31 July 2017.

Bholat, D., Hansen, S., Santos, P., and Schonhardt-Bailey, C. (2015). Text mining for central banks: handbook. *Centre for Central Banking Studies*, (33):1–19.

Blanco, A., Pino-Mejías, R., Lara, J., and Rayo, S. (2013). Credit Scoring Models for the Microfinance Industry Using Neural Networks: Evidence from Peru. *Expert Syst. Appl.*, 40(1):356–364.

Blumenstock, J., Cadamuro, G., and On, R. (2015). Predicting poverty and wealth from mobile phone metadata. *Science*, 350:1073–1076.

Bottou, L. (1998). *Online Algorithms and Stochastic Approximations*. Cambridge University Press. ISBN: 9780521652636.

Bracke, P. and Tenreyo, S. (2016). History dependence in the housing market. Staff Working Paper No. 630, Bank of England.

Bridges, J., Gregory, D., Nielsen, M., Pezzini, S., Radia, A., and Spaltro, M. (2014). Benefits and costs of bank capital. Staff Working Paper No. 486, Bank of England.

Brooke, M., Bush, O., Edwards, R., Ellis, J., Francis, B., Harimohan, R., Neiss, K., and Casper, S. (2015). Measuring the macroeconomic costs and benefits of higher UK bank capital requirements. Financial Stability Paper No.35, Bank of England.

Chan, H. K., Subramanian, N., and Abdulrahman, M. (2016). *Supply Chain Management in the Big Data Era*. Advances in Logistics, Operations, and Management Science. IGI Global. ISBN: 9781522509578.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15:1–15:58.

Cielinska, O., Joseph, A., Shreyas, U., Tanner, J., and Vasios, M. (2017). Gauging market dynamics using emir trade repository data: The case of the Swiss Franc de-pegging. Financial Stability Paper No. 41, Bank of England.

Coats, P. K. and Fant, L. F. (1993). Recognizing financial distress patterns using a neural network tool. *Financial Management*, 22(3):142–155.

Crawford, K. (2013). The hidden biases of big data. Harvard Business Review, Microsoft Research.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314.

Daas, P., Bart, B., van Den, H. P., and Puts, M. (2015). Big data as a source for official statistics. *Journal of Official Statistics*, 31(2):249–262.

Dagher, J. C., Dell'Ariccia, G., Laeven, L., Ratnovski, L., and Tong, H. (2016). Benefits and costs of bank capital. Working Paper SDN/16/04, International Monetary Fund.

Dendramis, Y., Giraitis, L., and Kapetanios, G. (2017). Estimation of time varying covariance matrices for large datasets. IAAE, 26-30 June 2017, Sapporo, Japan.

DiCiccio, T. J. and Efron, B. (1996). Bootstrap confidence intervals. *Statistical Science*, 11(3):189–212.

Efron, B. and Tibshirani, R. (1986). Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical Science*, 1(1):54–75.

Einav, L. and Levin, J. D. (2013). The data revolution and economic analysis. Working Paper 19035, National Bureau of Economic Research.

European Commission (10 October 2014). Solvency II: European commission delegated regulation (EU) 2015/35. http://eur-lex.europa.eu/eli/reg_del/2015/35/oj, last accessed: 10 Aug 2017.

European Commission (4 July 2012). European Market Infrastructure Regulation (EMIR): Regulation (EU) no 648/2012. http://eur-lex.europa.eu/eli/reg/2012/648/oj, last accessed: 10 Aug 2017.

Finlay, S. (2014). *Predictive Analytics, Data Mining and Big Data: Myths, Misconceptions and Methods*. Business in the Digital Economy. Palgrave Macmillan. ISBN: 9781137379283.

Firestone, S., Lorenc, A., and Ranish, B. (2017). An Empirical Economic Assessment of the Costs and Benefits of Bank Capital in the US. Finance and Economics Discussion Series 2017-034, Board of Governors of the Federal Reserve System (U.S.).

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, 486:75–174.

Freedman, D. and Peters, S. (1984). Bootstrapping a regression equation: Some empirical results. *Journal of the American Statistical Association*, 79(385):97–106.

Friedman, J., Hastie, T., and Tibshirani, R. (2009). *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin. ISBN: 978-0-387-84858-7.

Galton, F. (1907). Vox Populi. *Nature*, 75:450–451.

Giesecke, K., Sirignano, J., and Sadhwani, A. (2016). Deep learning for mortgage risk. Working paper, Stanford University. arXiv: 1607.02470.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Harding, M. and Lamarche, C. (2016). Empowering consumers through data and smart technology: Experimental evidence on the consequences of time-of-use electricity pricing policies. *Journal of Policy Analysis and Management*.

Harding, M. and Lovenheim, M. (2014). The effect of prices on nutrition: Comparing the impact of product- and nutrient-specific taxes. Working Paper 19781, National Bureau of Economic Research.

Hawkins, S., He, H., Williams, G., and Baxter, R. (2002). Data warehousing and knowledge discovery. *Lecture Notes in Computer Science*, 2454:170–180. ISBN: 9783540441236.

Hayashi, F. (2011). *Econometrics*. Princeton University Press. ISBN: 9781400823833.

Huang, A. (2008). Similarity measures for text document clustering. *Proceedings of the Sixth New Zealand Computer Science Research Student*, pages 49–56.

Joseph, A. and Chen, G. (2014). Composite centrality: A natural scale for complex evolving networks. *Physica D*, 267:58–67.

Khashman, A. (2011). Credit Risk Evaluation Using Neural Networks: Emotional Versus Conventional Models. *Appl. Soft Comput.*, 11(8):5477–5484.

Kirilenko, A., Kyle, A., Tuzun, T., and Samadi, M. (2017 (forthcoming)). The flash crash: High frequency trading in an electronic market. *Journal of Finance*.

Kirman, A. (2016). Complexity and economic policy: A paradigm shift or a change in perspective? A review essay on David Colander and Roland Kupers's complexity and the art of public policy. *Journal of Economic Literature*, 54(2):534–72.

Kleinberg, J., Ludwig, J., Mullainathan, S., and Obermeyer, Z. (2015). Prediction policy problems. *American Economic Review*, 105(5):491–95.

Kotsiantis, S. B. (2007). Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pages 3–24. IOS Press.

Kursa, M. and Rudnicki, W. (2010). Feature selection with the Boruta package. *Journal of Statistical Software*, 36(11):1–13.

Lacher, R. C., Coats, P. K., Sharma, S. C., and Fant, L. F. (1995). A neural network for classifying the financial health of a firm. *European Journal of Operational Research*, 85(1):53–65.

Lazarevic, A. and Kumar, V. (2005). Feature bagging for outlier detection. *11th ACM SIGKDD international conference on Knowledge Discovery in Data Mining*, pages 157–166.

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

Li, Q. and Wang, S. (1998). A simple consistent bootstrap test for a parametric regression function. *Journal of Econometrics*, 87(1):145 – 165.

Liao, W. (2005). Clustering of time series data - a survey. *Pattern Recognition*, 38(11):1857 – 1874.

Lin, H. W., Tegmark, M., and Rolnick, D. (2016). Why does deep and cheap learning work so well? *ArXiv e-prints*. 1608.08225.

Lipton, Z., Berkowitz, J., and Elkan, C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning. *ArXiv e-prints*. 1506.00019.

Mayer-Schönberger, V. and Cukier, K. (2013). *Big Data: A Revolution that Will Transform how We Live, Work, and Think*. Houghton Mifflin Harcourt. ISBN: 9780544002692.

Mbiti, I. and Weil, D. (2011). Mobile banking: The impact of M-Pesa in Kenya. Working Paper 17129, National Bureau of Economic Research.

Mehta, P. and Schwab, D. J. (2014). An exact mapping between the Variational Renormalization Group and Deep Learning. *ArXiv e-prints*. 1410.3831.

Mullainathan, S. and Spiess, J. (2017). Machine learning: An applied econometric approach. *Journal of Economic Perspectives*, 31(2):87–106.

Navia-Vazquez, A. and Parrado-Hernandez, E. (2006). Support vector machine interpretation. *Neurocomputing*, 69(13-15):1754 – 1759.

Ng, A. (2015). Machine Learning - Stanford University. Coursera MOOC.

Njuguna, C. and McSharry, P. (2017). Constructing spatiotemporal poverty indices from big data. *Journal of Business Research*, 70:318–327.

Office for National Statistics (2014). Underemployment and overemployment in the UK. `http://webarchive.nationalarchives.gov.uk/20160107092648/http://www.ons.gov.uk/ons/rel/lmac/underemployed-workers-in-the-uk/2014/rpt-underemployment-and-overemployment-2014.html`, last accessed: 10 Aug 2017.

Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198.

Pedregosa, F. e. a. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Portnoy, S. (1988). Asymptotic behavior of likelihood methods for exponential families when the number of parameters tends to infinity. *Ann. Statist.*, (1):356–366.

Powers, D. (2011). Evaluation: From precision, recall and f-measure to ROC., informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63.

Rajaraman, A. and Ullman, J. (2011). *Mining of Massive Datasets*, volume 1. Cambridge University Press.

Ramaswamy, S., Rastogi, R., and Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, page 427. ISBN: 1-58113-217-4.

Raskutti, G., Wainwright, M. J., and Yu, B. (2014). Early stopping and non-parametric regression: An optimal data-dependent stopping rule. *Journal of Machine Learning Research*, 15(1):335–366.

Salchenberger, L. M., Cinar, E. M., and Lash, N. A. (1992). Neural networks: A new tool for predicting thrift failures. *Decision Sciences*, 23(4):899–916.

Scholkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471.

Schubert, E., Wojdanowski, R., Zimek, A., and Kriegel, H. P. (2012). On evaluation of outlier rankings and outlier scores. *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 1047–1058. ISBN: 9781611972320.

Schwab, D. and Mehta, P. (2016). Comment on "why does deep and cheap learning work so well?". *ArXiv e-prints*. 1609.03541.

Sibanda, W. and Pretorius, P. (2012). Artificial neural networks- a review of applications of neural networks in the modeling of HIV epidemic. *International Journal of Computer Applications*, 44(16):1–4.

Silver, D. e. a. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484 – 489.

Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222.

Solomon, J. (2015). *Numerical Algorithms: Methods for Computer Vision, Machine Learning, and Graphics*. A. K. Peters, Ltd., Natick, MA, USA. ISBN: 1482251884, 9781482251883.

Steel, M. (2011). Bayesian Model Averaging and Forecasting. *Bulletin of E.U. and U.S. Inflation and Macroeconomic Analysis*, 200:30–41.

Tam, K. Y. and Kiang, M. Y. (1992). Managerial applications of neural networks: The case of bank failure predictions. *Management Science*, 38(7):926–947.

TechCrunch (2015). CrunchBase Database. `https://www.crunchbase.com/`, last accessed: 13th June 2016.

Tetlock, P. (2007). Giving content to investor sentiment: The role of media in the stock market. *Journal of Finance*, 62(3):1139–1168.

The Economist (21 Nov 2015). Open government data: Out of the box. `https://www.economist.com/news/international/21678833-open-data-revolution-has-not-lived-up-expectations-it-only-getting`, last accessed: 31 July 2017.

The Economist (25 Feb 2017). The age of the appacus. in fintech, China shows the way. `http://www.economist.com/news/finance-and-economics/21717393-advanced-technology-backward-banks-and-soaring-wealth-make-china-leader`, last accessed: 31 July 2017.

The Economist (25 June 2016). The return of the machinery question (special report on artificial intelligence). `http://www.economist.com/news/special-report/21700761-after-many-false-starts-artificial-intelligence-has-taken-will-it-cause-mass`, last accessed: 31 July 2017.

The World Bank (2016). Commodity Markets. `http://beta.worldbank.org/en/research/commodity-markets/`, last accessed 12 April 2016.

Timmermann, A. (2006). *Forecast Combinations*, volume 1 of *Handbook of Economic Forecasting*, chapter 4, pages 135–196. Elsevier.

Toole, J. L. e. a. (2015). Tracking employment shocks using mobile phone data. *Journal of The Royal Society Interface*, 12(107):20150185.

van der Maaten, L. and Hinton, G. (2008). Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

van der Vaart, A., Sandrine, D., and van der, L. (2006). Oracle inequalities for multi-fold cross validation. *Statistics & Risk Modeling*, 24(3):351–371.

Varian, H. (2014). Big data: New tricks for econometrics. *Journal of Economic Perspectives*, 28(2):3–28.

Wager, S. and Athey, S. (2015). Estimation and Inference of Heterogeneous Treatment Effects using Random Forests. *ArXiv e-prints*. 1510.04342.

Zeileis, A. e. A. (2002). strucchange: An R package for testing for structural change in linear regression models. *Journal of Statistical Software*, 7(2):1–38.

Zimek, A., Schubert, E., and Kriegel, H. P. (2012). A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining*, 5(5):363–387.

Appendix: Model cheat sheet

model class	S/US	reg/class	parametric	data size	norm	suited for	advantages	disadvantages
OLS	S	reg	yes	small - large	no	simple relations, hypothesis testing	interpretability, computability	inflexibility
Logit	S	class	yes	small - large	no	simple relations, hypothesis testing	interpretability, computability	inflexibility
naïve Bayes	S	class	no	small - large	no	simple benchmark	computability	independence assumption
k-NN	S	reg/class	no	small - medium	yes	clustered data, multiple regimes	flexibility	interpretability, COD
tree model	S	reg/class	no	small - large	no	complex relations, multiple regimes	flexibility, interpretability, computability	greedy, over-fitting
random forest	S	reg/class	no	small - medium	no	complex relations, multiple regimes	flexibility	computability, interpretability
artificial neural network (ANN)	S	reg/class	semi	mid - large	yes	complex relations, multiple scales	flexibility	computability, over-fitting, data hungry, interpretability
support vector machine (SVM)	S	reg/class	no	small - medium	yes	complex relations	flexibility, computability	over-fitting, interpretability
k-means	US	- -	no	small - large	yes	feature extraction, stylised facts, structure	interpretability	interpretability, COD
hierarchical clustering analysis (HCA)	US	- -	no	small - large	yes	feature extraction, stylised facts, structure	interpretability	interpretability, COD

Table 12: Machine learning model overview. Notes: S/US: supervised/unsupervised learning. Reg/class: Applicable to regression or classification problems. Semi-parametric (ANN) refers to the property that the parametric form (specification) of the model is pre-specified but weights within the network may not contribute to the final output effectively changing the model form. Data range refers to the usable or recommended size of the dataset. The choice of a dataset and the suitability of a model depend on the situation, but also the available computational resources. Feature normalisation can be applied to all models but may lead to a reduction of interpretability. Failing to do so is likely to impair model performance in cases where a model exploits the clustering of data points, i.e. where distance measures are required, or where different features interact. Suitability refers to the kind of problems or data properties a model may be particularly apt for. For instance, tree-based models are expected to perform well if the data describe different regimes with clear transitions between them. Advantages and disadvantages refer to the advantages a model may have over others or the kind of limitations one may encounter, respectively. Again, computational requirements and performance depend on the problem, the model, the data, the available hardware hard and software, as well as the implementation. The curse of dimensionality (COD) refers to the loss of structure in high-dimensional spaces. Interpretability may be advantage or disadvantage for clustering techniques, in the sense that a pronounced clustering structure may directly speak to stylised facts about the data or through up new questions. The performance of all supervised models may be improved through calibration using meta-algorithm, like regularisation or bagging.