Improved Mortality Forecasts Using Artificial Intelligence

Author: Rune Langergaard

Supervisor: Yunus Emre Ergemen

Number of characters: *105,610 characters (44 pages)*

Graphs/tables: *15 pages*

Total number of pages: *59 pages*

Department: *Department of Economics and Business Economics*

Danish title: *Forbedrede dødelighedsforudsigelser ved brug af kunstig intelligens*

Abstract

The Lee-Carter model has for a long-time been the standard model used in mortality forecasting. This thesis proposes and extensively test deep learning models and a multi-population factor model to overcome the limitations of the Lee-Carter model and its extensions. The main limitation of the Lee-Carter model is the inability to utilize additional variable information and extrapolate from the information of other populations, as it is fit to a single population. Furthermore, it is a linear forecasting model, which can pose problems during special occurrences and when the historical trend is not linear. Although extensions of the Lee-Carter model to multiple populations exist, these are often hard to calibrate without specific domain knowledge. Therefore a deep neural network is proposed to overcome the limitations mentioned above. Deep learning models are representational learning models, which have the ability to uncover the hidden relations of the data. Data representations are learned through multiple non-linear transformations of the data, that are optimized with respect to a predictive loss-function. Deep learning models can additionally handle multiple populations and automatically select the optimal model structure. The deep learning model is comprised of a neural network with several hidden layers.

Furthermore, temporal additions of the deep learning model are tested, which are specialized for time-series data. These recurrent models are proposed to track the temporal nature of the mortality data and model the time-dependence between observations. Furthermore, a set of simpler machine learning models are also tested to observe the importance of learning the hidden factors for representations. The models are compared through formal testing using model confidence set tests on all populations and ages. The tests show that model performance depends on the different age-groups. The deep learning models are found to outperform the classical stochastic factor models, and the standard deep feed-forward neural network is found to be the highest performing model. The multi-level factor model shows a slight advantage over the single-population models, having a more robust performance. This indicates an advantage in the ability to extrapolate from the information of multiple populations. The proposed models are also used for mortality projections. The projections show a greater convergence in mortality rates between males and females when using the deep learning models rather than the Lee-Carter model for mortality projections. This implicates a more considerable degree of fairness for unified pension schemes. A lower projected mortality rate implicates the need to increase the retirement age to keep a constant payout period scheme as many countries have. Therefore, it has significant implications for policy decisions, insurance pricing, and risk that a more precise and robust model is used for mortality forecasting.

Contents

1 Introduction

Future mortality rates for different age-groups poses an important economic question for society and social planning. It bears great significance for a nation's inter-generational resource transfer and socioeconomic demographic policy decisions (Lee and Carter, 1992; Deprez et al., 2017). Forecasts of mortality rates are used as inputs in actuarial calculations of pricing life insurance products (Levantesi and Pizzorusso, 2019). Furthermore, the forecasts also remain very important for pensions, that calculate the monthly payouts based on life expectancy when people retire. The link between retirement age and retirement payouts with forecasted life expectancy makes a precise value essential (Stoeldraijer et al., 2018). In Denmark, the allowed pension age is determined by the life expectancy of the population (Stoeldraijer et al., 2013; Kjærgaard et al., 2019a). Mortality rates are known to vary across age-groups and time, which makes models that can precisely predict future mortality rates important within both demography and actuarial science (Richman and Wüthrich, 2018). Mortality predictions also play an essential role for healthcare providers and social services in general (Kjærgaard et al., 2019b; Brouhns et al., 2002a). Mortality forecasts help to predict the durability of current pension schemes (Janssen, 2018). For these reasons, models used for mortality forecasting must be accurate (Booth and Tickle, 2008; Booth, 2006).

The Lee-Carter model presented by Lee and Carter (1992) (hereafter referred to as the LC model) has long been the benchmark method for mortality forecasting and has been the inspiration to a lot of extensions and variations (Booth and Tickle, 2008; Janssen, 2018). The LC model decomposes log-mortality rate effects into a time-specific effect and an age-specific parameter coefficient, using singular value decomposition (SVD). Mortality is then forecasted using a standard time-series forecasting method, where specifically the random walk with drift is generally used (Levantesi and Pizzorusso, 2019; Lee and Carter, 1992). The LC model is a linear single population model that studies each population individually without any dependency assumptions amongst the different populations. The LC model performs well when past trends have been linear (Booth and Tickle, 2008). Several models have proposed extensions of the LC model, such as including a cohort (birth year) effect, that accounts for both age and time (Plat, 2009; Renshaw and Haberman, 2006; Cairns et al., 2009; Reither et al., 2011).

Many of the currently used models are called single population models as they model a single population, e.g., a specific country and gender. This is a big limitation of the model. Sub-populations inside each country (e.g., genders) share much of the same trends. Accounting for these dependencies can provide extra information to the models and make forecasts more precise and robust. Furthermore, countries with similar welfare structures also share the same tendencies, such as medical advances. Utilizing this information can increase model forecasting performance. The main idea behind the stochastic factor multi-population models is that you can utilize population dependencies when they do not diverge, to capture a common trend, while still capturing population-specific trends as well (Enchev et al., 2017). The multi-population models can thereby increase mortality forecasting performance when trends in similar populations are used for model identification. A popular multi-population model is the Li-Lee model proposed by Li and Lee (2005) that captures a common and a population-specific effect using SVD estimation. The multi-population models have been found to outperform the single-population models when used on non-diverging populations (Shair et al., 2017; Richman and Wüthrich, 2018; Li et al., 2017). The disadvantage of the multi-population models is that they require specialized knowledge about the populations in the sample. It requires that one knows which populations have similar trends and which are diverging from each other. Failing to recognize heterogeneity between groups can limit the use of extrapolation between populations, and it can lead to large forecasting errors (Booth and Tickle, 2008). Both single-population and multi-population models can run into problems when past trends are non-linear.

During the twentieth-century mortality has declined in many countries, as there have been reductions in mortality from infectious, cardio-circulatory, and cancer diseases (Levantesi and Pizzorusso, 2019). Recent research has shown that mortality in later years has been stagnating (Kallestrup-Lamb et al., 2019; Case and Deaton, 2015). This could indicate that the popular linear models, such as the LC model, could fail to incorporate such irregularities in the trend. Stoeldraijer et al. (2013) points to several studies that have found non-linear mortality trends. Furthermore, as noted by Deprez et al. (2017), the LC model fails to handle special events such as the pandemic in the early twentieth century. Therefore, it can be an advantage to use a non-linear model, especially in times where the declining mortality trend found in the latter half of the twentieth century is no longer observed, or when a special situation occurs, such as the current global coronavirus pandemic. Such methods could be to include cohort effects, additional explanatory variables, or trends from other countries. One approach is to use machine learning methods, which are often good at finding correlations and patterns in the data. Recent advances have been made using statistical machine learning techniques in mortality forecasting. Deprez et al. (2017); Levantesi and Pizzorusso (2019) finds improved performance of the LC model by using machine learning to improve the fit of the model. Richman and Wüthrich (2018) used a deep neural network as a stand-alone model to make more precise mortality forecasts than both single and multi-population models. Mortality forecasting models need to have high representational learning abilities to perform well, where models as the LC model use SVD to capture linear correlations through variance in the data. Many machine learning models can capture non-linear correlations by making non-linear partitions or transformations of the data.

Machine learning methods use data to optimize parameters using the accumulated knowledge it has learned from the data. Machine learning can be useful in detecting patterns, even unknown and unidentifiable ones, as well as hidden correlations. Multiple algorithms can model the non-linear patterns found in the data (Levantesi and Pizzorusso, 2019). Levantesi and Pizzorusso (2019); Shickel et al. (2017) points out that machine learning is yet to become more prevalent in demography, which is due to the fact that the more complex methods often lack interpretability. Many advances have in recent years, been made in making machine and deep learning models more interpretable (Lisboa, 2013; Doshi-Velez and Kim, 2017; Du et al., 2019; Adadi and Berrada, 2018; Shickel et al., 2017; Molnar, 2019). This permits the use of advanced flexible machine learning methods, such as neural networks, while still maintaining high model interpretability. Thus, while Deprez et al. (2017); Levantesi and Nigri (2019); Levantesi and Pizzorusso (2019); Hainaut (2018) have improved performance of the classic stochastic mortality forecasting models, by using machine learning, in the future these machine learning models will be able to stand on their own to make improved mortality forecasts. Research has found that using additional information could improve current mortality forecasting. With enormous amounts of healthcare data produced every day, machine learning could be the driving force in health, actuarial, and demography sciences in the future. The US alone produced around 164 billion gigabytes of healthcare data in 2013 and is expected to produce fifteen times this amount in 2020 (Stanford, 2017). To utilize all the extra information and data, methods that can handle big data is needed. As pointed out by Davenport and Kalakota (2019), the field of health stands to benefit largely by combining the big data amounts with artificial intelligence and machine learning methods. The Danish government takes the same stance in their new national strategy for the use of artificial intelligence, pointing out that the health sector will be one of the key sectors that stand to benefit the most of using machine learning (Regeringen et al., 2019). Therefore, the advanced deep learning methods can make improved mortality forecasts over the strong baseline stochastic factor models, as the deep learning models can utilize more information, handle non-linearities, and have strong representational learning abilities. Furthermore, Janssen (2018) points out that several advances have been made where additional health information has been used, such as smoking and obesity data, to make more precise mortality forecasts. Kjærgaard et al. (2019a) notes that additional information can be useful, but it can be hard to use with the current

models, as these features also must be forecasted. The task of forecasting these additional variables, such as health indices, GDP, smoking prevalence, and other relevant factors, is more complicated than forecasting the mortality patterns themselves. This additional information can be modeled into the deep learning models, which is an additional upside to using these more advanced models. Deep learning models also have disadvantages, and amongst the biggest is the need for large amounts of data to get high performance. In this thesis, the deep learning models outperform the baseline stochastic factor models and the standard statistical machine learning models, showing great promise for future use. The ability to outperform the baseline models using the currently available mortality data, indicates a large potential for future use, as more data and information becomes available.

The remainder of the thesis is, therefore, organized as follows. Section 2 present the mortality forecasting problem. In section 3 the baseline LC model is presented, together with variations and extensions. The idea of using information from multiple populations in stochastic factor models is introduced in section 4. Then machine learning models are introduced in section 5 to use both information from multiple populations and focus more on the forecasting problem. Then deep learning models are introduced extensively in section 6, which are the primary models proposed to increase mortality forecasting performance. Here the basic structure of deep learning models is explained and various extensions that could potentially improve model performance further. Section 7 introduces the traditional loss functions used for similar problems, together with a formal test for comparing model performance. In section 8, an extensive empirical study is performed using data from twenty-three countries, and model performance is compared. Section 9 concludes the thesis and points towards future research and potential use for the proposed models in this thesis.

2 Mortality modeling and forecasting

Mortality modeling dates back to Gompertz law of mortality that was published in 1825 (Willemse and Koppelaar, 2000). Gompertz observed that the mortality rate, when plotted on a log scale, was approximately linear in age over most of adult life (Currie, 2016). Mortality forecasting is a newer subject that dates only a few decades back, and have only recently become more popular after the stochastic models were introduced, namely the Lee-Carter model proposed by Lee and Carter (1992). Up until the introduction of these methods, mortality forecasting involved a large degree of subjective judgment. In general, as pointed out by Booth and Tickle (2008), there are three main classes of mortality forecasting models.

The most popular class of mortality forecasting models is the extrapolative class of models that extrapolate from patterns found in age and time (Haberman and Renshaw, 2011). Extrapolative methods assume that future trends essentially follow the past historical trends (Booth and Tickle, 2008; Stoeldraijer et al., 2013). One of the strengths of the extrapolative methods is the robustness and accuracy they possess when the age-specific log mortality rates have a linear trend (Booth et al., 2006). The second class of models is the explanative models that use certain epidemiological models of mortality from specific causes of death. For these models, the key explanative variables are known and can be measured. An example of such a model is the explanative relationship between smoking and lung cancer. The third class of models is the expectation models that depend on subjective opinions from experts expectation about future mortality, with varying degrees of formality (Booth and Tickle, 2008). A given model can be a mixture of the classes (Stoeldraijer et al., 2013). The choice of class often depends on such criteria as data availability, projection period, and historical mortality trends. The extrapolative models require the most data to perform well, as there is less subjective opinion involved (Stoeldraijer et al., 2013).

In most applications, the sex-specific mortality rates (probabilities) is the primary interest. When the number of deaths is of interest, then these can also be estimated from the forecasted mortality rates, as is suggested in Booth (2006). Furthermore, one can study the overall mortality, or it can be further split into cause-of-death. One can further disaggregate the results into other cohorts, such as spatial or socioeconomic groups (Booth and Tickle, 2008).

Besides from the target of interest, one must also make sure to use an appropriate input period into the given forecasting model. Stoeldraijer et al. (2013) notes that the estimation period primarily is the main determinant for differences in outcomes, especially when there are non-linearities in the trend. Denmark has a history of having less linear time trends for women. Therefore often short historical input periods are used to account for short-term trends. This can reduce the near future forecasting error, as only the recent trends are accounted for in the model estimation. As mentioned by both Renshaw and Haberman (2006) and Haberman and Renshaw (2011), there has generally been a downward trend in mortality from the 1950s. A reason for the recent stagnating trend, seen in Denmark is according to European-Commission et al. (2017) that the Danish population has a longer life expectancy than a decade ago, but there are some noticeable risk factors. Alcohol consumption is a significant risk factor in Denmark, and obesity is also rising. A gender gap in alcohol consumption is present, where 47% of men reported heavy alcohol consumption compared with 28% of women reporting the same. Cancer and cardiovascular diseases are two of the leading causes of death in Denmark, and the mortality rate due to these diseases is the fourth highest in the EU. The number of deaths related to Alzheimer's has more than doubled, reflecting an aging population. Denmark, on the other hand, displays a low socioeconomic gap in reported health. In Indicators (2014) it was found that the life expectancy in the US was generally improving slower than most other OECD countries, compared to their health spending. Although higher health spending per capita is associated with lower mortality rates, it is not always the case. One of the explaining factors, in this case, is the high obesity rates in the US. These findings, coupled together with the current global coronavirus pandemic, indicate a need to accommodate these trends, as they differ from the historical past that is generally observed after the 1950s. This can be done through data transformation and using advanced models that can capture non-linearities.

This thesis use data from The Human Mortality Database (Wilmoth and Shkolnikov) for all countries that have data available for the years 1950 - 2016. Here data is used for females, males, and the total population for each of the countries. Furthermore, each of the models will utilize as much information as possible. Single population models will only be able to use information from a given country and sub-group within the country. In contrast, multi-population models can use data for the same sub-group in multiple countries, and machine learning models can use all data simultaneously. Data for the years 1950 - 1999 is used as training data to estimate the models and 2000 - 2016 is used as test data for out-of-sample forecasting. Results are based on the test data that is unseen for the models. Twenty-three countries satisfy the data requirements with three subgroups in each country, giving a total of sixty-nine populations. Furthermore, mortality rates for each age for the ages of 50 - 95 are modeled for each of the populations. For the target, the mortality rates (or central death rates) are used, which are defined as

$$m_{x,t} = \frac{\text{Number of deaths during calendar year } t \text{ aged } x}{\text{Average population during calendar year } t \text{ aged } x}. \tag{1}$$

Given the assumption that the force of mortality remains constant over each year for each age, then it is equal to the central mortality rate $m_{x,t} = \mu_{x,t}$. The force of mortality $\mu_{x,t}$ is the instantaneous death rate at exact time t for individuals aged x at time t (Cairns et al., 2009). Most models use the central mortality rate as the target, such as the LC and RH model (Lee and Carter, 1992; Renshaw and Haberman, 2006), and refer to it as the mortality rate. The central mortality rate is modeled as the target for all models in this thesis, as the comparison is made with the baseline LC model. The target $m_{x,t}$ will be referred to as the mortality rate throughout the thesis. Other papers refer the initial mortality rate $q_{x,t}$ as the mortality rate,

which is the probability that an individual aged exactly x at exact time t will die between t and $t+1$ (Plat, 2009; Cairns et al., 2009). The models suggested by Cairns et al. (2006, 2009), were originally modeled using a binomial assumption about the number of deaths $D_{x,t}$, but can also be modeled using the Poisson assumption that the other models use. Using the binomial assumption will estimate the closely related initial mortality rate $q_{x,t}$, using a logit link function. The assumption of constant force of mortality, can then be used to estimate $m_{x,t} = -log\,(1 - q_{x,t})$. As mentioned, in this thesis the Poisson assumption is maintained, and the log-link function is used to estimate the log-mortality rate $\log\,(m_{x,t})$.

3 Single population models

Single population models are models that are purposefully made for fitting a single population, e.g., the mortality rates of a gender or the total population in a specific country, as is done in this thesis.

These models can be estimated under different error term distributional assumptions. Lee and Carter (1992) assumes that the error terms are normally distributed and uses SVD[1] to find the least-squares solution of the latent factors with a log link function. Brouhns et al. (2002a) instead assumes that the number of deaths is Poisson distributed and uses this to estimate the model with maximum-likelihood. It is also possible to assume that the number of deaths follow a binomial distribution and minimize the binomial deviance with a logit link function (Haberman and Renshaw, 2011). In this thesis, the single population models are fitted under the same premise that the number of deaths follows a Poisson distribution, and a log link function to the mortality rates is used. The LC model proposed by Lee and Carter (1992) with Gaussian error terms and SVD is also estimated to compare performance against the GLM[2] version proposed by Brouhns et al. (2002a).

3.1 Lee-Carter Model

The LC model proposed by Lee and Carter (1992) is the most popular extrapolative model for mortality forecasting. It does not make any use of knowledge about medical, behavioral, or social influences on the mortality rates. Instead, it uses statistical time series methods to estimate the factors that influence the mortality over time and the corresponding factor loadings. The LC model fits the mortality rates $m_{x,t}$ for a single population with the model

$$\log\,[m_{x,t}] = a_x + b_x k_t + \varepsilon_{x,t}, \tag{2}$$

for appropriately chosen sets of age-specific constants. In this model, x corresponds to a given age and t to a specific year. The parameter b_x tells something about which ages mortality rates decline rapidly and which slowly decline over time. Negative mortality rates are impossible in this model, as the exponential transformation is used to get the mortality rates, which is an advantage when forecasting with the model. The error term $\varepsilon_{x,t}$ has mean 0 and variance σ_ε^2 and is assumed to be Gaussian. It represents the age-specific historical influences not captured by the model (Lee and Carter, 1992).

To estimate the model, Lee and Carter (1992) impose the following constraints,

$$\sum_x b_x = 1, \qquad \sum_t k_t = 0. \tag{3}$$

[1] SVD is shortly expanded upon in appendix E.

[2] A broadened explanation of GLM can be found in appendix D

The parameter a_x is due to these constraints the average of $\log(m_{x,t})$ across all time periods for a specific age x. Without these constraints, then the model will have multiple solutions. As there are no observable regressors in this model, fitting with ordinary least-squares is not possible. Lee and Carter (1992) suggests finding the factor and the related loadings using singular value decomposition (SVD). This finds the least-squares solution when applied to the log mortality rates after the time averages have been subtracted,

$$\log(m_{x,t}) - a_x = b_x k_t. \tag{4}$$

The first right and left vectors and leading value of the SVD, after applying the normalization in eq. (4) using the constraints from eq. (3) will be a unique solution. From the SVD $M = UDV'$ then b_x and k_t is found as,

$$\hat{b}_x = \frac{U_{x,1}}{\sum_x U_{x,1}}, \qquad \hat{k}_t = V_{t,1} D_{1,1} \sum_x U_{x,1} \tag{5}$$

Brouhns et al. (2002a) points out that one of the main disadvantages of the SVD estimation procedure proposed by Lee and Carter (1992) is that it assumes the error terms are homoskedastic. For inference, it is assumed that the errors are normally distributed, which can be unrealistic in the mortality rate forecasting problem. The logarithm of the observed mortality rates is more variable for older ages than at younger ages because of the smaller absolute number of deaths at older ages (Brouhns et al., 2002b).

Brouhns et al. (2002a) extend the LC-model by switching to a generalized linear model (GLM), by changing the random Poisson variation of the number of deaths with an additive error term of the log mortality rates. To overcome the disadvantage of assuming normally distributed errors, Brouhns et al. (2002a) assumes that the number of deaths is a random counting variable, which is Poisson distributed.

$$D_{x,t} \sim Poisson\left(E_{x,t} m_{x,t}\right), \qquad \text{with } m_{x,t} = \exp\left(a_x + b_x k_t\right), \tag{6}$$

where the parameters are still subject to the same constraints as in the Lee and Carter (1992) modeling method.

Instead of using SVD for estimation of a_x, b_x, and k_t, the parameters are determined by maximizing the log-likelihood function of the model in eq. (6). The log-likelihood function is given by

$$L(a, b, k) = \sum_{x,t} D_{x,t}\left(a_x + b_x k_t\right) - E_{x,t} \exp\left(a_x + b_x k_t\right) + constant. \tag{7}$$

Brouhns et al. (2002a) use a numerical Newton step method to maximize the log-likelihood function. Brouhns et al. (2002a) finds improved performance over the SVD estimation process.

To estimate the model parameters from eq. (2), Villegas et al. (2015) suggest rewriting the parameters to

$$(a_x, b_x, k_t) \to \left(a_x + c_1 b_x, \frac{1}{c_2} b_x, c_2\left(k_t - c_1\right)\right) \tag{8}$$

where the given constraints of the LC-model are imposed by letting,

$$c_1 = \frac{1}{n} \sum_t k_t, \qquad c_2 = \sum_x b_x \tag{9}$$

The reparameterization is done because the stochastic models only are identifiable after applying the model constraints, which ensures uniqueness of the parameters. The parameter constraints are applied through a reparameterization constraint function v that maps an arbitrary vector θ of parameters

$$\theta = \left(a_x, b_x^{(1)}, ..., b_x^{(N)}, k_t^{(1)}, ..., k_t^{(N)}, b_x^{(0)}, \gamma_{t-x}\right), \tag{10}$$

into a vector of transformed parameters

$$v\left(\theta\right) = \tilde{\theta} = \left(\tilde{a}_x, \tilde{b}_x^{(1)}, ..., \tilde{b}_x^{(N)}, \tilde{k}_t^{(1)}, ..., \tilde{k}_t^{(N)}, \tilde{b}_x^{(0)}, \tilde{\gamma}_{t-x}\right), \tag{11}$$

which satisfies the model constraints without affecting the output of the model. This allows maximum-likelihood optimization to be performed on the transformed parameter-vector $\tilde{\theta}$ (Villegas et al., 2015).

3.2 Renshaw-Haberman model

The Renshaw-Haberman model (RH-model) proposed by Renshaw and Haberman (2006) generalizes the LC-model by incorporating a cohort-effect. Each cohort is defined as the group of people born at a particular year (found as $t - x$). The RH-model is then given as,

$$\log\left[m_{x,t}\right] = a_x + b_x^{(1)} k_t + b_x^{(0)} \gamma_{t-x}. \tag{12}$$

The rationale for incorporating a cohort comes from observed differences between cohorts and the rate at which their mortality rates decrease (Cairns et al., 2009; Renshaw and Haberman, 2006). The error structure in the model is imposed by specifying the second-moment properties, which allow for multiple error distribution options. The most common for mortality forecasting is the Poisson distribution (Renshaw and Haberman, 2006). To estimate the model, Renshaw and Haberman (2006) assume that the number of deaths follow a Poisson distribution, and use a log link function for the output. Renshaw and Haberman (2006) imposes the following constraints to find a unique solution,

$$\sum_x b_x^{(1)} = 1, \qquad \sum_t k_t = 0, \qquad \sum_x b_x^{(0)} = 1, \qquad \sum_{c=t_1-x_k}^{t_n-x_1} \gamma_c = 0 \tag{13}$$

The parameters are mapped to an output invariant transformation, that incorporates the model constraints,

$$\left(a_x, b_x^{(1)}, k_t, b_x^{(0)}, \gamma_{t-x}\right) \rightarrow \left(a_x + c_1 b_x^{(1)} + c_2 b_x^{(0)}, \frac{1}{c_3} b_x^{(1)}, c_3\left(k_t - c_1\right), \frac{1}{c_4} b_x^{(0)}, c_4\left(\gamma_{t-x} - c_2\right)\right), \tag{14}$$

where $c_1, c_2, c_3, c_4 \neq 0$ are real parameter constraints. The following constant values, will impose the constraints from eq. (13)

$$c_1 = \frac{1}{n}\sum_t k_t, \qquad c_2 = \frac{1}{n+k-1}\sum_{c=t_1-x_k}^{t_n-x_1} \gamma_c, \qquad c_3 = \sum_x b_x^{(1)}, \qquad c_4 = \sum_x b_x^{(0)} \tag{15}$$

Haberman and Renshaw (2011) suggests setting $b_x^{(0)} = 1$, as a simpler model, which gives good performance and resolve stability issues of the original model.

Mortality forecasts of this model are generated using univariate ARIMA processes for k_t and γ_{t-x}, under the assumption of independence between the period effects and the cohort effects (Villegas et al., 2015). This thesis uses the random walk with drift model to forecast the two processes.

3.3 APC Model

The APC model was suggested by Currie (2006) as a simplification of the RH-model, that sets both $b_x^{(1)} = 1$ and $b_x^{(0)} = 1$, such that the model becomes

$$\log\left[m_{x,t}\right] = a_x + k_t + \gamma_{t-x} \tag{16}$$

It was proposed to overcome some of the robustness issues of the RH model (Currie, 2006; Plat, 2009).

To ensure identification, the following constraints are imposed,

$$\sum_t k_t = 0, \qquad \sum_{c=t_1-x_k}^{t_n-x_1} \gamma_c = 0, \qquad \sum_{c=t_1-x_k}^{t_n-x_1} c\gamma_c = 0 \tag{17}$$

The APC estimation can also be made easier by making a parameter transformation, as pointed out by Villegas et al. (2015) following Haberman and Renshaw (2011)

$$(a_x, k_t, \gamma_{t-x}) \to (a_x + \phi_1 - \phi_2 x, k_t + \phi_2 t, \gamma_{t-x} - \phi_1 - \phi_2 (t-x)) \tag{18}$$

$$(a_x, k_t, \gamma_{t-x}) \to (a_x + c_1, k_t - c_1, \gamma_{t-x}). \tag{19}$$

The constants ϕ_1 and ϕ_2 can be found by regressing γ_{t-x} on $t-x$ and imposing the constraints on a_x and γ_{t-x} from eq. (18). Then afterwards the constraint on k_t can be imposed by the transformation in eq. (19) and setting $c_1 = \frac{1}{n}\sum_t k_t$.

3.4 CBD model

Cairns et al. (2006) proposed the CBD model with two age-period terms with pre-specified parameters, such that the model is given as

$$\log\left[m_{x,t}\right] = b_x^{(1)} k_t^{(1)} + b_x^{(2)} k_t^{(2)}. \tag{20}$$

The parameters are then fixed to $b_x^{(1)} = 1$ and $b_x^{(2)} = (x - \bar{x})$, where $\bar{x}$ is the average age in the data. This means that the final model becomes,

$$\log\left[m_{x,t}\right] = k_t^{(1)} + (x - \bar{x})\, k_t^{(2)} \tag{21}$$

In this model, there are no identifiability issues and, thereby, no model constraints.

Cairns et al. (2009) suggested several extensions of the CBD model, specifically with older ages in mind (Haberman and Renshaw, 2011).

3.4.1 M7 extension

Cairns et al. (2009) suggest extending the CBD model by adding a cohort effect and a quadratic age effect. The extension was proposed because of the observed curvature in US data. The M7 extension of the CBD model is given as

$$\log\left[m_{x,t}\right] = k_t^{(1)} + (x - \bar{x})\, k_t^{(2)} + \left((x - \bar{x}) - \hat{\sigma}_x^2\right) k_t^{(3)} + \gamma_{t-x}, \tag{22}$$

where $\hat{\sigma}_x^2$ is the average value of $(x-\bar{x})^2$. Cairns et al. (2009) suggest constraints for identifiability,

$$\sum_{c=t_1-x_k}^{t_n-x_1} \gamma_c = 0, \qquad \sum_{c=t_1-x_k}^{t_n-x_1} c\gamma_c = 0, \qquad \sum_{c=t_1-x_k}^{t_n-x_1} c^2\gamma_c = 0, \tag{23}$$

which will ensure that the cohort effect fluctuates around zero (Villegas et al., 2015). The output of the model is invariant to the transformation,

$$\left(k_t^{(1)}, k_t^{(2)}, k_t^{(3)}, \gamma_{t-x}\right) \rightarrow \begin{pmatrix} k_t^{(1)} + \phi_1 + \phi_2\left(t-\bar{x}\right) + \phi_3\left(\left(t-\bar{x}\right) + \hat{\sigma}_x^2\right), k_t^{(2)} - \phi_2 - 2\phi_3\left(t-\bar{x}\right), \\ k_t^{(3)} + \phi_3, \gamma_{t-x} - \phi_1 - \phi_2\left(t-x\right) - \phi_3\left(t-x\right)^2 \end{pmatrix}, \tag{24}$$

where the constants ϕ_1, ϕ_2, and ϕ_3 can be found from the regression,

$$\gamma_{t-x} = \phi_1 + \phi_2\left(t-x\right) + \phi_3\left(t-x\right)^2 + \epsilon_{t-x}, \qquad \epsilon_{t-x} \overset{i.i.d.}{\sim} N\left(0, \sigma^2\right). \tag{25}$$

Then the constraints can be imposed by using the parameter transformation from eq. (24) (Villegas et al., 2015).

3.4.2 M6 and M8 extension

Cairns et al. (2009) also suggested more simple additions to the CBD model, the M6 model,

$$\log\left[m_{x,t}\right] = k_t^{(1)} + \left(x-\bar{x}\right)k_t^{(2)} + \gamma_{t-x}, \tag{26}$$

and the M8 model,

$$\log\left[m_{x,t}\right] = k_t^{(1)} + \left(x-\bar{x}\right)k_t^{(2)} + \left(x_c - x\right)\gamma_{t-x}. \tag{27}$$

The parameter x_c is a constant that needs to be estimated (Villegas et al., 2015). These two models were suggested by Cairns et al. (2009) to allow the CBD model to take the cohort effect into account. Cairns et al. (2009) suggests the M8 model from the experience of fitting the CBD model. It was found that the cohort effect diminishes over time, which means $\beta_x^{(3)} = \left(x_c - x\right)$ is decreasing over time, instead of leaving it constant as in the M6 model in eq. (26). Cairns et al. (2009) also found that the CBD model is not as robust. The M6 and M8 model were found to be more robust. However, the M8 model can be slow in convergence.

3.5 Plat model

Plat (2009) expand upon the CBD-model that was proposed by Cairns et al. (2006) with a model that has three age-period effects and a static intercept as in the LC-model. The Plat model becomes,

$$\log\left[m_{x,t}\right] = a_x + k_t^{(1)} + \left(\bar{x}-x\right)k_t^{(2)} + \left(\bar{x}-x\right)^+ k_t^{(3)} + \gamma_{t-x}, \tag{28}$$

where $\left(\bar{x}-x\right)^+ = \max\left(0, \bar{x}-x\right)$. The a_x parameter, ensures that the basic shape of the mortality curve over ages fit with the historical observations as in the LC model. Whereas, $k_t^{(1)}$ represents changes in the level of mortality for all ages. This factor should not be mean reverting, because it is not expected that higher mortality improvements in some years are compensated by lower mortality improvements in other years (Plat, 2009). The factor $k_t^{(2)}$ allows changes in mortality to vary between ages, to reflect the historical observation that improvement rates can be different between ages. The last factor $k_t^{(3)}$ is added

because historical data indicate different dynamics of mortality for lower ages (Plat, 2009). The two extra factors $k_t^{(2)}$ and $k_t^{(3)}$ allow the model to have non-trivial correlation structures between ages.

The Plat model imposes the following constraints to ensure identifiability,

$$\sum_t k_t^{(1)} = 0, \quad \sum_t k_t^{(2)} = 0, \quad \sum_t k_t^{(3)} = 0, \quad \sum_{c=t_1-x_k}^{t_n-x_1} \gamma_c = 0, \quad \sum_{c=t_1-x_k}^{t_n-x_1} c\gamma_c = 0, \quad \sum_{c=t_1-x_k}^{t_n-x_1} c^2\gamma_c = 0 \tag{29}$$

The first three constraints ensure that the period factors are centered around zero. The last three constraints ensure that the cohort effect fluctuates around zero, and that it has no linear or quadratic trend.

Much like the other models, Villegas et al. (2015) suggests using the two parameter transformations, that the output is invariant to, which follows the method of Haberman and Renshaw (2011),

$$\left(a_x, k_t^{(1)}, k_t^{(2)}, k_t^{(3)}, \gamma_{t-x}\right) \rightarrow \begin{pmatrix} a_x + \phi_1 - \phi_2 x + \phi_3 x^2, k_t^{(1)} + \phi_2 t + \phi_3\left(t^2 - 2\bar{x}t\right), \\ k_t^{(2)} + 2\phi_3 t, k_t^{(3)}, \gamma_{t-x} - \phi_1 - \phi_2\left(t-x\right) - \phi_3\left(t-x\right)^2 \end{pmatrix} \tag{30}$$

$$\left(a_x, k_t^{(1)}, k_t^{(2)}, k_t^{(3)}, \gamma_{t-x}\right) \rightarrow \left(a_x + c_1 + c_2\left(\bar{x}-x\right) + c_3\left(\bar{x}-x\right)^+, k_t^{(1)} - c_1, k_t^{(2)} - c_2, k_t^{(3)} - c_3, \gamma_{t-x}.\right) \tag{31}$$

Villegas et al. (2015) finds the constants ϕ_1, ϕ_2, and ϕ_3 from the regression,

$$\gamma_{t-x} = \phi_1 + \phi_2\left(t-x\right) + \phi_3\left(t-x\right)^2 + \epsilon_{t-x}, \qquad \epsilon \overset{i.i.d.}{\sim} N\left(0, \sigma^2\right), \tag{32}$$

and then use the transformation from eq. (30) for the cohort effect parameter γ_{t-x} and transform the period effect factors via the transformation from eq. (31). The constants from eq. (31) will be,

$$c_i = \frac{1}{n}\sum_t k_t^{(i)}, \qquad i = 1, 2, 3. \tag{33}$$

Plat (2009) suggest using a simplified model, when only older ages are modeled,

$$\log\left[m_{x,t}\right] = a_x + k_t^{(1)} + \left(\bar{x} - x\right)k_t^{(2)} + \gamma_{t-x} \tag{34}$$

This model use the same transformations as the above Plat model, with the omission of the constraints involving $k_t^{(3)}$ and c_3 Villegas et al. (2015). This thesis uses the original Plat model from eq. (28).

4 Multi population models

Extending the single population factor models involve adding terms that are formed by combinations of pooled mortality rates from multiple populations, and terms that are specific to the individual populations (Richman and Wüthrich, 2018). It is natural to believe that countries' mortality trends are correlated. Whenever mortality rates decline in a country due to improvements, often these improvements spread to other countries (Enchev et al., 2017). Utilizing extra information should increase model performance. Most of the popular multi-population models are not able to handle divergences between populations, which pose a challenge for estimation of the models. This makes knowledge about the countries a requirement for optimal performance. It is a disadvantage in using the multi-population models (Richman and Wüthrich, 2018; Enchev et al.,

2017). Richman and Wüthrich (2018) observed that no general theory has been able to explain why the multi-population specification is more or less optimal than their single population counterparts. In this thesis, the multi-level factor model proposed by Choi et al. (2018) is used to overcome the task of determining non-diverging country groupings. First, the popular Li-Lee extension of the LC model proposed by Li and Lee (2005) is presented. Afterward, the multi-level factor model is introduced and explained.

4.1 Li-Lee model

Li and Lee (2005) wanted to use the fact that mortality patterns and trajectories in closely related countries are similar and unlikely to diverge in the long-run. This is believed to improve mortality forecasts for groups of countries. Li and Lee (2005) found that single population models are likely to have increasing divergence, even though the countries seem to follow similar mortality trends. Li and Lee (2005) therefore, proposed the Li-Lee model, which has a common factor for the group of countries in the model and an individual country trend that allows countries to differ in the short run. To avoid the forecast divergence between similar countries when using the LC model proposed by Lee and Carter (1992), the Li-Lee model includes the common factor and let the country-specific factor go to zero in the long-run. This ensures convergence between similar populations used in the same model.

The Li-Lee model is given as,

$$\log\left[m_{x,t,i}\right] = a_{x,i} + B_x K_t + b_{x,i} k_{t,i} + \varepsilon_{x,t,i}, \qquad \text{for } 0 \leq t \leq T. \tag{35}$$

The common factors B_x and K_t is obtained from applying the SVD procedure from Lee and Carter (1992) on all the multiple populations at once. This is done by applying the SVD procedure on the average log mortality rates of the populations. The intercept $a_{x,i}$ is estimated individually for each population as the intercept does not cause long-run divergence. Using only the common factor in the Li-Lee model would keep constant age-specific death ratios between the populations in the model. In forecasting, there would be neither divergence nor convergence. After finding the intercepts $\hat{a}_{x,i}$ and the common factor $\hat{K}_t$ and its factor loadings $\hat{B}_x$ using SVD, then the population-specific factor $\hat{k}_{t,i}$ is found by applying SVD on the residuals

$$\log\left[m_{x,t,i}\right] - \hat{a}_{x,i} - \hat{B}_x \hat{K}_t. \tag{36}$$

This means that the population-specific factor is obtained from the first-order vectors of the SVD applied to the residual matrix (Li and Lee, 2005). The vector $b_{x,i}$ describes differences between the patterns of change by age in mortality for the i'th population and the group as a whole. Li and Lee (2005) notes that populations can be modeled as a group when they have similar socioeconomic conditions and close connections that are expected to continue in the long-run, which requires subjective judgment of the populations. Thus, the included populations should not diverge in the long run in the Li-Lee model.

4.2 Multi-level factor model

The purpose of the multi-level factor model proposed by Choi et al. (2018) is to develop a model that contains both global and country-level factors. Choi et al. (2018) develop a sequential procedure that identifies the global and country factors separately. In the first step, the global factors are estimated using canonical correlation analysis (CCA). Although the initial estimator only uses data from two populations, it consistently estimates the global factor after the sequential procedure. Using

the initial estimator obtained from CCA, then the country factors can be estimated using SVD. Hereafter, the global factors are re-estimated with SVD using the country factors. The country factors are also re-estimated. This procedure is continued until convergence. The multi-level factor model is given as,

$$y_{x,i,t} = \gamma'_{x,i} G_t + \lambda'_{x,i} F_{i,t} + e_{x,i,t}, \tag{37}$$

where x, in this case, is the age, t is the time and i is the i'th population. The output is here given as $y_{x,i,t} = \log[m_{x,i,t}]$. This model differs from the Li-Lee model in that the factor loadings for the global factor is country specific, as well as the possibility of containing multiple country-specific factors. Furthermore, the estimation technique also differ between the two models. The model is flexible in that it allows each population to have a varying number of ages $x = x_1, ..., x_{N_m}$. Choi et al. (2018) assumes the following about the global and country factors

(i) $\{G_t\}$, $\{F_{1t}\}$,..., $\{F_{Mt}\}$ are zero-mean, stationary-processes that satisfy the conditions for the law of large numbers and central limit theorem.

(ii) $\{G_t\}$, $\{F_{1t}\}$,..., $\{F_{Mt}\}$ are uncorrelated. That is, $E\left[F_{i,t}F'_{m,t}\right] = 0$ for all t and $i \neq m$ and $E\left[G_t F'_{i,t}\right] = 0$ for all t and i.

(iii) $\{G_t\}$, $\{F_{1t}\}$,..., $\{F_{Mt}\}$ satisfy conditions for the law of large numbers and the central limit theorem applied to their self- and cross-products.

(iv) $E\left[e_i e'_m\right] = 0$ for every i and m with $i \neq m$.

(v) $N_1, ..., N_M$ and $N = N_1 + ... + N_M$ are of the same order of magnitude.

Without assuming that the factors are uncorrelated, then identification of the global factors using CCA would not be possible. Assumption (iv) implies that the error terms of each population used in the model are uncorrelated. The last assumption means that no population can have a dominantly large or small number of individuals (in this case age groups). The model also requires that the number of populations M is finite.

Model estimation is as described earlier set-up in the following way (superscript is used to denote the iteration number).

1. Select two countries and obtain the global factor estimator $\hat{G}_t^{(1)}$ using CCA.
2. Find the respective factor loadings $\hat{\gamma}_{x,i}^{(1)} = \left(G^{(1)\prime}G^{(1)}\right)^{-1} G^{(1)\prime}Y_i$ for the global factor using data for each of the populations $i = 1, ..., M$. The resulting vector will be of dimensions $(N_i \times 1)$ where N_i is the number of age-groups for population i.
3. Estimate the country factors $\hat{F}_{i,t}^{(1)}$ and loadings $\lambda_{x,i}^{(1)}$ using SVD on the residuals $Y_{x,i,t} - \hat{\gamma}_{x,i}^{(1)\prime}\hat{G}_t^{(1)}$.

Choi et al. (2018) suggest using the two countries that yield the maximum sample mean of their eigenvalues in step 1. For all subsequent iterations $(j = 2, ...)$, step 1 will be replaced with

1. Estimate the global factor using SVD on the residual matrix $Y_{x,i,t} - \hat{\lambda}_{x,i}^{(j)\prime}\hat{F}_{i,t}^{(j)}$, where each populations residual vector is stacked.

Choi et al. (2018) shows that the factors are consistently estimated using this strategy. They also show through a simulation study that the performance of the model is better when the second iteration is included because the global factor will be estimated using more information.

The number of static factors must also be determined and there are several ways of doing this. The multi-level factor model assumes that the number of global factors is predetermined, and the number of country factors is found using information criteria after eliminating the information from the global factor. Choi et al. (2018) consider the following information criteria

- $IC_{p2} = \ln\left(V_i\left(k_i, F_{k_i}^{(1)}\right)\right) + \ln\left(\min\{N_i, T\}\right)\left(\frac{k_i(N_i+T)}{N_i T}\right)$
- $BIC = T \cdot tr\left(\ln\left(\frac{1}{T} Y_i^{G\prime} M_{\tilde{F}_{k_m}^{(1)}} Y_i^G\right)\right) + \ln\left(N_i T\right)\left[k_i\left(N_i + T\right) + N_i\right]$
- $HQ_c = T \cdot tr\left(\ln\left(\frac{1}{T} X_i^{G\prime} M_{\tilde{F}_{k_m}^{(1)}}\right)\right) + c\ln\left(N_i T\right)\left[k_i\left(N_i + T\right) + N_i\right]$

where $V_i\left(k_i, F_{k_i}^{(1)}\right)$ is the sum of squared residuals for country i in step 2 when k_i country factors are estimated by $\tilde{F}_{k_m}^{(1)}$ and $tr\left(\cdot\right)$ is the trace function. In HQ_c there is a constant c that needs to be chosen, which will determine the trade-off between fit and number of parameters. Choi et al. (2018) note that previous studies have found BIC and HQ_c to perform well in finite samples. It is also found that IC_{p2} overestimates the number of factors when there is strong serial correlation. BIC and HQ_c tend to underestimate when the number of cross-section observations N_i is small[3]. BIC will be applied in this thesis, as it is found to perform the best among the information criteria in Choi et al. (2018). The multi-level factor model will be estimated separately for each gender and the total population, using information from all the included countries.

Through testing of the two proposed multi-population models, similar performance was found for the two models. Therefore, only the results of the multi-level factor model are presented in section 8. The multi-level factor model requires less specialized domain knowledge about the countries' welfare systems and in-depth similarities. Even though this specialized knowledge can be beneficial and increase the performance of most models, a model that does not need specialized domain knowledge is usually preferred. In the next section, machine learning models are presented, which can potentially increase forecasting performance, as they have a broader focus on the prediction task, and have the ability to introduce non-linearities in the forecasting.

5 Machine learning models

Machine learning is a field within AI, where the algorithms are allowed to accumulate knowledge learned from estimation data (Richman, 2018; Goodfellow, 2016). Machine learning is broadly classified into three groups: supervised, unsupervised, and reinforcement learning. Supervised learning is the most popular group of models, where a model is trained to predict an output y (dependent variable), using a set of features X. I.e. the goal is to learn $f\left(X\right) = y$. These models span over well-known models such as generalized linear models (GLM), linear regressions, to deep neural networks. Unsupervised learning is the application of machine learning algorithms to find patterns and structures only within the X set of features. A well-known example here is the principal component analysis (PCA). Reinforcement learning is the part of machine learning, where an

[3] $N_i \leq 20$, which is not the case in this application.

agent learns from rewards signals in real-time to improve decision making (Sutton, 1998). The mortality forecasting problem is a supervised learning problem when posed inside the machine learning domain. Next periods log-transformed mortality rates are the targets, and the features are year, age, gender, and current periods log mortality rate. Additionally, cohorts can pose as a feature. Mullainathan and Spiess (2017) summarises supervised machine learning as trying to optimize the following equation

$$\min L\left(f\left(X\right),y\right), \qquad \text{for all } f \in F, \qquad s.t. R\left(f\right) < c. \tag{38}$$

In eq. (38) it states that the goal is to minimize a loss function $L\left(\hat{f}\left(X\right),y\right)$ with respect to a certain model complexity level c. Therefore, machine learning focuses on the problem of prediction, i.e., to produce predictions of y from X. This is a fundamentally different problem than the one solved by many econometric models where the focus is on the parameter estimation, i.e., to produce parameters β that can explain the underlying relationship between y and X. There is a trade-off in parameter interpretability when one uses the more complex models rather than the simple linear regression models. Therefore, if one is willing to trade-off some model interpretability for more complexity and a larger focus on the predictions, there is a potential gain in using flexible machine learning models, such as the ones presented in this section and section 6. The parameters produced in the machine learning models are rarely consistent, and they often introduce bias into the models to lower the variance of the predictions (Mullainathan and Spiess, 2017).

A machine learning model must have very high representational capabilities to succeed in the mortality forecasting problem. The stochastic factor models are good at estimating the year and age mortality-pattern factors. The machine learning models presented in this thesis estimate the hidden patterns using non-linear transformations of the data. Especially the deep learning algorithms of machine learning are known to have very strong representational abilities (Bengio et al., 2013). The machine learning algorithms presented in this section are more dependent on the input variables, where only age, year, country, and sub-population is available, and they might have limited success. To have high performance, the model must have a way of uncovering the more hidden correlations. As explained in section 6, it can be successfully done using neural network embeddings.

5.1 Random Forest (RF)

The random forest model is a tree-based algorithm derived from the bagging method. It uses an ensemble of decision trees made with bootstrap samples. The components of the random forest algorithm are explained below.

5.1.1 Decision trees

Decision trees can be applied to both regression and classification settings. The basic idea is that data is split based on the input features to get homogeneous subgroups. I.e. the predictor space is divided into J distinct and non-overlapping regions $(R_1, ..., R_J)$. Every observation that falls into the same region has the same output prediction. Typically in regression problems, such as the mortality forecasting problem, one would find the split regions using the residual sum of squares (RSS) as a loss function, i.e., the goal is to minimize

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j}\right)^2. \tag{39}$$

More specifically the split variable j and split point s that minimizes

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}\left(y_i-c_1\right)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}\left(y_i-c_2\right)^2\right], \tag{40}$$

where c_1 and c_2 is the prediction in region 1 and region 2, respectively. The prediction in a regression tree is most often given as the mean outcome of the observations in the region. As it is computationally infeasible to study all possible data-splits, a top-down greedy approach is used. For each sequential split, the algorithm only considers the current step and not all potential following steps (James, 2013). Each step chooses the single split of a feature that gives the largest reduction in the RSS. The splitting process is continued until some stopping criterion is met, which usually is that no region contains more than five observations. One can prune the trees using a cost-complexity loss function, which works much like the BIC information criteria, as it is a trade-off between fit and size of the decision-tree.

Comparing decision trees to a classic linear regression, it can be seen as an automatic way of looking at which interaction variables to include. In many problems, it is infeasible to study all possible interactions. Many interactions are often irrelevant to the problem, not providing any additional information. Decision trees are finding the relevant interactions automatically when minimizing the loss function (Mullainathan and Spiess, 2017).

5.1.2 Bootstrap aggregation (bagging)

Because regression trees are very variable and sensitive to the data, they are often used through the bagging technique. Bagging uses the average prediction of many decision trees to reduce the variance of the final prediction. Thus, the essential idea behind bagging is to average many noisy but approximately unbiased models to reduce the variance (Hastie, 2009). Averaging over B identically distributed (i.d.) models with pairwise correlation of ρ and individual variance σ^2 will give a variance of

$$\rho\sigma^2+\frac{1-\rho}{B}\sigma^2. \tag{41}$$

As can be seen from eq. (41), increasing the number of bootstrap samples B makes the second term smaller. It means that the total variance improvement over using decision trees is limited by the pairwise correlation.

Each of the B bootstrap training sets is based on a decision tree, and the final bagging prediction is the average outcome of the individual bootstrap models. Outcomes are calculated for each bootstrap training set $\hat{f}^{*1}(x)$, $\hat{f}^{*2}(x)$, ..., $\hat{f}^{B}(x)$ using B bootstrap samples, and the outcome is given as

$$\hat{f}_{bag}(x)=\frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x). \tag{42}$$

Each decision tree is not pruned in the bagging method, which causes each tree to have a lower bias. The decision trees are often correlated, which makes the effect of lowering the variance smaller. Averaging many highly correlated quantities does not lead to as substantial a variance reduction. The random forest extension to the bagging method is commonly preferred because it lowers the correlation between the decision trees (James, 2013). As mentioned above, and seen in eq. (41), the variance improvement was limited by the correlation between the decision trees, and thus being able to lower this correlation can give a larger variance reduction (Hastie, 2009). Therefore, the goal of the random forest is to lower the pairwise correlation by only allowing a subset of the features at each split, and thereby lowering the variance of the final model.

5.1.3 Random forest algorithm

As in bagging, the random forest algorithm also builds B decision trees using bootstrapped training samples. When building the decision trees, each time a split is considered, only a random sample of the p features is considered as splitting candidates. As a result, this causes the individual decision trees to be more different from each other, and thereby less correlated. A common choice is to choose the number of splitting candidates as $m = \sqrt{p}$ such that only a fraction of the available features is considered at each split (James, 2013). Comparing the random forest algorithm to the ridge regression, which is a regularized version of a linear regression, it can be noted that they are trying to do the same job. A ridge regression will add L2-penalization to linear regressions to lower the variance, at the cost of added bias. Random forest will also try to lower the variance to minimize the bias-variance trade-off, but the variance lowering technique differs from the ridge regression. Ridge regression penalizes high parameter values to lower the variance, whereas the random forest will use model averaging and decorrelation of the ensemble to lower the variance. Decision trees are much like linear regressions a relatively unbiased method but suffer from having high variance. Ridge regression and random forest seek to overcome this disadvantage of these methods. Opposite from the ridge regression, the random forest is a non-parametric method that works with the splitting rules (Mullainathan and Spiess, 2017). Regularization for random forest algorithms can also be provided through the aforementioned size of the trees. Smaller trees will be more robust at the cost of higher bias.

5.2 Boosting

Boosting is another way to improve the prediction results of decision trees. Boosting does not use bootstrap samples. Boosting grows the decision trees sequentially, such that each tree is using information from the previously grown trees. Instead of growing a large decision tree that overfits the training data, boosting will instead use weak learners to fit the data slowly (James, 2013). Thus, the idea is to combine outputs of many weak learners to form a powerful committee. A weak learner is a predictor whose error rate is only slightly better than random guessing, and the common choice is a decision tree with only a few splits. Boosting is a way of fitting an additive model, where each predictor is a regression tree (Hastie, 2009). Each tree is fit to the pseudo residuals of the current model calculated via the gradient. By fitting small trees to the residual, the model slowly improves in areas where it is not performing well. The loss function is minimized using the calculated first-order gradients in the gradient boosting algorithm. Gradient boosting uses gradient descent to optimize directional movement. The gradient for the m'th iteration is given as,

$$-\hat{g}_m(x_i) = -\left[\frac{\partial \hat{R}(f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}^{(m-1)}(x)} = -\left[\frac{\partial L(y_i, f(x))}{\partial f(x_i)}\right]_{f(x)=\hat{f}^{(m-1)}(x)}. \tag{43}$$

The current solution is then updated as,

$$f_m = f_{m-1} - \rho_m \hat{g}_m(x_i), \tag{44}$$

where ρ_m is the learning rate at the m'th iteration. The process is repeated for each iteration. It is a greedy algorithm because the gradient descent is the local direction where the loss is decreasing.

Chen and Guestrin (2016) extended the gradient boosting algorithm to be more scalable, which they called extreme gradient boosting (XGB). XGB allows trees to be of different sizes, and it uses Newton boosting instead of gradient boosting. Tree sizes are determined using standard penalization. It will likely learn better tree structures from doing this (Nielsen, 2016).

Using second-order derivatives with the Newton method provides more information about the direction of the gradient, which is often helpful for minimizing the loss. The XGB algorithm also uses a split finding algorithm that uses the feature quantiles instead of searching through all values (Chen and Guestrin, 2016).

5.3 Support Vector Machines (SVM)

The SVM algorithm is also a non-linear predictor that is an extension of the soft margin algorithm. The soft margin algorithm uses the feature space hyperplane that separates the observations the most while allowing some observations to be on the wrong side of the margin. The algorithm uses a cost function, that defines how much slack is allowed. SVM makes use of non-linear kernel functions to transform the feature space, making the prediction space non-linear. The SVM algorithm for regression problems makes use of a different loss-function where only residuals larger in absolute value than some constant will contribute to the loss (James, 2013). The optimization problem is a convex optimization problem with constraints. The problem can be re-expressed as,

$$\min_{\beta,\beta_0} \|\beta\|^2 + C\sum_{i=1}^{N} \xi_i, \text{ s.t. } \xi_i \geq 0, y_i\left(x_i'\beta + \beta_0\right) > 1 - \xi_i, \tag{45}$$

where C is a parameter that can be tuned, controlling how much slack is allowed. Slack means that some observations are allowed to be on the other side of the separating hyperplane. The procedure is more flexible when the feature space is enlarged using kernels or other basis expansions such as polynomials. This allows a more flexible division of the feature space. A popular choice of kernel function is the radial kernel,

$$K\left(x, x'\right) = \exp\left(-\gamma\|x - x'\|^2\right) \tag{46}$$

In the regression problem

$$H\left(\beta, \beta_0\right) = \sum_{i=1}^{N} V\left(y_i - f\left(x_i\right)\right) + \frac{\lambda}{2}\|\beta\|^2, \tag{47}$$

is minimized, where λ controls how much to penalize large parameter values in order to lower the variance of the model. Here

$$V_\epsilon\left(r\right) = \begin{cases} 0, & \text{if } |r| < \epsilon \\ |r| - \epsilon, & \text{otherwise,} \end{cases} \tag{48}$$

which means error sized less than ϵ are ignored. Using a kernel then the estimation function $f\left(x\right)$ is

$$\hat{f}\left(x\right) = \sum_{i=1}^{N} \hat{a}_i K\left(x, x_i\right). \tag{49}$$

With SVMs, it is the data observations that are wrongly classified that are in the support vector. Only the observations in the support vector are given a positive weight $\hat{a}_i > 0$. Therefore, the estimation mechanism differs from linear regression with regularization, such as the ridge regression or the LASSO regression. In a LASSO regression, it is all observations that are used to make the parameter estimates. In contrast for the support vector regression, it will only be the observations that are in the support vector. Therefore, the difference from a regularized regression is which observations are used for parameter estimation and the ability to transform the input-space into a highly non-linear space using kernel functions (Awad and Khanna, 2015).

5.4 Forecasting strategies

In general, there are two main ways of doing time-series forecasts with machine learning models: the recursive and the direct forecasting method. The recursive forecasting method will set-up a model that is estimated with the output variable being the one-period ahead value. Thus, when multi-step ahead forecasting is performed, input values will be forecasts. If the one-step ahead model is defined as $\hat{f}$, which is only a function of previous output values, then forecasting are given by

$$\hat{y}_{T+h} = \begin{cases} \hat{f}\left(y_T, ..., y_0\right), & \text{if } h = 1 \\ \hat{f}\left(\hat{y}_{T+h-1}, ..., \hat{y}_{T+1}, y_T, ..., y_{h-1}\right) & \text{if } h \in (2, ..., T) \\ \hat{f}\left(\hat{y}_{T+h-1}, ..., \hat{y}_h\right) & \text{if } h > T \end{cases} \tag{50}$$

Therefore, at some point, all input values can be forecast values. The recursive strategy is sensitive to the accumulation of errors with the forecast horizon. Errors will propagate forward because the forecasts are used to determine subsequent forecasts (Taieb et al., 2012a). Direct forecasting specifies a forecasting model for each step-ahead forecasting period. This means that a model will be estimated to do one-step ahead forecasting, a model will be estimated to do two-step ahead forecasting, and so forth. Therefore, h models will be needed for h-step ahead forecasting. The direct forecasting method is, for that reason, more computational as it requires a model for each step-ahead forecasting period, but will also be less biased. Using direct forecasting can lead to irregularities in the forecasting function, as the individual step-ahead models can be quite different (Taieb et al., 2012b). The recursive forecasting method is used in this thesis, such that forecasted values will be used for multi-step ahead forecasts.

The machine learning techniques presented in this section do not have strong representational capabilities when it comes to finding representations of data with latent features, such as is the case with single population mortality forecasting models. The machine learning models can bring non-linearities that can potentially improve performance but lack the ability to learn hidden patterns. Zhong et al. (2016) noted that PCA has long been used for representation learning, but these models are typically linear. Deep learning can potentially improve performance over both the linear models that use PCA and the non-linear machine learning models, as it has strong representational capabilities as well as the possibility to introduce non-linearities. Thus, overcoming the limitations of both the classic LC models and the machine learning models presented in this section. Deep learning is a field within machine learning, that has a broader focus on learning representations. Deep learning methods allow for the use of multiple populations in the same model, such that it can utilize more information and automatically account for population dependencies.

6 Representational learning and deep learning

Models such as the LC model can be seen as linear regression models where the mortality rate is predicted using features (independent variables) that summarizes the relationship between age and time on mortality rates. The LC model does not directly use the original features such as year and age, but make use of the SVD to make a principal components regression to derive factor variables. In some cases, this is too rigid, especially when the mortality rate does not show a linear trend. Therefore, one can use representational learning, which is an approach where algorithms are allowed automatically create the set of optimal features for a problem (Richman and Wüthrich, 2018). Especially deep learning with neural networks is a popular method in representation learning. Neural networks can be used as universal function approximators, that for

example, can capture the LC model and its extensions (Wuthrich and Buser, 2019; Richman and Wüthrich, 2018; Richman, 2018; Kock et al., 2011). The performance of many machine learning models relies heavily on the set of inputs chosen to represent the data. For mortality rate forecasting, this can cause problems for the machine learning models unless they are very flexible at learning representations on their own. Most successful mortality rate models rely on the ability to uncover the latent factors, where many have had success using the above mentioned principal component regression method. It is therefore highly desirable in this setting that a model is less dependent on feature engineering and have a strong ability to uncover the underlying hidden factors. A good representation is one that can capture the posterior distribution of the underlying explanatory factors. Deep learning models have strong representational abilities, and they are formed by the composition of multiple non-linear transformations (Bengio et al., 2013). Deep learning is a field within machine learning, that focuses on multiple levels of representations to transform the inputs into hidden units that represent the data generating process (Aggarwal, 2018; Goodfellow, 2016; LeCun et al., 2015). Deep learning spans over deep neural network techniques that use multiple hidden layers to learn representations of the data.

6.1 Feed-Forward Neural Networks

As pointed out by Lipton et al. (2015), neural networks are especially well suited for tasks where the underlying features are not directly interpretable. In the mortality forecasting problem, the underlying features are latent and have usually been estimated using the principal components method. Neural networks provide an additional way of uncovering these representations, and can, therefore, potentially be performing very well in the mortality forecasting setting. Input is a sequence of features $\left(x^{(1)}, x^{(2)}, ..., x^{(T)}\right)$ and similarly the same for the targets $\left(y^{(1)}, y^{(2)}, ..., y^{(T)}\right)$. As shown in fig. 1, the standard

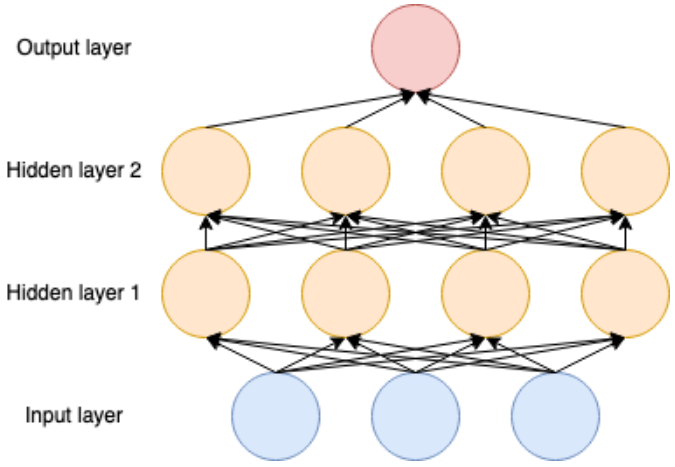

Figure 1: Neural network base structure using 2 hidden layers.

neural network processes information sequentially layer-wise through the network from the input layer to the output layer. Multi-layer neural networks are referred to as deep feed-forward neural networks (FFNN) because successive layers feed into one another in a forward direction from the inputs to the output layer (Aggarwal, 2018). A neural network aims to have a high representational ability by mimicking the neural structure of the brain (Bengio et al., 2013). Each layer has a set of neurons/units connected to all units in the previous layer and connects to all units in the next layer, which are the orange circles seen in fig. 1. Each unit has an associated activation function $a_j(\cdot)$ which can introduce non-linearities in the dependencies. The

value of each neuron j is calculated as a weighted sum of the incoming neurons,

$$v_j = a_j \left(\sum_{j'} w_{jj'} \cdot v_{j'} \right), \tag{51}$$

where $w_{jj'}$ is the "to-from" weight from neuron j' to j. The neuron also includes an intercept, that is commonly called bias in neural network literature (Aggarwal, 2018). As can be seen from eq. (51), without the activation function, this is equal to a linear regression. Common activation function choices are

- *Sigmoid:* $\sigma(z) = \frac{1}{1+e^{-z}}$.
- *Tanh:* $\phi(z) = \frac{\left(e^{z}-e^{-z}\right)}{e^{z}+e^{-z}}$.
- *ReLu:* $ReLu(z) = \max(0, z)$.

The derivatives of these activation functions are useful for the later backpropagation optimization algorithm,

- Sigmoid derivative: $\frac{\partial \sigma(z)}{\partial z} = \frac{\exp(-z)}{(1+\exp(-z))^2} = \sigma(z)(1-\sigma(z))$
- Tanh derivative: $\frac{\partial \phi(z)}{\partial z} = \frac{4 \cdot \exp(2z)}{(\exp(2z)+1)^2} = 1 - \phi(z)^2$
- ReLu derivative: $\frac{\partial ReLu(z)}{\partial z} = \begin{cases} 1, & \text{if } z > 0. \\ 0, & \text{otherwise,} \end{cases}$

Especially, the ReLu activation function allows for a lot of flexibility and has been found in other applications to be successful (Lipton et al., 2015). One of the reasons that the ReLu activation function performs well, is because the derivative is well behaved. A linear activation function is used for the output layer since the output is continuous in the mortality forecasting problem. The sigmoid function is useful when performing computations that should be interpreted as probabilities, as the output is between $(0, 1)$. The tanh function outputs a value between $[-1, 1]$ and is preferred when computations are desired to be both positive and negative. It also has a larger gradient than the sigmoid, which makes it easier to train. The ReLu is ultimately easier to train than the other two, which has made it a widely more popular activation function in hidden layers over the recent years (Aggarwal, 2018).

A single-layer neural network with only the input and output layer, and without an activation function is equal to a linear regression that is estimated with gradient-descent numerical optimization. Therefore, the single-layer and deep neural networks are flexible and range over a vast number of models that they can approximate, herein the LC model (Zhang et al., 2020; Richman and Wüthrich, 2018). Neural networks are trained using a gradient descent algorithm called backpropagation, proposed by Rumelhart et al. (1985).

6.1.1 Backpropagation

All outputs of a layer can be calculated given the information from the previous layers, and therefore the weights can also be optimized in this manner. The derivatives are calculated starting from the output and going back to the input layer, and then

all weights are updated. Optimization in a neural network is done by minimizing a loss function $L(\hat{y}, y)$. The loss function in the regression problem is typically the mean squared error loss function. The backpropagation optimization algorithm use chain rule derivatives with respect to each parameter to find the gradient descent direction, which minimizes the loss. The loss surface is non-convex, and therefore it is not ensured that backpropagation will reach global minimum. The result, therefore, depends on the initialization of the parameters. First, the derivative with respect to the weights in the output layer is calculated and then so on for the previous layers. After calculating all the gradients, then the weights are updated in the direction that minimizes the loss. Updates usually happen in batches of data, such that a sample of the data is used to update the weights in each update iteration. The first step is to compute the derivative of the loss with respect to the output o, $\frac{\partial L}{\partial o}$. Subsequently, the derivatives are calculated in a backward direction through the network using chain rule derivatives. The parameter from the hidden layer h_{r-1} to h_r denoted as $w_{r,r-1}$ will have the chain rule derivative that is given as

$$\frac{\partial L}{\partial w_{r,r-1}} = \underbrace{\frac{\partial L}{\partial o} \cdot \left[\sum_{[h_r, h_{r+1}, \ldots, h_k, o] \in \mathcal{P}} \frac{\partial o}{\partial h_k} \prod_{i=r}^{k-1} \frac{\partial h_{i+1}}{\partial h_i} \right]}_{\Delta(h_r, o) = \frac{\partial L}{\partial h_r}} \frac{\partial h_r}{\partial w_{r,r-1}}. \tag{52}$$

The partial derivative is therefore computed by aggregating the product of the partial derivatives from all paths from h_r to the output o. The value h_r is the post-activation value $h_r = a_h(x_{h_r})$, and x_{h_r} is the input that goes into the hidden layer h_r. The set of paths to the output is denoted as $\mathcal{P}$. The value of $\Delta(h_r, o) \frac{\partial L}{\partial h_r}$ is what is calculated in the backpropagation update, as the last part can be calculated beforehand which is the derivative of the weight with respect to the activation function. This quantity can be calculated as

$$\Delta(h_r, o) = \frac{\partial L}{\partial h_r} = \sum_{h : h_r \Rightarrow h} \frac{\partial L}{\partial h} \frac{\partial h}{\partial h_r} = \frac{\partial h}{\partial h_r} \Delta(h, o), \tag{53}$$

where $h : h_r$ denotes all the subsequent layers from h_r. When computing derivatives for layer h_r then all the subsequent layers h will already be calculated when evaluating $\Delta(h_r, o)$. What is calculated here is therefore $\frac{\partial h}{\partial h_r}$. The layer $h = a_h(x_h)$ transform the inputs x_h, which is a linear combination of the previous units through the activation function a_h. This means that the amount $\frac{\partial h}{\partial h_r}$ is computed as,

$$\frac{\partial h}{\partial h_r} = \frac{\partial h}{\partial x_h} \frac{\partial a_h(x_h)}{\partial x_h} = \frac{\partial a_h(x_h)}{\partial x_h} w_{h_r, h} \tag{54}$$

The backward recursion is linear in the number of edges between units in the network, which is a useful feature that ensures that computation is feasible (Aggarwal, 2018). The negative gradient indicates the direction of the steepest descent in the loss-surface, where the error on average is low (LeCun et al., 2015; Lipton et al., 2015). Backpropagation can use dynamic programming to increase speed, as the derivatives for a layer can be used for all the preceding layers (Aggarwal, 2018). Backpropagation updating is done in batches of observations to make updates more feasible. After the whole dataset has been used in batches, this is called an epoch. The backpropagation is then run for an appropriate amount of epochs until the parameter values have converged (Zhang et al., 2020).

6.1.2 Gradient descent strategy

It is normal to use the steepest descent algorithm, where the update happens with respect to the steepest descent of the gradient. Data usually is, as in this thesis, updated in batches using stochastic gradient descent, which updates parameters for a sample of observations at each update. Therefore, the model parameters are first initialized, typically at random, with

values around zero. Then model parameters are updated iteratively using batches of the data for each layer starting backward at the output layer, updating the parameters in the negative gradient direction. The gradients are calculated with respect to all batch observations, to add up their local gradient and execute them for all of these observations at once (Aggarwal, 2018). Stochastic gradient descent is a greedy algorithm, as it only considers the steepest direction for the sample of observations that it is currently updating.

It is a good idea to use a decaying learning rate for the updating mechanism in the backpropagation algorithm. The learning rate measures how much of the new update is added to the current parameter values. Starting with a high learning rate will make the algorithm go faster towards optima, and the decay will ensure that it does not diverge at a later point. The decaying learning rate is used in this thesis, where the learning rate decreases with a certain factor when the backpropagation algorithm does not improve the loss for a certain amount of epochs (Aggarwal, 2018). Using a constantly high learning rate leaves the derivatives oscillating around the optima that it has found. Instead, one can use parameter-specific learning rates that are specific to each parameter. It is done because parameters with large partial derivatives often are oscillating, and parameters with small partial derivatives tend to be more consistent in the movement towards optima (Aggarwal, 2018). The popular Adam gradient descent strategy is well suited for stochastic gradient descent and is used in this thesis. A_i is the exponential weighted average value of the $i'th$ parameter

$$A_i \Leftarrow \rho A_i + (1-\rho)\left(\frac{\partial L}{\partial w_i}\right)^2, \qquad \forall i. \tag{55}$$

Where ρ is a decaying parameter. Smoothing is performed with a different decaying parameter ρ_f in

$$F_i \Leftarrow \rho_f F_i + (1-\rho_f)\left(\frac{\partial L}{\partial w_i}\right), \qquad \forall i. \tag{56}$$

The Adam update is then given by

$$w_i \Leftarrow w_i - \frac{\alpha_t}{\sqrt{A_i}+\epsilon} F_i, \qquad \forall i, \tag{57}$$

where α_t is the learning rate at update iteration t (Aggarwal, 2018; Kingma and Ba, 2014). It is smoothed by using the previous values of itself. By dividing with the squared gradient A_i, then it is ensured that the weight update gets smaller and smaller, ensuring that an optimum is not missed. When it oscillates a lot, then A_i is large, and the updates are reduced.

Neural networks often face problems with overfitting, meaning that the algorithm is not able to generalize accurately, fitting too much to the estimation data. To try and overcome overfitting in a neural network, regularly dropout is used.

6.1.3 Dropout

Dropout tries to make the network spread the weight out on all the features, rather than focusing on a few, potentially spurious, relations in the data (Zhang et al., 2020). Dropout holds out a portion of the neurons during training. Dropout is generally specified for each layer during training, where the probability for a neuron to be dropped in the given layer is determined by a dropout probability p_j (Zhang et al., 2020). If a node is dropped during training, then all incoming and outgoing connections are dropped as well. The idea is much like the random forest algorithm, where only a subset of split-variables is considered at each split. For a neural network, it forces the algorithm to focus on all neurons rather than just a few. It is also possible to use batched normalization to stabilize optimization.

6.1.4 Batch normalization

Batch normalization is a technique that normalizes the inputs in each batch. The neural network is updated using batches of data, and to make these inputs more stable, the inputs are normalized in each batch. It transforms the inputs of a batch to have zero mean and unit variance. The parameters change during training, as only a batch of the data is seen in each update. When the inputs are normalized using this technique, then the derivative is better behaved. It is less likely to be stuck in a local optimum and the gradient vanishing/exploding. Covariate shifts between batches are reduced, and the network converge faster (Ioffe and Szegedy, 2015).

6.1.5 Embeddings

Another way of transforming the input space is by using embeddings. Richman and Wüthrich (2018) found that using embedding connections improved performance for neural networks in the mortality forecasting setting. This thesis finds similar results, where performance is improved by using embeddings to transform the feature-space of the categorical variables. Categorical variables are often modeled as dummy-variables when input into machine learning models. It transforms the input into a sparse vector of binary dummy variables. This can create problems in estimation if there are not representations for some of the values to learn the relationship between the output and the given category. One can instead transform the categorical variables using embeddings. Embeddings are continuous representations of discrete variables. Embedding vectors maps the discrete variable on a continuous scale, mapping the relationship between the categories. It can be the relationship between different ages if the embeddings are modeled on the age-dimension in the mortality forecasting framework. The relationships are learned during optimization using skip-gram connections. The skip-gram connections take a single input of w and output m context outputs. The goal is for the skip-gram connection to estimate $P\left(w_1, w_2, ..., w_m \mid w\right)$ (Aggarwal, 2018). Skip-grams use the context of inputs appearing to have closely related values to help reduce dimensionality. Using embeddings for each categorical feature allows a dense representation to be learned for each feature before being combined in the neural network (Richman, 2018). It gives a way of learning patterns for discrete features better, as they are learned in groups rather than each category of the variable. Skip-gram embedding layers are used throughout the deep learning models as they are found to improve the models. The skip-gram connection method can be viewed as an application of exponential-family principal components analysis (EPCA) to an integer matrix of co-occurrence counts (Cotterell et al., 2017). Skip-gram is EPCA using the multinomial distribution and log link function. When using embeddings in a neural network, the values are updated during backpropagation to optimize with respect to the loss function. In the mortality rate forecasting problem, it means the embeddings are finding ages, countries, genders, and cohorts related to the same mortality rates.

6.2 Deep learning using recurrent neural networks

A disadvantage of using feed-forward neural networks for time-series predictions is that they rely on the assumption of independence between each observation. After processing each observation, the entire state is lost. If data points are not independent but instead have some dependence in time and space, then this can be a poor assumption. Recurrent neural network (RNN) models are neural network models with the ability to sequentially process information without assuming independence between the data points. RNN models allow hidden layers from the previous time-steps to affect the current time-step. RNNs are very expressive, and they are said to be *"Turing complete"*, which means they can simulate any algorithm, given that enough data and computational resources are provided (Aggarwal, 2018).

The output of a hidden layer at time t with direct influence from the previous time-step is calculated as

$$h^{(t)} = a_j \left(W_{hx} x^t + W_{hh} h^{(t-1)} + b_h \right) \tag{58}$$

Potentially the direct influence from previous time-steps can be arbitrarily long. W_{hx} is a weight matrix between the hidden-layer and the inputs and W_{hh} is a weight matrix of recurrent weights between the hidden layer from period t and period $t-1$. Connections between time-steps are called recurrent edges.

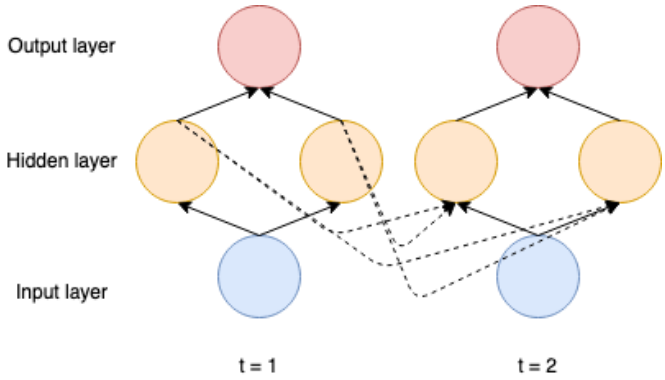

Figure 2: RNN base structure with 1 hidden layer.

As seen in fig. 2 the RNN model can be interpreted as the deep neural network explained in section 6.1 with one neural network for each time-step, where each layer takes an extra input, which is the output from previous steps. From this intuitive understanding of the RNN model, it is clear that much of the same principles can be used during optimization. The backpropagation through time algorithm that was proposed by Werbos (1990) is commonly used for training the RNNs.

6.2.1 Backpropagation through time

An assumption in the backpropagation algorithm is that weights in different layers are distinct from one another. The backpropagation through time starts by assuming that parameters in the temporal layers are independent of each other. After calculating the gradient for each of these parameters, the contribution from different time-steps is added to create a unified update for each weight parameter, thereby accounting for the temporal dependence. Therefore, the backpropagation through time algorithm starts as the normal backpropagation by computing the loss. Then, the gradient for each weight parameter is calculated, treating all parameters in each temporal layer as unique. As with backpropagation, this process starts backward from the last period. Finally, all shared weights are added to compute the parameter update as,

$$\frac{\partial L}{\partial W_{xh}} = \sum_{t=1}^{T} \frac{\partial L}{\partial W_{xh}^{(t)}} \tag{59}$$

This procedure is very similar to the backpropagation algorithm but is adjusted to accommodate for the temporal dimension (Aggarwal, 2018). This can be a very computational process if the time-dependencies are very long, and therefore usually, a truncated backpropagation through time algorithm is used. This algorithm updates only with certain lookback through time, such that the whole sequence is not used, but only a certain amount of time-steps back.

A problem that is sometimes seen for regular RNN models is that the long-range dependencies are hard to optimize, as the gradient explodes or vanishes (Bengio et al., 1994; Lipton et al., 2015; Chung et al., 2014). It happens because the long-term dependencies are hidden by the effect of short-term dependencies. To overcome this problem, often gated recurrent neural networks are used instead. The gradient of the sigmoid function is never larger than 0.25, and thus long-range dependencies will quickly have the gradient vanish. The ReLu activation function is less prone to having the gradient vanish when the input values operate in the region where the gradient is equal to one for this activation function (Aggarwal, 2018). The gradient descent updating mechanism is made for infinitesimally small steps, and the gradient can vanish for long dependencies or explode depending on the size of the gradient. The LSTM and GRU are gated recurrent neural networks proposed to overcome the challenges of regular RNN models.

6.2.2 Long Short Term Memory (LSTM) neural network

LSTM hidden layers were introduced by Hochreiter and Schmidhuber (1997), which introduces a memory cell to replace each unit in the layer. The instability caused in RNN models is the direct result of successive multiplication of the weight matrix at various time-steps (Aggarwal, 2018).

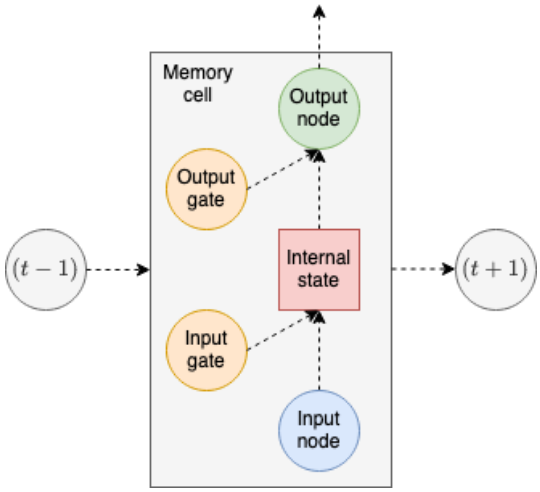

Figure 3: LSTM memory cell structure.

The memory cell was introduced to overcome the vanishing gradient problem. Each memory cell contains a self-connected time weight that is fixed to a value of one. This ensures that the gradient is constant, and therefore does not vanish nor explode. The name is derived from the fact that the network has a long memory by using information from previous time-steps and short memory from previous information in the network at the same time-step. The structure of the memory cell can be seen in fig. 3. In the memory cell structure from fig. 3, the memory cell retains long-term information by updating the state in increments from previous states. This ensures that the successive multiplication problem is alleviated. If the states in different time-steps share a higher level of similarity through this persistence, then it is also harder for the gradient to be drastically different (Aggarwal, 2018).

Input nodes take inputs $x^{(t)}$ and the values from the hidden layer at the previous time-step $h^{(t-1)}$, and transforms it using an activation function

$$g^{(t)} = a\left(W^{gx}x^{(t)} + W^{gh}h^{(t-1)} + b_g\right). \tag{60}$$

Input gates is a weight between zero and one for each of the input node inputs, using a sigmoid activation function. Given a value of zero for a particular input then that means that the information from that input will be cut-off. A value of one, on the other hand, means that all information from that input will be used.

$$i^{(t)} = \sigma\left(W^{gx}x^{(t)} + W^{gh}h^{(t-1)} + b_g\right). \tag{61}$$

Internal states has a self-connected recurrent edge with a weight of one. This means that information will flow between time-steps without the gradient exploding or vanishing. It collects the information from the input gate, the input node, and the previous internal state.

$$s^{(t)} = g^{(t)} \odot i^{(t)} + f^{(t)} \odot s^{(t-1)}, \tag{62}$$

where $\odot$ is point-wise matrix multiplication and $f^{(t)}$ is a forget-gate. Forget gates work much like the input-gate and weights each of the inputs coming from the previous internal state between zero and one.

Output gates weights the output of the internal state, just like the input gate did with the input from the input node. Ultimately $h^{(t)}$ is the output produced by the memory cell in the output node. The output gate produces the weight that is multiplied onto the value of the internal state to produce $h^{(t)}$.

Output node produces the final value of the memory cell that is used much like other neurons in a normal neural network.

$$h^{(t)} = a\left(s^{(t)}\right) \odot o^{(t)}, \tag{63}$$

where $o^{(t)}$ is the vector of weights from the output gate (Lipton et al., 2015).

The partial derivative of the internal state from eq. (62) with respect to the previous value is going to be the forget gate value. They are often initially high, which means the gradient decay will be slow (Aggarwal, 2018). Like the LSTM neural network, gated recurrent unit (GRU) networks were also proposed to overcome the gradient challenges of the normal RNN.

6.2.3 Gated recurrent unit (GRU) neural network

GRU neural networks were proposed by Cho et al. (2014b,a) to overcome the challenges of a normal RNN. The main difference from an RNN is the way that it incorporates gating into the hidden state variable that allows for time-dependencies. It thereby includes a mechanism that tells the network when to update the time-dependencies and when to reset them. Thus, it can keep important information stored and learn to ignore irrelevant temporary information. The GRU hidden layer will replace each node with a GRU cell just like the LSTM did with the memory cell. The GRU cell structure can be seen in fig. 4.

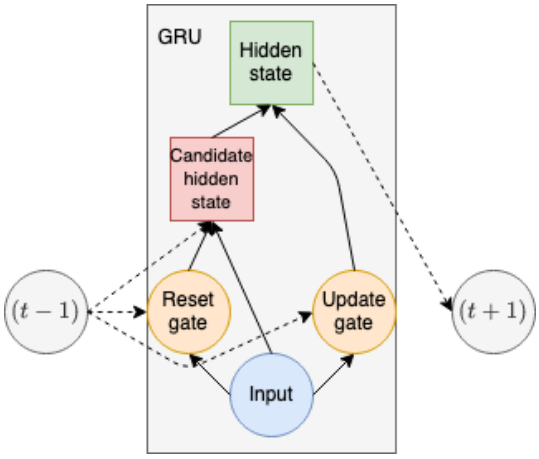

Figure 4: GRU cell structure.

The GRU cell takes in inputs and hidden state value from the previous period. These go into the reset gate that outputs a weight between zero and one determining how much previous information should be kept. Likewise, these inputs are also used in an update gate that determines how much of the new information should be used. The update gate determines how much information from previous and current states needs to be passed along to the future. On the other hand, the reset gate is used to determine how much of the previous information needs to be forgotten. Thus, even though these two gates work in the same way, they do two different tasks. The reset gate is calculated as

$$R^{(t)} = \sigma\left(X^{(t)}W^{rx} + H^{(t-1)}W^{rh} + b_r\right), \tag{64}$$

and the update gate as

$$Z^{(t)} = \sigma\left(X^{(t)}W^{zx} + H^{(t-1)}W^{zh} + b_r\right). \tag{65}$$

The candidate hidden state is calculated using a hyperbolic tangent activation function as,

$$\tilde{H}^{(t)} = \phi\left(X^{(t)}W^{hx} + \left(R^{(t)} \odot H^{(t-1)}\right)W^{hh} + b_h\right). \tag{66}$$

Afterward, the candidate hidden state and the update gate is used to compute the hidden state of the node at time t,

$$H^{(t)} = Z^{(t)} \odot H^{(t-1)} + \left(1 - Z^{(t)}\right) \odot \tilde{H}^{(t)} \tag{67}$$

Thus, as it can be seen from eq. (67), the update gate determines to which extend the current hidden state is from the previous hidden state and how much of the current information is used. When the update gate $Z^{(t)}$ is close to one, then this essentially means that the old state is retained and the current inputs $X^{(t)}$ is ignored.

GRU differs from LSTM in that it does not have an input gate that controls how much of the current state that the internal hidden state is exposed to. Similarly, the GRU network does not have an output gate either, which controls how much the rest of the network is exposed to the information from the recurrent unit cell (Chung et al., 2014). Here the GRU unit exposes the rest of the network to the full information. As with LSTM, GRU also stores a partial copy of the previous hidden states, which makes the gradient flow more stable during backpropagation (Aggarwal, 2018).

6.3 Proposed deep learning structure

The FFNN structure proposed in this thesis can be seen in fig. 14. The proposed model transforms the categorical values using embeddings (section 6.1.5), which finds hidden structures and relationships in the categorical variables. The available categorical variables in this thesis are age, country, subpopulation, and possibly also cohort. Subpopulations available for each country are males, females, and the total population. The model also inputs the current period's log mortality rate and the year t. When forecasting for more than one year ahead of time, the current periods log mortality rate input will be a forecasted value. The output of the model is the log-mortality rate for country c and subpopulation i for year $t+1$.

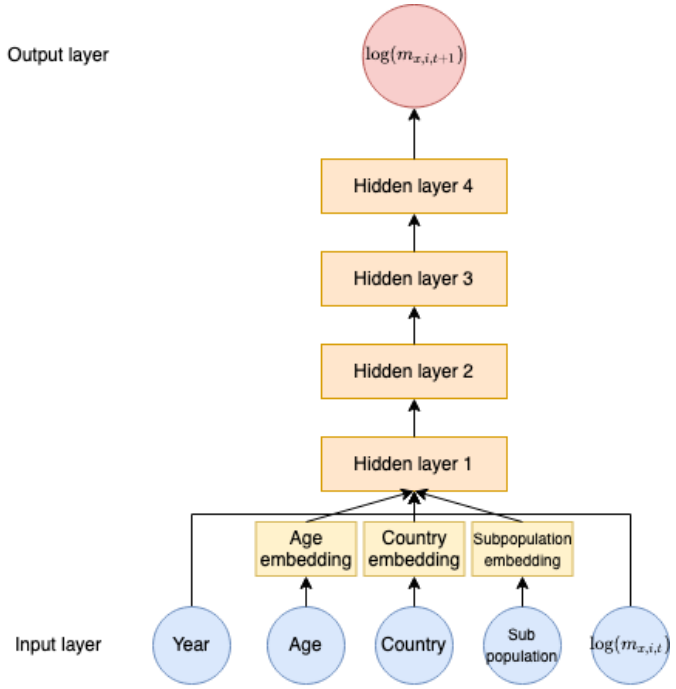

Figure 5: Proposed deep feed-forward neural network model structure.

The model that is proposed in fig. 14 will additionally have four hidden layers that will ensure the high flexibility of the model. Furthermore, each of the hidden layers uses the ReLu activation function, which is flexible and well-behaved when training the model. Each layer will also during training have a 5% dropout-rate and use batch normalization to avoid overfitting. A very similar model is proposed for the temporal models, which can be seen in appendix C. The difference here is that the temporal model will have a recurrent layer on top of the embedding layer, capturing the value's dependencies. The recurrent layer can either be a normal RNN layer, as presented at the beginning of section 6.2, the LSTM layer (section 6.2.2) or the GRU layer (section 6.2.3). The temporal models will still also have four normal feed-forward layers before the output.

Through extensive model testing, the base structure described in fig. 14 proved to be the best and is therefore used for all the neural network models. Especially crucial for the model performance was the inclusion of the embedding step, which significantly improved the model performance. Additionally, the number of hidden layers, activation function, and dropout were tuned for optimal performance.

7 Model performance comparison

7.1 Loss function

Given the availability of a large number of models raises the need for a statistical procedure to find the set of best models. For forecasting, the models are mainly valued on their predictive abilities on an unseen test dataset. In this thesis, the models are estimated on data from 1950 - 1999 and their predictive ability is measured on the unseen data from 2000 - 2016. Typical loss functions are

- Mean squared error (MSE): $L\left(Y,\hat{f}\left(X\right)\right)=\frac{1}{T+N_x}\sum_x\sum_t\left(Y_{x,t}-\hat{f}\left(X_{x,t}\right)\right)^2$
- Mean absolute error (MAE): $L\left(Y,\hat{f}\left(X\right)\right)=\frac{1}{T+N_x}\sum_x\sum_t\left|Y_{x,t}-\hat{f}\left(X_{x,t}\right)\right|$
- Root mean squared error (RMSE): $L\left(Y,\hat{f}\left(X\right)\right)=\sqrt{\frac{1}{T+N_x}\sum_x\sum_t\left(Y_{x,t}-\hat{f}\left(X_{x,t}\right)\right)^2}$

The choice of the loss function usually depends on how much focus is put on the estimated values that are far from the actual values. The MSE loss function punishes values far from the actual values more than the MAE loss function. Thus, the MSE loss function is chosen as the appropriate loss function in the mortality forecasting problem.

The expected test error is the forecasting error over the test data $\mathcal{T}$,

$$Err_{\mathcal{T}}=E\left[L\left(Y,\hat{f}\left(X\right)\right)\mid\mathcal{T}\right]. \tag{68}$$

Estimation/training dataset error is not of interest here as this would lead to overfitting of the training data and poor generalization to the test data. Because the goal is to forecast future mortality rates, then the target should be the test error.

It is usually not expected that a single model will dominate all competitors, either because they are statistically alike or because data does not provide enough information to discriminate the performance of the models (Bernardi and Catania, 2015). The goal of the models is to perform well on unseen data to make precise mortality projections. Therefore, a statistical test for significance in forecasting error is used to test the performance difference of the models. One popular method to test for differences between model performances of multiple models is the model confidence set proposed by Hansen et al. (2011). Typically when applying statistical learning models such as the deep learning models, one would have training, validation, and test data. Training data is used to estimate the model, validation data is used to find the optimal hyperparameters, and test data is used to evaluate the generalization ability of the model (James, 2013; Hastie, 2009).

7.2 Model confidence set

The model confidence set (MCS) of Hansen et al. (2011) consists of a sequence of statistical tests, with the null hypothesis of equal predictive ability (EPA). Given that the null hypothesis is rejected, this indicates that one or more models perform better than others. Therefore, for each round that the EPA test is rejected, the worst-performing model is eliminated. This elimination procedure starts with the full set of models M_0 and eliminates models from the set of best models until the EPA

test is no longer rejected. When the null hypothesis is no longer rejected, then the set of superior models $\hat{M}^*_{1-\alpha}$ at a confidence level of $1-\alpha$ is found.

The EPA test statistic is calculated for an arbitrary loss function that satisfies weak stationarity conditions. This allows the procedure to be applied on different grounds, such as the predictive ability as done in Hansen and Lunde (2005) and in this thesis, or it could be the goodness-of-fit as done in Hansen et al. (2011). Hansen and Lunde (2005); Bollerslev et al. (1994) points towards the natural choice of MSE when the loss is not defined with respect to variances. One could also use mean absolute error (MAE) that is more robust to outliers. The loss for the i'th model at time t can formally be defined as

$$l_{i,t} = L\left(Y_t, \hat{Y}_{i,t}\right) \tag{69}$$

The procedure starts with the initial set of models $\hat{M}^0$, and for a given confidence level $1-\alpha$, it will deliver a smaller set of models, that is the superior set of models $\hat{M}^*_{1-\alpha}$ with $m^* \leq m$ models. The difference in loss between two models can be defined as

$$d_{ij,t} = l_{i,t} - l_{j,t}, \qquad i,j = 1,...,m, \qquad t = 1,...,T, \tag{70}$$

and the simple difference of model i relative to the other models at time t is given as,

$$d_{i\cdot,t} = \frac{1}{m-1}\sum_{j\in M} d_{ij,t}, \qquad i = 1,...,m. \tag{71}$$

The EPA hypothesis for a given set of models M can be formulated as

$$\begin{aligned} H_{0,M} &: c_{ij} = 0, \qquad \text{for all } i,j = 1,...,m \\ H_{A,M} &: c_{ij} \neq 0, \qquad \text{for some } i,j = 1,...,m, \end{aligned} \tag{72}$$

or as

$$\begin{aligned} H_{0,M} &: c_{i\cdot} = 0, \qquad \text{for all } i = 1,...,m \\ H_{A,M} &: c_{i\cdot} \neq 0, \qquad \text{for some } i = 1,...,m, \end{aligned} \tag{73}$$

where $c_{ij} = E\left[d_{ij}\right]$ and $c_{i\cdot} = E\left[d_{i\cdot}\right]$ (Bernardi and Catania, 2015).

Then the following tests are constructed to test the hypothesis in eq. (72) and eq. (73) respectively,

$$t_{ij} = \frac{\bar{d}_{ij}}{\sqrt{v\hat{a}r\left(\bar{d}_{ij}\right)}}, \qquad t_{i\cdot} = \frac{\bar{d}_{i\cdot}}{\sqrt{v\hat{a}r\left(\bar{d}_{i\cdot}\right)}}. \tag{74}$$

Here $\bar{d}_{i\cdot} = \frac{1}{m-1}\sum_{j\in M}\bar{d}_{ij}$ is the loss of the i'th model relative to the average losses across the models in the remaining model set M. The number $\bar{d}_{ij} = \frac{1}{m}\sum_{t=1}^{T} d_{ij,t}$ measures the relative loss between the i'th and j'th models. The variances $v\hat{a}r\left(\bar{d}_{i\cdot}\right)$ and $v\hat{a}r\left(\bar{d}_{ij}\right)$ are the bootstrapped estimates of the respective variances. Hansen et al. (2011) suggests performing a block-bootstrap with 5000 resamples. The block length p is the maximum number of significant parameters obtained by fitting an $AR\left(p\right)$ process on all the d_{ij} terms.

The two test statistics in eq. (74) can be mapped to

$$T_{R,M} = \max_{i,j\in M}|t_{ij}|, \qquad T_{\max,M} = \max_{i\in M} t_{i\cdot}, \tag{75}$$

which can be used to test the EPA hypothesis. The test-distributions under the null hypothesis are found using bootstrap as their distributions are non-standard (Bernardi and Catania, 2015). The elimination rule is given as

$$e_{R,M} = \arg\max_i \left\{ \sup_{j \in M} \frac{\bar{d}_{ij}}{\sqrt{\hat{var}\left(\bar{d}_{ij}\right)}} \right\}, \qquad e_{\max,M} = \arg\max_{i \in M} \frac{\bar{d}_{i\cdot}}{\sqrt{\hat{var}\left(\bar{d}_{i\cdot}\right)}}. \tag{76}$$

As suggested by Bernardi and Catania (2015), one can use the set of superior models $\hat{M}^*_{1-\alpha}$ to make a combined forecast, hoping to overcome the weaknesses of a single model. Typically the models display different levels of misspecification, and hence a strategy that pools information of models can improve upon the forecasting results. Using the results in section 8, one can potentially deploy a strategy that uses different models for different age-ranges.

8 Empirical study

Using data retrieved from the Human Mortality Database (Wilmoth and Shkolnikov), and with the procedure described at the end of section 2, a large empirical model comparison study is performed. Both in terms of average and median MSE values and in terms of the MCS approach explained in section 7.2. This is done to test the performance of many populations. The study gives an overview of the models' robustness, overall performance, and sensitivity to different ages. Furthermore, the best performing model is used to make future mortality projections, and these are compared to the projections of the baseline LC model. Section 8.1 compares the performance of the models using the formal MCS test approach, and section 8.2 evaluates the mortality projections.

8.1 Performance comparison

The MCS can be calculated at different levels. The MCS compares the loss function values up to any given forecast period. At each time there will be an error for the forecasted value at each age, which in this case, is 46 different ages. Thus, one can compute the mean error across all ages, or one can study the error at each age. For each time step, there are 69 evaluations, which means there are 69 times that it is possible to be in the set of superior models.

8.1.1 Across all ages using MSE

The reported results for the model confidence set is reported in table 1. For each time step, the loss is the MSE value across all ages.

Deep learning models	**Number of times in MCS**	**% in MCS**	**Average MSE***	**Median MSE***
FFNN (2000 epochs)	58	84.06%	1.01	0.61
FFNN (500 epochs)	54	78.26%	1.51	0.84
FFNN (with cohort)	60	86.96%	**0.92**	0.47
FFNN (with factors)	50	72.46%	1.17	0.61
FFNN (with factor loadings)	6	8.70%	3.20	2.65
FFNN (200 epochs)	31	44.93%	1.55	0.91
GRU (with cohort, no dropout)	45	65.22%	1.24	0.83
GRU (with cohort)	38	55.07%	1.30	0.95
LSTM (with cohort, no dropout)	**61**	**88.41%**	1.02	**0.44**
LSTM (with cohort)	47	68.12%	1.08	0.70
RNN (with cohort, no dropout)	50	72.46%	1.60	0.59
RNN (with cohort)	41	59.42%	1.26	0.67
Single population machine learning models				
Boosting (sp)	3	4.35%	3.30	2.78
RF (sp)	3	4.35%	3.29	2.94
SVM (sp)	1	1.45%	4.03	3.72
XGB (sp)	2	2.90%	3.82	3.03
Multi population factor model				
Multi level factor	31	44.93%	2.04	0.98
Multi population machine learning models				
Boosting (mp)	2	2.90%	4.35	3.77
RF (mp)	1	1.45%	5.25	3.88
SVM (mp)	6	8.70%	3.09	2.40
XGB (mp)	3	4.35%	3.74	3.25
Single population factor models				
APC	13	18.84%	2.36	1.74
CBD	30	43.48%	2.50	0.84
LC (GLM)	39	56.52%	1.78	0.89
LC (SVD)	30	43.48%	35.45	0.93
M6	23	33.33%	2.60	1.56
M7	6	8.70%	6.80	3.78
M8	6	8.70%	727.09‡	5.03
Plat	32	46.38%	1.85	0.82
RH	26	37.68%	138650504‡	1.94

*Table 1: *All MSE values are multiplied with* 10^4. ‡ *Extreme values are caused by extremeties for some populations, where the fit is poor.*

As it can be seen from table 1, the deep learning models outperform all of the other models. Amongst the highest performing models are the FFNN, LSTM, and RNN models. As can be seen from table 1, the RH model suffers severely from robustness issues, and it displays large extremities for some countries. The M8 model also suffers from some robustness problems, but not quite as extreme as the RH model. The M8 extension was proposed to overcome the robustness issues of the CBD model but makes the problem worse in this case. The highest performing single population model is the LC model estimated using the maximum likelihood method proposed by Brouhns et al. (2002a). Adding the factors or factor loadings from the multi-level factor model to the deep learning model gave no added performance. It is also seen that using more epochs in the estimation of the deep learning models improves performance. However, the improvement becomes marginally smaller as the number of epochs is increased. It can also be noted from table 1 that the deep learning models have the lowest average and median MSE values.

As mentioned, the MCS can also be applied at a deeper level for each age to see if the models have any ages where they are not performing well compared to the other models.

8.1.2 For individual ages using squared error loss

At each age, the error at any given forecast period is the squared error (SE) loss. At each age, there are again 69 populations, meaning that each model has the possibility of being in the superior set of models 69 times for each age. From table 2, it can be seen that the recurrent models are not performing as well for the 50 - 59 age group. The recurrent models are not able to capture the patterns for these ages. This age group has low mortality, and the recurrent models fit too little to this behavior. The gradient can be very small in this area, and it can cause the gradient to vanish for the recurrent models[4]. The best performing models in this age group are the FFNN models and the multi-level factor model. For ages 60 - 69, there is starting to be a more clear time trend, and the temporal addition to the deep learning models becomes more useful. Both the deep learning models, the multi-level factor model, and the single-population models have very high performance in this interval, as can be observed from table 3.

For ages 70 - 79, it can be seen from table 4 that the single-population models have inferior performance. This age group has seen a larger mortality reduction over time than much of the other age groups. Therefore, single-population models tend to underestimate this improvement, causing them to have poor performance. It is a potentially important age group for policy evaluations regarding the retirement age. As observed in table 4, it is the recurrent deep learning models that perform best for these ages, where the linear models have poor performance.

For ages 80 - 89, it can be observed in table 5 that the recurrent models are still amongst the highest performing models together with the FFNN models.

For the oldest ages, seen in table 6, the deep learning and the single-population models are both performing very well, with a slight edge for the deep learning models. Here it can also be noticed that the multi-level factor model also performs well. When many models are performing well, this can also be a sign that these ages do not provide much information to distinguish the model's performance from each other.

These individual age results can be summarized to table 7. The total number of possible times in the superior set of models is

[4]The observed behavior of the RNN model vs. the FNNN model for the 50 - 59 age group can be seen in appendix F

	Age									
Deep learning models	50	51	52	53	54	55	56	57	58	59
FFNN (2000 epochs)	52	49	49	54	56	55	51	59	60	61
FFNN (500 epochs)	**54**	53	49	57	54	**56**	**64**	**65**	**65**	**66**
FFNN (with cohort)	14	14	16	13	17	15	18	17	27	37
FFNN (with factors)	52	**55**	**62**	**59**	**61**	55	59	64	63	64
FFNN (with factor loadings)	29	24	27	21	24	23	26	23	29	39
FFNN (200 epochs)	26	17	23	24	25	25	27	23	30	43
GRU (with cohort, no dropout)	5	7	6	7	7	9	10	9	14	17
GRU (with cohort)	5	6	6	7	6	7	9	8	13	18
LSTM (with cohort, no dropout)	6	7	7	7	6	7	10	10	15	16
LSTM (with cohort)	6	7	6	6	7	6	7	7	14	13
RNN (with cohort, no dropout)	6	5	6	5	6	7	8	9	11	13
RNN (with cohort)	6	5	5	5	4	5	6	7	12	23
Single population machine learning models										
Boosting (sp)	3	3	2	2	4	6	8	14	22	33
RF (sp)	1	0	0	0	0	2	3	3	5	4
SVM (sp)	0	0	0	0	0	0	1	1	2	3
XGB (sp)	10	15	23	23	23	20	29	29	33	40
Multi population factor model										
Multi level factor	48	53	58	52	50	52	54	50	52	53
Multi population machine learning models										
Boosting (mp)	6	7	10	11	13	17	16	21	28	36
RF (mp)	0	0	0	0	0	0	0	0	1	0
SVM (mp)	4	5	6	8	9	11	10	12	15	21
XGB (mp)	26	24	23	26	17	17	18	18	29	37
Single population factor models										
APC	36	36	36	35	38	34	39	37	43	47
CBD	31	36	35	36	31	37	37	40	46	49
LC (GLM)	39	40	45	43	41	35	38	34	45	48
LC (SVD)	42	41	44	42	38	36	39	36	44	51
M6	36	31	28	27	22	27	27	24	27	31
M7	24	19	21	20	15	19	19	15	17	16
M8	4	4	5	7	8	12	14	19	27	35
Plat	45	35	35	35	33	34	32	27	33	36
RH	10	13	16	15	13	20	22	26	36	37

Table 2: Individual age MCS results for ages 50-59.

	Age									
Deep learning models	60	61	62	63	64	65	66	67	68	69
FFNN (2000 epochs)	63	65	66	67	66	**69**	68	55	48	49
FFNN (500 epochs)	**69**	**69**	**69**	**69**	**69**	**69**	**69**	50	47	45
FFNN (with cohort)	53	67	68	**69**	**69**	68	67	35	34	32
FFNN (with factors)	65	68	**69**	**69**	**69**	**69**	**69**	45	42	43
FFNN (with factor loadings)	44	64	66	65	63	50	17	5	5	5
FFNN (200 epochs)	48	62	66	67	66	58	50	15	16	18
GRU (with cohort, no dropout)	32	50	**69**	67	**69**	68	**69**	**67**	66	65
GRU (with cohort)	36	54	67	**69**	**69**	67	66	63	**68**	**66**
LSTM (with cohort, no dropout)	25	46	68	**69**	**69**	67	67	63	65	62
LSTM (with cohort)	30	51	67	**69**	**69**	**69**	**69**	65	63	62
RNN (with cohort, no dropout)	22	45	68	**69**	**69**	68	**69**	65	66	64
RNN (with cohort)	39	64	66	67	66	66	66	58	58	59
Single population machine learning models										
Boosting (sp)	45	55	60	58	59	54	46	31	27	25
RF (sp)	11	18	29	25	25	20	5	3	1	3
SVM (sp)	6	5	5	1	2	1	1	1	0	1
XGB (sp)	39	57	62	62	58	51	33	13	10	11
Multi population factor model										
Multi level factor	56	64	65	65	65	65	59	27	24	23
Multi population machine learning models										
Boosting (mp)	46	54	58	57	54	48	38	21	13	16
RF (mp)	0	0	0	0	0	0	0	0	0	0
SVM (mp)	27	34	39	37	35	31	26	19	18	21
XGB (mp)	42	63	64	64	62	59	31	11	5	5
Single population factor models										
APC	52	60	62	62	60	59	54	21	16	18
CBD	55	65	66	66	65	63	58	20	15	10
LC (GLM)	49	62	63	63	62	57	47	17	15	17
LC (SVD)	49	63	65	63	63	60	50	16	12	14
M6	41	52	48	47	45	36	29	12	7	7
M7	21	26	24	22	23	20	20	8	6	6
M8	46	61	64	65	62	57	44	4	1	2
Plat	42	59	59	59	60	58	50	10	5	6
RH	47	52	52	52	52	49	46	35	41	38

Table 3: Individual age MCS results for ages 60-69.

	Age									
Deep learning models	70	71	72	73	74	75	76	77	78	79
FFNN (2000 epochs)	46	42	45	46	44	40	47	50	45	49
FFNN (500 epochs)	47	40	36	44	40	36	48	51	45	48
FFNN (with cohort)	31	22	23	33	32	28	34	30	34	37
FFNN (with factors)	42	35	36	37	35	36	38	39	39	38
FFNN (with factor loadings)	6	2	5	4	4	3	1	2	1	4
FFNN (200 epochs)	22	17	18	19	12	10	16	14	15	13
GRU (with cohort, no dropout)	55	56	53	57	54	**56**	58	**62**	**60**	53
GRU (with cohort)	39	29	18	26	19	20	20	22	17	25
LSTM (with cohort, no dropout)	58	51	57	56	54	49	56	49	50	49
LSTM (with cohort)	**61**	52	45	43	41	36	36	30	33	37
RNN (with cohort, no dropout)	**61**	**60**	58	64	58	**56**	**61**	58	59	**60**
RNN (with cohort)	60	**60**	**62**	**65**	**60**	**56**	57	54	58	55
Single population machine learning models										
Boosting (sp)	28	27	28	25	21	24	15	20	24	22
RF (sp)	4	1	4	2	1	2	1	1	1	5
SVM (sp)	1	0	1	1	1	1	0	0	1	1
XGB (sp)	12	5	9	3	5	6	6	4	5	7
Multi population factor model										
Multi level factor	25	19	16	23	18	17	16	15	16	17
Multi population machine learning models										
Boosting (mp)	12	16	13	17	8	15	12	14	17	13
RF (mp)	0	0	1	5	3	3	0	1	5	7
SVM (mp)	18	12	16	16	9	7	7	4	4	2
XGB (mp)	7	2	5	5	1	4	3	4	3	1
Single population factor models										
APC	13	17	12	13	11	8	10	14	11	15
CBD	6	3	3	3	1	2	0	0	2	6
LC (GLM)	20	17	21	19	18	14	16	12	19	19
LC (SVD)	18	13	19	21	18	15	16	14	20	16
M6	10	7	10	11	10	12	11	11	14	15
M7	5	3	5	6	3	3	1	2	2	1
M8	3	0	3	2	1	2	0	0	1	0
Plat	4	5	3	3	1	2	1	1	1	3
RH	37	34	32	36	32	26	33	32	29	30

Table 4: Individual age MCS results for ages 70-79.

	Age									
Deep learning models	80	81	82	83	84	85	86	87	88	89
FFNN (2000 epochs)	47	48	49	52	50	51	56	57	60	59
FFNN (500 epochs)	48	41	49	47	48	49	52	45	52	55
FFNN (with cohort)	34	29	37	44	42	50	49	53	62	58
FFNN (with factors)	42	36	42	49	46	48	52	53	54	55
FFNN (with factor loadings)	3	4	5	1	3	2	2	3	3	4
FFNN (200 epochs)	18	19	19	28	23	23	30	34	36	40
GRU (with cohort, no dropout)	57	54	61	63	63	56	53	56	44	46
GRU (with cohort)	20	19	30	51	55	58	65	65	**66**	**63**
LSTM (with cohort, no dropout)	56	48	58	63	62	59	64	61	61	61
LSTM (with cohort)	47	38	53	59	48	54	54	57	55	54
RNN (with cohort, no dropout)	57	**62**	**66**	**64**	**66**	**62**	**66**	**66**	64	59
RNN (with cohort)	**65**	60	64	63	65	59	60	58	63	**63**
Single population machine learning models										
Boosting (sp)	20	15	22	17	14	8	10	8	9	6
RF (sp)	5	5	3	3	1	1	0	1	4	7
SVM (sp)	1	0	1	1	0	0	1	4	3	6
XGB (sp)	11	4	4	5	3	2	4	4	11	7
Multi population factor model										
Multi level factor	13	13	14	15	21	21	25	29	33	38
Multi population machine learning models										
Boosting (mp)	18	17	20	24	15	10	4	5	6	6
RF (mp)	9	8	20	33	5	0	1	7	8	15
SVM (mp)	3	1	1	1	1	3	3	4	5	9
XGB (mp)	6	3	4	3	0	3	0	4	5	8
Single population factor models										
APC	20	26	30	44	46	49	49	49	51	49
CBD	7	8	8	16	21	29	38	41	46	48
LC (GLM)	19	15	19	26	32	34	39	39	40	39
LC (SVD)	16	14	17	20	26	28	39	36	41	43
M6	22	19	23	28	26	30	36	37	43	40
M7	2	4	6	14	18	26	35	41	42	36
M8	1	0	0	0	0	0	2	2	7	7
Plat	2	6	7	8	13	21	28	42	50	51
RH	29	25	25	29	26	28	27	31	33	34

Table 5: Individual age MCS results for ages 80-89.

	Age					
Deep learning models	90	91	92	93	94	95
FFNN (2000 epochs)	60	62	63	62	60	65
FFNN (500 epochs)	55	56	66	64	63	63
FFNN (with cohort)	63	**68**	**67**	65	**66**	**68**
FFNN (with factors)	56	57	59	61	62	61
FFNN (with factor loadings)	8	15	14	23	27	30
FFNN (200 epochs)	50	51	53	54	54	53
GRU (with cohort, no dropout)	46	46	50	62	50	66
GRU (with cohort)	58	55	49	55	49	52
LSTM (with cohort, no dropout)	60	62	64	**66**	65	65
LSTM (with cohort)	61	57	59	58	58	57
RNN (with cohort, no dropout)	**65**	60	59	64	59	65
RNN (with cohort)	61	54	56	59	59	52
Single population machine learning models						
Boosting (sp)	8	16	11	20	26	35
RF (sp)	12	13	14	30	31	53
SVM (sp)	9	19	22	34	55	69
XGB (sp)	12	15	16	18	23	35
Multi population factor model						
Multi level factor	48	48	54	58	56	53
Multi population machine learning models						
Boosting (mp)	7	11	10	11	20	26
RF (mp)	29	38	46	59	55	48
SVM (mp)	13	25	34	46	48	52
XGB (mp)	10	14	12	29	24	39
Single population factor models						
APC	41	37	36	28	26	22
CBD	57	54	52	51	48	47
LC (GLM)	53	57	57	61	60	58
LC (SVD)	54	52	55	56	51	48
M6	50	49	41	40	40	33
M7	35	32	29	28	24	26
M8	13	15	14	16	17	16
Plat	56	63	60	59	56	57
RH	42	37	41	42	41	40

Table 6: Individual age MCS results for ages 90-95.

$46 \cdot 69 = 3174$ because there are 46 individual ages and 69 analyzed populations.

From table 7, it can be seen that the deep learning models are performing better than the single-population and multi-level factor models. Here it can also be noted that the multi-level factor model also slightly improves performance over the highest performing LC single population model. The best model, the FFNN model with a high number of epochs, is found in the superior set of models 81.06% of the time. From these results, it can be seen that each of the models has strengths and weaknesses regarding performance for different ages. The most robust model, in this case, is the FFNN model.

The linear models tend to perform poorly for ages where the trend is no longer linear. This can be seen with the poor performance of the linear models at the ages 70 - 85. These ages are essential for pension calculations, and therefore accurate forecasts for older ages are vital. The linear models tend to overestimate mortality in the forecasting years 2000 - 2016 since the downward trend in these years tends to be steeper than the estimation years. It can also be seen in fig. 8, where the trend changes. Without extrapolation from other countries, then a shift in trend cannot be accounted for. In fig. 6, it can be noted for

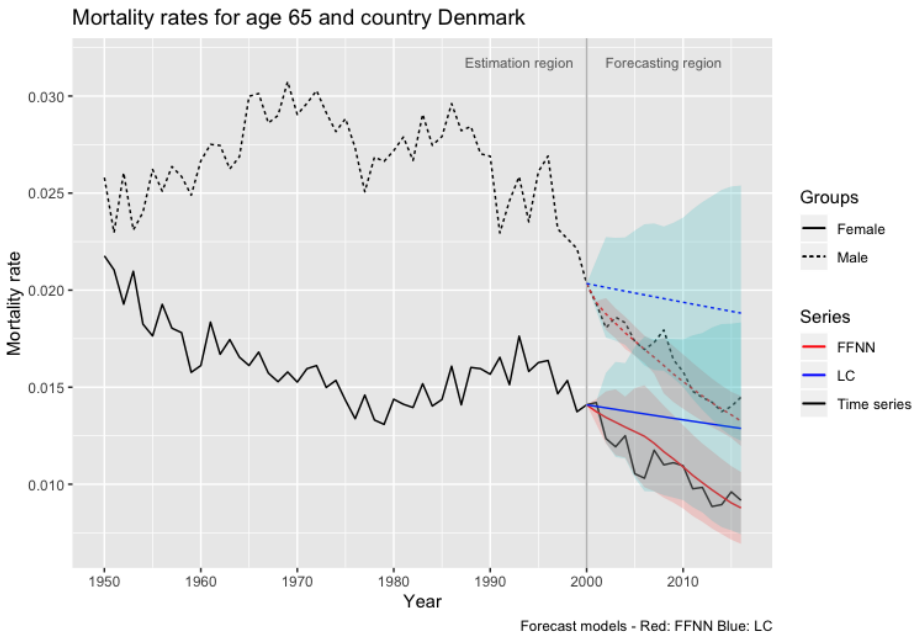

Figure 6: Mortality rates for Danish (DNK) males and females for age 65, together with forecasts and 95% *confidence bands from the LC model (blue) and neural network model (red). The forecasts are made for the years 2000-2016.*

the Danish males that the linear LC model makes poor predictions for the future mortality rates, whereas the more non-linear FFNN is better suited for this task. From fig. 6, it is observed that the FFNN model is better at fitting to the mortality rates. As mentioned, when there is no clear trend or a non-linear trend, the single-population models have a poor fit. Some countries have a more irregularly behaved mortality rate, such as Slovakia. As seen in fig. 8, the mortality rates only have a weak trend in the estimation years, and the neural network model extrapolates from the behavior of other countries to make a more precise forecast. The LC model, which only uses information from a single population, cannot extrapolate from other populations, and it can be seen from fig. 8 that it predicts rising mortality rates for males. The neural network deep learning models can also over extrapolate from the information of other countries. In fig. 8, the neural network model over extrapolates on the

Deep learning models	**# Number of times in MCS**	**% in MCS**
FFNN (2000 epochs)	**2517**	**81.06%**
FFNN (500 epochs)	2496	80.39%
FFNN (with cohort)	1909	61.48%
FFNN (with factors)	2410	77.62%
FFNN (with factor loadings)	828	26.67%
FFNN (200 epochs)	1470	47.34%
GRU (with cohort, no dropout)	2150	69.24%
GRU (with cohort)	1755	56.52%
LSTM (with cohort, no dropout)	2196	70.72%
LSTM (with cohort)	1976	63.64%
RNN (with cohort, no dropout)	2280	73.43%
RNN (with cohort)	2235	71.98%
Single population machine learning models		
Boosting (sp)	1036	33.37%
RF (sp)	363	11.69%
SVM (sp)	262	8.44%
XGB (sp)	877	28.24%
Multi population factor model		
Multi level factor	1756	56.55%
Multi population machine learning models		
Boosting (mp)	917	29.53%
RF (mp)	407	13.11%
SVM (mp)	732	23.57%
XGB (mp)	840	27.05%
Single population factor models		
APC	1572	50.63%
CBD	1458	46.96%
LC (GLM)	1683	54.20%
LC (SVD)	1634	52.62%
M6	1272	40.97%
M7	790	25.44%
M8	663	21.35%
Plat	1356	43.67%
RH	1523	49.05%

Table 7: Individual age MCS results summarized.

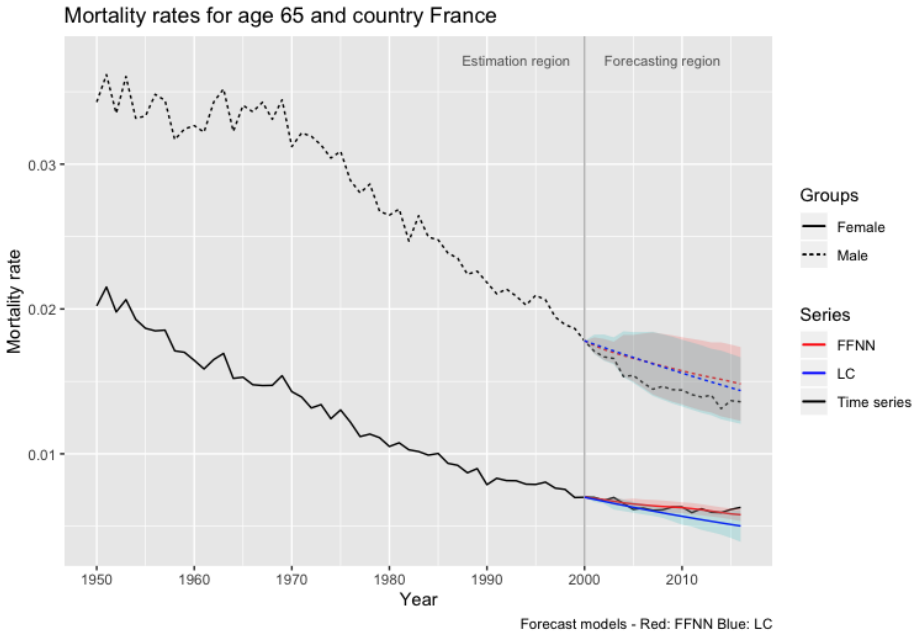

Figure 7: Mortality rates for French (FRA) males and females for age 65, together with forecasts and 95% *confidence bands from the LC model (blue) and neural network model (red). The forecasts are made for the years 2000-2016.*

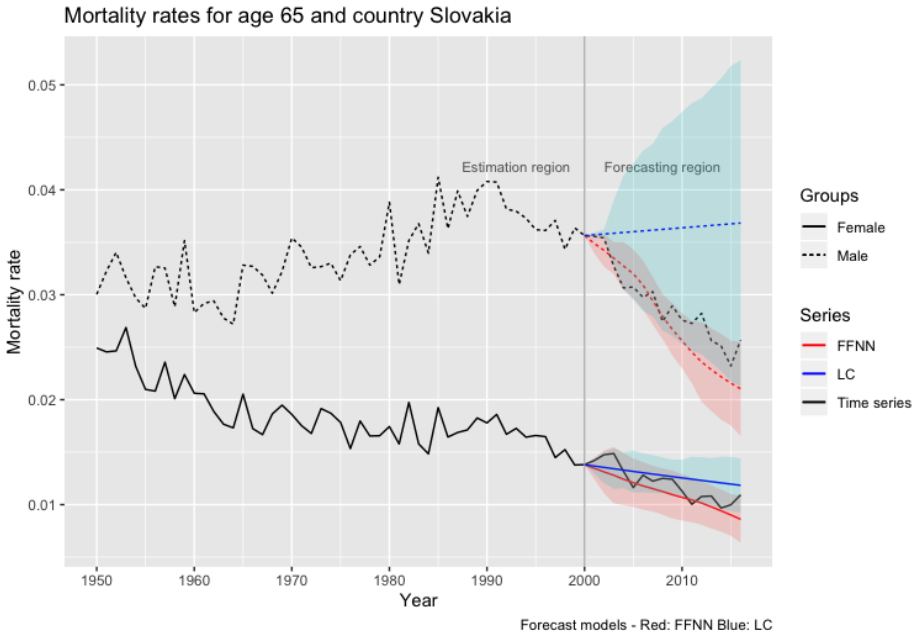

Figure 8: Mortality rates for Slovakian (SVK) males and females for age 65, together with forecasts and 95% *confidence bands from the LC model (blue) and neural network model (red). The forecasts are made for the years 2000-2016.*

information from other countries, and the forecast on males is too steep compared to the real mortality rates in the forecasting period. On the other hand, the LC model follows the upward trend seen in the estimation years and forecast a rising mortality trend, which leads to poor results.

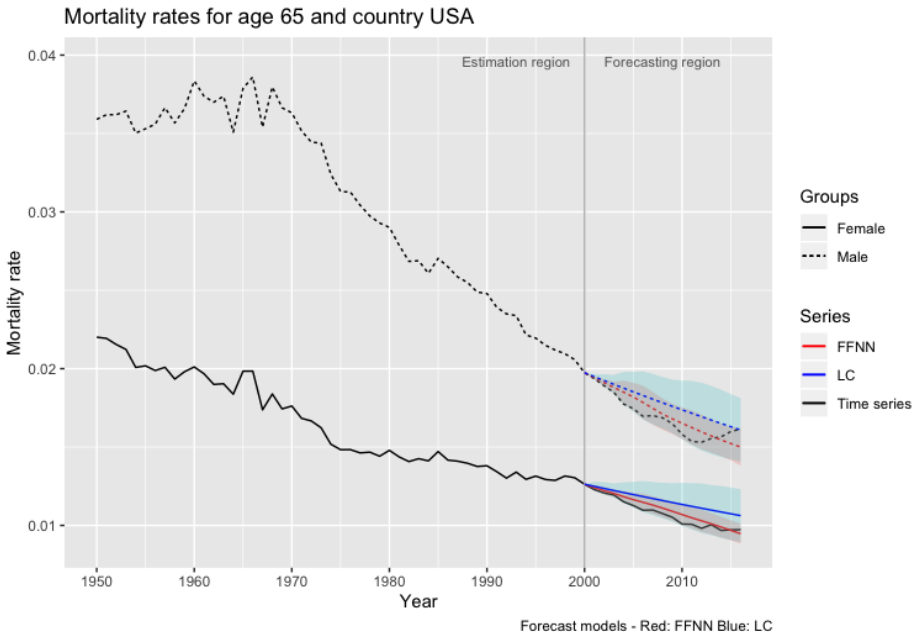

Figure 9: Mortality rates for American (SVK) males and females for age 65, together with forecasts and 95% confidence bands from the LC model (blue) and neural network model (red). The forecasts are made for the years 2000-2016.

For countries with a clearer trend and well-behaved mortality rates, such as France and the US, the linear LC model and FFNN model make very similar forecasts. This can be seen in fig. 7 and fig. 9. Here the uncertainty in the two models is also very similar, and the confidence bands can be seen to be almost on top of each other. For countries where the trend is unclear, and the mortality rate is more volatile, then the LC model has much larger confidence bands than the FFNN model. This can be seen in fig. 6 and fig. 8. Therefore, the simple LC model can fail to include information from other populations, which can provide valuable information when the trend from previous years is unclear and irregularly behaved.

8.2 Mortality projections and policy evaluation

A comparison can also be made between the mortality projections of the baseline LC model and the FFNN model. The FFNN model and LC model are re-estimated with all the available data (1950 - 2016). Forecasted mortality projections are made for the years 2017 - 2060. The graphs are adjusted for jump-off rate bias such that they begin where the actual mortality rates stop. Most countries such as Denmark, Sweden, the Netherlands, and Slovakia have a close link between mortality, life expectancy, and pension systems (OECD, 2017). The link can either be between the pension payment and the expected mortality age, or between statutory retirement age and the expected mortality age. Denmark and the Netherlands use the latter method to regulate the retirement age and have a fixed payout period after retirement (OECD, 2012). When the mortality rate for a particular age is falling, especially for the people at the current pension age, this is an indicator of a longer life

expectancy. Thus the retirement age should be higher. One can further discuss if the schemes should be different between the genders, as indicated by the difference in mortality rates seen in fig. 10, fig. 11, fig. 12, and fig. 13. Likewise, as discussed in Kjærgaard et al. (2019a), socio-economic status could also play a role in mortality differences, which points toward having different retirement ages if the scheme is to have a fixed payout timeline. Denmark implemented a pension scheme in 2007, that aims to increase pension age in line with lower mortality rates and higher life expectancies. It is done to keep the pension payout timeline fixed to 14.5 years (Finansministeriet, 2017). This is a national strategy regardless of differences in genders and socio-economic subgroups or other differentials between sub-groups. If there are significant differences between groups, it can be discussed whether a non-differencing strategy is fair. If the group with the highest mortality rate gets the same retirement age as the group with the lowest mortality rate, then it would mean that more people in that group get fewer years with pension payments on average than those in the lowest mortality rate group. Differences matter for the distributional consequences of a pension system.

The mortality projections can then be used to study if there is an expected convergence between sub-groups that could indicate distributional changes for the pension system and convergence towards a fairer system. Looking at fig. 10, there is a small

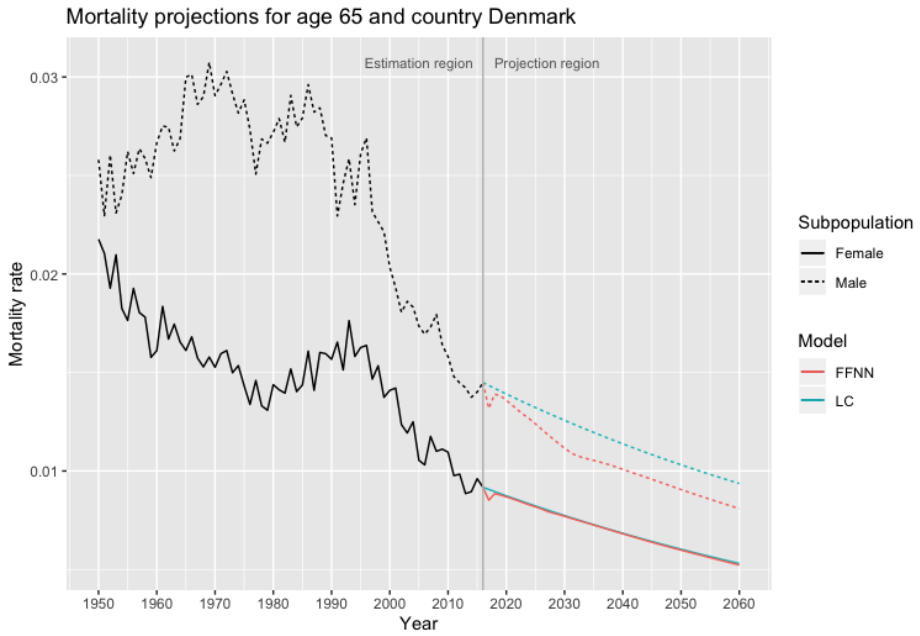

Figure 10: Mortality rates projections for 65 year old Danish males and females, using the LC (blue) and FFNN (red) models.

convergence between the two curves for males and females over time for the projections of the FFNN model. The males are expected to have a larger decline in the mortality rate, which is in line with them benefiting more from medical advances on large diseases that they are more affected by. On the other hand, the expected convergence is much larger in Slovakia, as can be seen in fig. 11, where the mortality rate for males is expected to have a much larger drop towards the same mortality rate as females. The projections for the LC model is less converging than those of the FFNN model. The FFNN model can utilize information across all countries to find a general falling trend and a larger falling trend for males than females. Thus, the FFNN model projects that a common system between groups is fairer than the linear LC model. From fig. 6 and fig. 8 there is also a much larger uncertainty in the LC model estimates than the FFFN model estimates. Accounting for the differences

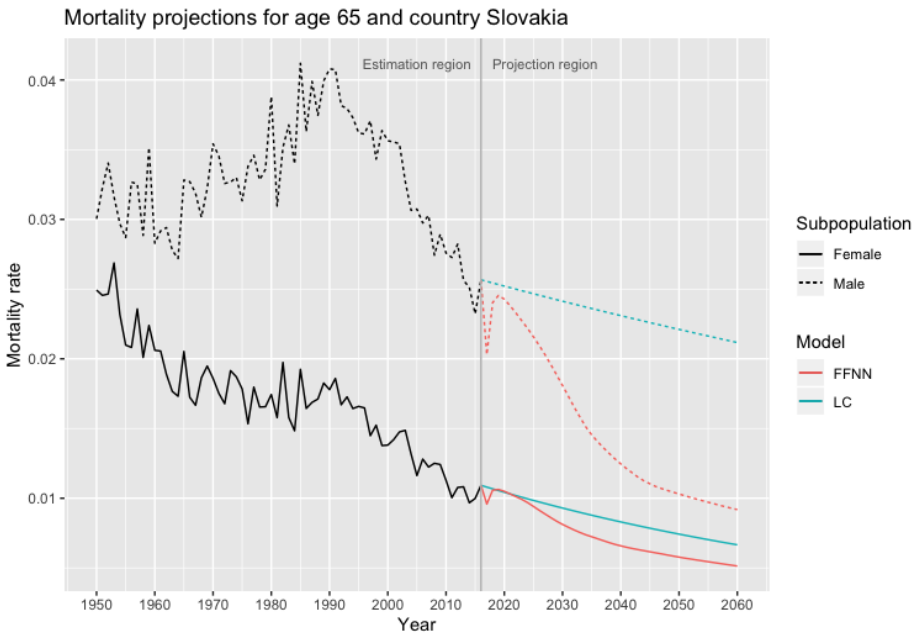

Figure 11: Mortality rates projections for 65 year old Slovakian males and females, using the LC (blue) and FFNN (red) models.

between groups can, therefore, help pension companies with longevity risk Kjærgaard et al. (2019a). Especially for countries with similar behavior as Slovakia (fig. 11), can the importance of incorporating information from other countries be seen. It is more likely that the trend will be falling, rather than continue to increase because they adapt to medical advances. For

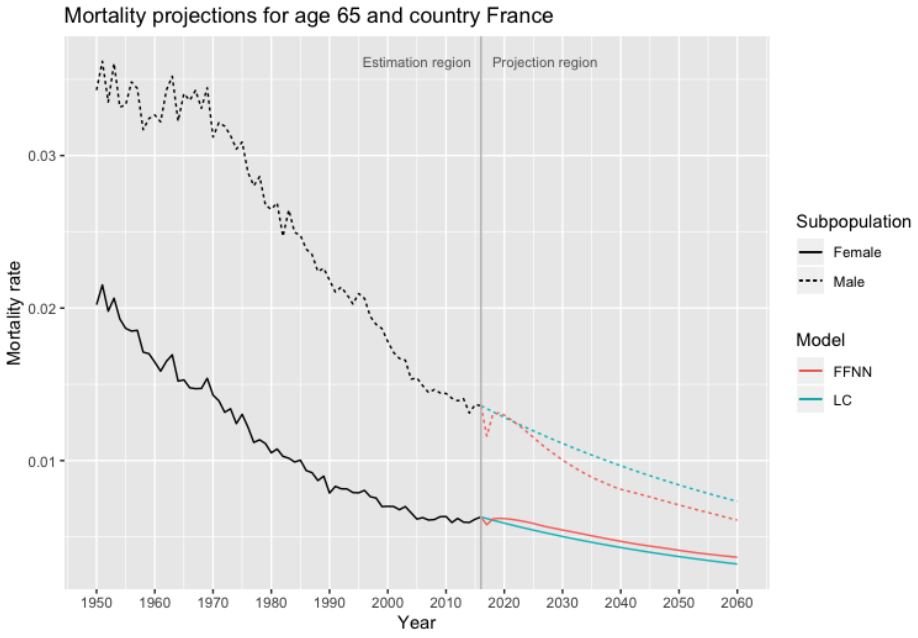

Figure 12: Mortality rates projections for 65 year old French males and females, using the LC (blue) and FFNN (red) models.

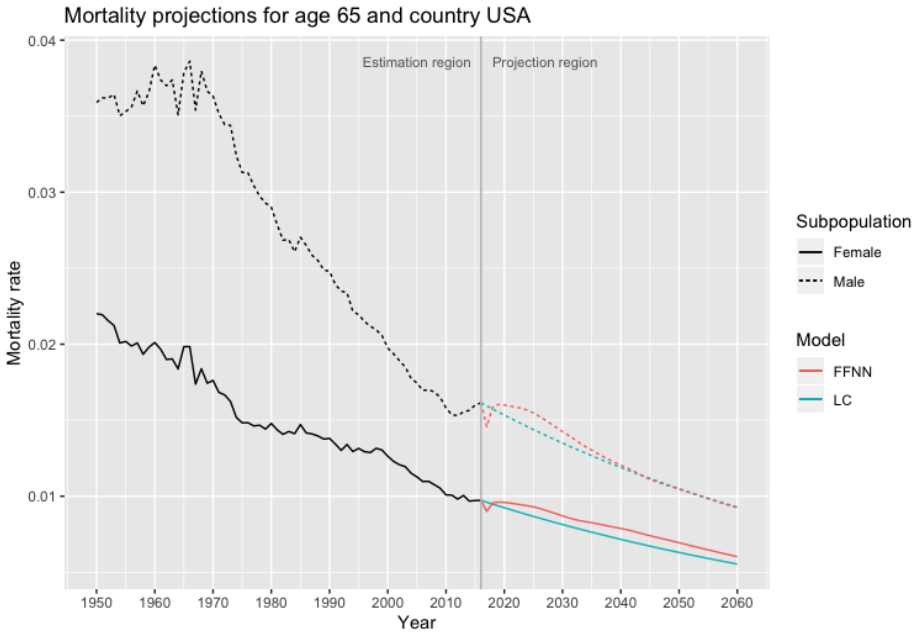

Figure 13: Mortality rates projections for 65 year old American males and females, using the LC (blue) and FFNN (red) models.

France and the US, the projections of the LC and FFNN models resemble each other, as can be seen in fig. 12 and fig. 13, respectively. The trends are more linear, and the extrapolation from the historical trend in the LC model results in similar projections as the FFNN model. As can be seen in fig. 12 and fig. 13, there is a small convergence between males and females in these countries. The study of these four countries indicates a convergence between the two genders' mortality rates, which suggests that a unified scheme in the long-run is fair.

9 Conclusion and Outlook

In this thesis an extensive empirical comparison of current mortality forecasting models is made, together with the introduction of new methods and models that show better performance than current popular mortality forecasting models. The study shows that state-of-the-art deep learning models can be successfully transferred to the mortality forecasting problem to make improved forecasts. The simple linear models can fail to capture stagnating mortality rates and other non-linear mortality trends, for example, a global pandemic. The presented multi-level factor model and deep learning models are new in the mortality forecasting literature. These models showed the importance of using several populations to extrapolate knowledge amongst populations to make a stronger and more robust forecast for each population. The thesis additionally presented several machine learning models, such as random forest, boosting, SVM, and deep neural networks. A thorough comparison of the classical factor models, and machine learning models showed that representation learning is essential for mortality forecasting models. The Random forest, boosting, SVM models failed to capture the representations thoroughly. Their ability to transform the feature space could not completely capture the mortality patterns as well as the classical factor models, such as the LC model. These models can instead be useful as an addition to the factor models, such as presented in Deprez et al. (2017); Levantesi and Nigri (2019). Deep neural networks, which is a deep learning technique, has very high representational

learning abilities, and it was found that these models were the highest performing models, outperforming the LC model and its extensions. Another advantage of the models presented in this thesis is that they do not require additional health-domain knowledge, such as the multi-population Li-Lee model. Most current multi-population models require domain knowledge, to avoid diverging populations being estimated together. The models' performance was compared using out-of-sample forecasting and using the model confidence-set approach, which gives a formal test of the models' predictive ability. The thesis also showed how to incorporate time-dependencies into the high-performing deep learning models, using recurrent neural networks. The additional recurrent layer adds complexity to the model but was not found to increase performance over the feed-forward neural networks. The deep learning models outperformed all other models presented in this thesis. The use of deep learning models will only become more relevant in the future when additional variables, information, and data becomes available. From the results found in this thesis, many future research areas can potentially also benefit from deep learning methods.

Several papers point towards the need for the widely popular single-population mortality forecasting models to incorporate additional information or use other populations mortality history to extrapolate from (Brouhns et al., 2002a; Li and Lee, 2005; Richman and Wüthrich, 2018; Murtin et al., 2017; Levantesi and Nigri, 2019; Deprez et al., 2017; Stoeldraijer et al., 2013). Brouhns et al. (2002a) points out that the LC model and the extensions hereof does not make use of any additional information besides the mortality rates. Therefore, it is unable to forecast sudden improvements in mortality due to the discovery of new medical treatments. Similarly, as mentioned, future deteriorations caused by pandemics or the apparition of new diseases cannot enter the model. Therefore, the currently most popular models in mortality forecasting literature are purely an extrapolation of past trends. It has long been a critique point of these models (Brouhns et al., 2002a; Gutterman and Vanderhoof, 1998). It seems more relevant than ever to incorporate additional information, with the rapidly increasing data growth and current world situation. It will also be ever more relevant to use methods that can handle the enormous amounts of health-data (Stanford, 2017). Most countries produce large quantities of health data every day. Many countries like Denmark have a central person register number and electronic health records that all information is linked to, giving new possibilities for making models such as mortality forecasting models even stronger. Besides the considerable potential deep learning has shown in this thesis with regards to mortality forecasting, another reason that it is gaining more attention in the field is that it can handle the unstructured nature of health data. As pointed out Davenport and Kalakota (2019), the potential reach much further than mortality forecasting. Deep learning can also be used to understand further differences between socio-economic groups. Murtin et al. (2017); Kallestrup-Lamb et al. (2019); Kjærgaard et al. (2019a) found differences between groups, as well as Case and Deaton (2015), that found an increase in all-cause mortality for middle-class groups. Doctor notes and diagnosis from the electronic health records can be used to gather information on symptoms using natural language processing and deep learning. It can give valuable insights into the relationship between socio-economic status and diagnosis. Schwartz et al. (2003) found that socio-economic status plays a significant role in which stage cancer is detected, and a better understanding of the differences between these groups and how they are treated could potentially help reduce inequalities and mortality for more at-risk groups.

The current world situation facing a global pandemic, models better suited for such special situations are needed. It becomes imperative for policy decision-makers to evaluate the projected course of the pandemic. Goodfellow et al. (2014) proposed the generative adversarial network (GAN) as a deep learning model that has proven powerful at generating data (Che et al., 2017; Choi et al., 2017). The GAN model uses two deep neural networks adversarial to each other. One is the generator that generates new data and the other is the discriminative network that detects if data is generated or real. Competing against each

other creates a powerful model for generating new data and, thereby, projected timelines. When data is scarce, this can be very helpful, either to the models' projections on their own, but also to use the projections as data in other models to boost their predictive ability. Thus, it is possible to get a quicker understanding of how new diseases progress over time and make more robust policy decisions.

In mortality forecasting, recent advances can equally be transferred over to the deep learning models. This thesis showed improved mortality forecasts using deep learning models. Recent years have shown several advances in cause-specific mortality forecasting. One such improvement was made by Kjærgaard et al. (2019b), that utilized compositional data analysis (CoDa) to improve cause-specific forecasting models. CoDa consists of vectors where each component is a percentage of some whole (Atchinson, 2005). Elements in the CoDa vectors are therefore positive, and the elements sum to a constant value (Kjærgaard et al., 2019b). This method can be transferred to deep learning models, which can potentially improve current cause-specific mortality models.

Therefore, the improvements found in this thesis using deep learning models for mortality forecasting with the currently available data is only the first step in the potential use for AI in health, demography, and health-economics, where the potential is enormous Regeringen et al. (2019); Davenport and Kalakota (2019); Caroprese et al. (2018).

References

Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). IEEE Access, 6:52138–52160, 2018.

Charu Aggarwal. Neural networks and deep learning : a textbook. Springer, Cham, Switzerland, 2018. ISBN 978-3-319-94463-0.

Alan Agresti. Foundations of linear and generalized linear models. John Wiley & Sons Inc, Hoboken, New Jersey, 2015. ISBN 978-1-118-73003-4.

J Atchinson. A concise guide to compositional data analysis. In 2do Compositional Data Analysis Workshop CoDaWork, volume 5, pages 17–21, 2005.

Mariette Awad and Rahul Khanna. Support vector regression. In Efficient Learning Machines, pages 67–80. Springer, 2015.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks, 5(2):157–166, 1994.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence, 35(8):1798–1828, 2013.

Mauro Bernardi and Leopoldo Catania. The model confidence set package for r. 2015.

Tim Bollerslev, Robert F Engle, and Daniel B Nelson. Arch models. Handbook of econometrics, 4:2959–3038, 1994.

Heather Booth. Demographic forecasting: 1980 to 2005 in review. International journal of forecasting, 22(3):547–581, 2006.

Heather Booth and Leonie Tickle. Mortality modelling and forecasting: A review of methods. Annals of actuarial science, 3 (1-2):3–43, 2008.

Heather Booth, Rob J Hyndman, Leonie Tickle, and Piet De Jong. Lee-carter mortality forecasting: a multi-country comparison of variants and extensions. Demographic research, 15:289–310, 2006.

Natacha Brouhns, Michel Denuit, and Jeroen K Vermunt. A poisson log-bilinear regression approach to the construction of projected lifetables. Insurance: Mathematics and economics, 31(3):373–393, 2002a.

Natacha Brouhns, Michel Denuit, Jeroen K Vermunt, et al. Measuring the longevity risk in mortality projections. Bulletin of the Swiss Association of Actuaries, 2:105–130, 2002b.

Andrew JG Cairns, David Blake, and Kevin Dowd. A two-factor model for stochastic mortality with parameter uncertainty: theory and calibration. Journal of Risk and Insurance, 73(4):687–718, 2006.

Andrew JG Cairns, David Blake, Kevin Dowd, Guy D Coughlan, David Epstein, Alen Ong, and Igor Balevich. A quantitative comparison of stochastic mortality models using data from england and wales and the united states. North American Actuarial Journal, 13(1):1–35, 2009.

Luciano Caroprese, Pierangelo Veltri, Eugenio Vocaturo, and Ester Zumpano. Deep learning techniques for electronic health record analysis. In 2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA), pages 1–4. IEEE, 2018.

Anne Case and Angus Deaton. Rising morbidity and mortality in midlife among white non-hispanic americans in the 21st century. Proceedings of the National Academy of Sciences, 112(49):15078–15083, 2015.

Zhengping Che, Yu Cheng, Shuangfei Zhai, Zhaonan Sun, and Yan Liu. Boosting deep learning risk prediction with generative adversarial networks for electronic health records. In 2017 IEEE International Conference on Data Mining (ICDM), pages 787–792. IEEE, 2017.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pages 785–794, 2016.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259, 2014a.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014b.

Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. arXiv preprint arXiv:1703.06490, 2017.

In Choi, Dukpa Kim, Yun Jung Kim, and Noh-Sun Kwark. A multilevel factor model: Identification, asymptotic theory and applications. Journal of Applied Econometrics, 33(3):355–377, 2018.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555, 2014.

Ryan Cotterell, Adam Poliak, Benjamin Van Durme, and Jason Eisner. Explaining and generalizing skip-gram through exponential family principal component analysis. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 175–181, 2017.

SF Crosbie and GN Hinch. An intuitive explanation of generalised linear models. New Zealand journal of agricultural research, 28(1):19–29, 1985.

Iain D Currie. Smoothing and forecasting mortality rates with p-splines. Talk given at the Institute of Actuaries, 2006.

Iain D Currie. On fitting generalized linear and non-linear models of mortality. Scandinavian Actuarial Journal, 2016(4): 356–383, 2016.

Thomas Davenport and Ravi Kalakota. The potential for artificial intelligence in healthcare. Future healthcare journal, 6(2): 94, 2019.

Philippe Deprez, Pavel V Shevchenko, and Mario V Wüthrich. Machine learning techniques for mortality modeling. European Actuarial Journal, 7(2):337–352, 2017.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608, 2017.

Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. Communications of the ACM, 63(1): 68–77, 2019.

Vasil Enchev, Torsten Kleinow, and Andrew JG Cairns. Multi-population mortality models: fitting, forecasting and comparisons. Scandinavian Actuarial Journal, 2017(4):319–342, 2017.

European-Commission, WHO, OECD, European Observatory on Health Systems, and Policies. State of health in the eu: Denmark country health profile 2017. 2017.

Finansministeriet. Det danske pensionssystem nu og i fremtiden. 2017.

Ian Goodfellow. Deep learning. The MIT Press, Cambridge, Massachusetts, 2016. ISBN 978-0262035613.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

Sam Gutterman and Irwin T Vanderhoof. Forecasting changes in mortality: a search for a law of causes and effects. North American Actuarial Journal, 2(4):135–138, 1998.

Steven Haberman and Arthur Renshaw. A comparative study of parametric mortality projection models. Insurance: Mathematics and Economics, 48(1):35–55, 2011.

Donatien Hainaut. A neural-network analyzer for mortality forecast. ASTIN Bulletin: The Journal of the IAA, 48(2):481–508, 2018.

Peter R Hansen and Asger Lunde. A forecast comparison of volatility models: does anything beat a garch (1, 1)? Journal of applied econometrics, 20(7):873–889, 2005.

Peter R Hansen, Asger Lunde, and James M Nason. The model confidence set. Econometrica, 79(2):453–497, 2011.

Trevor Hastie. The elements of statistical learning : data mining, inference, and prediction. Springer, New York, 2009. ISBN 0387848576.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

OECD Indicators. Health at a glance 2013. 15, 2014.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

Gareth James. An introduction to statistical learning : with applications in R. Springer, New York, NY, 2013. ISBN 978-1-4614-7137-0.

Fanny Janssen. Advances in mortality forecasting: introduction, 2018.

Malene Kallestrup-Lamb, Søren Kjærgaard, Carsten PT Rosenskjold, et al. Insight into stagnating life expectancy: Analysing cause of death patterns across socio-economic groups. Technical report, Department of Economics and Business Economics, Aarhus University, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Søren Kjærgaard, Yunus Emre Ergemen, Marie-Pier Bergeron Boucher, Jim Oeppen, Malene Kallestrup-Lamb, et al. Longevity forecasting by socio-economic groups using compositional data analysis. Technical report, Department of Economics and Business Economics, Aarhus University, 2019a.

Søren Kjærgaard, Yunus Emre Ergemen, Malene Kallestrup-Lamb, Jim Oeppen, and Rune Lindahl-Jacobsen. Forecasting causes of death by using compositional data analysis: the case of cancer deaths. Journal of the Royal Statistical Society: Series C (Applied Statistics), 2019b.

Anders Bredahl Kock, Timo Teräsvirta, et al. Forecasting with nonlinear time series models. Oxford handbook of economic forecasting, pages 61–87, 2011.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. nature, 521(7553):436–444, 2015.

Ronald D Lee and Lawrence R Carter. Modeling and forecasting us mortality. Journal of the American statistical association, 87(419):659–671, 1992.

Susanna Levantesi and Andrea Nigri. A random forest algorithm to improve the lee–carter mortality forecasting: impact on q-forward. Soft Computing, pages 1–15, 2019.

Susanna Levantesi and Virginia Pizzorusso. Application of machine learning to mortality modeling and forecasting. Risks, 7 (1):26, 2019.

Johnny Siu-Hang Li, Wai-Sum Chan, and Rui Zhou. Semicoherent multipopulation mortality modeling: The impact on longevity risk securitization. Journal of Risk and Insurance, 84(3):1025–1065, 2017.

Nan Li and Ronald Lee. Coherent mortality forecasts for a group of populations: An extension of the lee-carter method. Demography, 42(3):575–594, 2005.

Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019, 2015.

Paulo JG Lisboa. Interpretability in machine learning–principles and practice. In International Workshop on Fuzzy Logic and Applications, pages 15–21. Springer, 2013.

Christoph Molnar. Interpretable machine learning. Lulu. com, 2019.

Sendhil Mullainathan and Jann Spiess. Machine learning: an applied econometric approach. Journal of Economic Perspectives, 31(2):87–106, 2017.

Fabrice Murtin, Johan Mackenbach, Domantas Jasilionis, and Marco Mira d'Ercole. Inequalities in longevity by education in oecd countries. 2017.

Roger Newson. Review of generalized linear models and extensions by hardin and hilbe. The Stata Journal, 1(1):98–100, 2001.

Didrik Nielsen. Tree boosting with xgboost-why does xgboost win" every" machine learning competition? Master's thesis, NTNU, 2016.

OECD. Oecd pensions outlook 2012. 2012.

OECD. Pensions at a glance. 2017.

Richard Plat. On stochastic mortality modeling. Insurance: Mathematics and Economics, 45(3):393–404, 2009.

Regeringen, Finansministeriet, and Erhvervsministeriet. National strategi for kunstig intelligens. 2019.

Eric N Reither, S Jay Olshansky, and Yang Yang. New forecasting methodology indicates more disease and earlier mortality ahead for today's younger americans. Health Affairs, 30(8):1562–1568, 2011.

Arthur E Renshaw and Steven Haberman. A cohort-based extension to the lee–carter model for mortality reduction factors. Insurance: Mathematics and economics, 38(3):556–570, 2006.

Ronald Richman. Ai in actuarial science. Available at SSRN 3218082, 2018.

Ronald Richman and Mario V Wüthrich. A neural network extension of the lee–carter model to multiple populations. Annals of Actuarial Science, pages 1–21, 2018.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Kendra L Schwartz, Heather Crossley-May, Fawn D Vigneau, Karl Brown, and Mousumi Banerjee. Race, socioeconomic status and stage at diagnosis for five common malignancies. Cancer Causes & Control, 14(8):761–766, 2003.

Syazreen Shair, Sachi Purcal, and Nick Parr. Evaluating extensions to coherent mortality forecasting models. Risks, 5(1):16, 2017.

Benjamin Shickel, Patrick James Tighe, Azra Bihorac, and Parisa Rashidi. Deep ehr: a survey of recent advances in deep learning techniques for electronic health record (ehr) analysis. IEEE journal of biomedical and health informatics, 22(5): 1589–1604, 2017.

Health Stanford. Harnessing the power of data in health. Stanford Medicine 2017 Health Trends Report, 2017.

Lenny Stoeldraijer, Coen van Duin, Leo van Wissen, and Fanny Janssen. Impact of different mortality forecasting methods and explicit assumptions on projected future life expectancy: The case of the netherlands. Demographic Research, 29: 323–354, 2013.

Lenny Stoeldraijer, Coen van Duin, Leo van Wissen, and Fanny Janssen. Comparing strategies for matching mortality forecasts to the most recently observed data: exploring the trade-off between accuracy and robustness. Genus, 74(1):16, 2018.

Richard Sutton. Reinforcement learning : an introduction. MIT Press, Cambridge, Mass, 1998. ISBN 978-0262193986.

Souhaib Ben Taieb, Gianluca Bontempi, Amir F Atiya, and Antti Sorjamaa. A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. Expert systems with applications, 39(8): 7067–7083, 2012a.

Souhaib Ben Taieb, Rob J Hyndman, et al. Recursive and direct multi-step forecasting: the best of both worlds, volume 19. Citeseer, 2012b.

Andres Villegas, Vladimir K Kaishev, and Pietro Millossovich. Stmomo: An r package for stochastic mortality modelling. In 7th Australasian Actuarial Education and Research Symposium, 2015.

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

WJ Willemse and H Koppelaar. Knowledge elicitation of gompertz'law of mortality. Scandinavian Actuarial Journal, 2000 (2):168–179, 2000.

John R Wilmoth and Vladimir M Shkolnikov. Human mortality database. Available at www.mortality.org or www.humanmortality.de. Accessed: 06-02-2020.

Mario V Wuthrich and Christoph Buser. Data analytics for non-life insurance pricing. Swiss Finance Institute Research Paper, (16-68), 2019.

Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into Deep Learning. 2020. https://d2l.ai.

Guoqiang Zhong, Li-Na Wang, Xiao Ling, and Junyu Dong. An overview on data representation learning: From traditional feature learning to recent deep learning. The Journal of Finance and Data Science, 2(4):265–278, 2016.

Appendix

A List of abbreviations

Abbreviation	Meaning
AI	Artificial Intelligence
CCA	Canonical Correlation Analysis
EPA	Equal predictive ability
FFNN	Feed Forward Neural Network
GLM	Generalized Linear Model
GNM	Generalized non-linear model
GRU	Gated Recurrent Unit
LC model	Lee-Carter model
LSTM	Long-Short Term Memory
MCS	Model Confidence Set
PCA	Principal Component Analysis
RF	Random Forest
RNN	Recurrent Neural Network
SVD	Singular Value Decomposition
SVM	Support Vector Machine
XGB	Extreme Gradient Boosting

B Code guide

All code used in the thesis is found in the *Code.zip* attachment to the thesis. The attached *Code.zip* contains three elements.

1. The folder called *Scripts*. This folder contains all the scripts that have been used to get the results using the functions in the package.
2. The folder called *mthesis*. This folder contains the package and the data used in the thesis.
3. The *mthesis_1.0.tar.gz* file. This file also contains the package but is build as a source file ready to install.

The code to the R-package can also be found in the Github repository at the following link: Improved Mortality Forecasts using Artificial Intelligence. There are small differences between the attachment and the Github code, as Github is publicly available to everyone. The Github repository does not include data that is used in the thesis, but contains a guide to getting the needed data instead.

B.1 Installing package dependencies

First, the package dependencies must be installed. This is done with the following code that automatically checks if each dependency package is installed. It will proceed to install the packages that are not installed.

```r
# Define needed packages #
vPackages = c("CatEncoders", "StMoMo",
              "MortalityLaws", "HMDHFDplus",
              "reshape", "RcppArmadillo")

# Loop over the dependency packages #
for (sPackage in vPackages) {

  # Check if the package is installed #
  # If not then install it #
  if(!require(sPackage, character.only = TRUE)){
    install.packages(sPackage)
  }

  # Load package #
  library(sPackage, character.only = TRUE)

}
```

Listing 1: "R code for installing package dependencies"

B.2 Installing package

Code is packed as an R package (mthesis_1.0.tar.gz file), which can be installed by using the command,

```r
# Install the package #
install.packages("path_to_file", repos = NULL, type="source")

# Load the package after installation #
library(mthesis)
```

Listing 2: "R code for installing package"

The path must be replaced with the actual path to the appended package tarball file. The package depends both on R and C++ code. The C++ code is written with Rcpp and RcppArmadillo to integrate it with R.

The package can also be installed from the Github link using the following command:

```r
# Must first install devtools package #
if(!require(devtools)){
  install.packages("devtools")
}

# Load devtools package #
```

```r
library(devtools)

# Install using devtools github function #
devtools::install_github("rune-l/Improved-Mortality-Forecasts-Using-Artificial-Intelligence/tree/master/mthesis")

# Load the package after installation #
library(mthesis)
```

Listing 3: "R code for installing package using Github link"

It must be noted that the Github code does not include any data. This must be retrieved from the Human Mortality Database (Wilmoth and Shkolnikov).

C Recurrent model structure

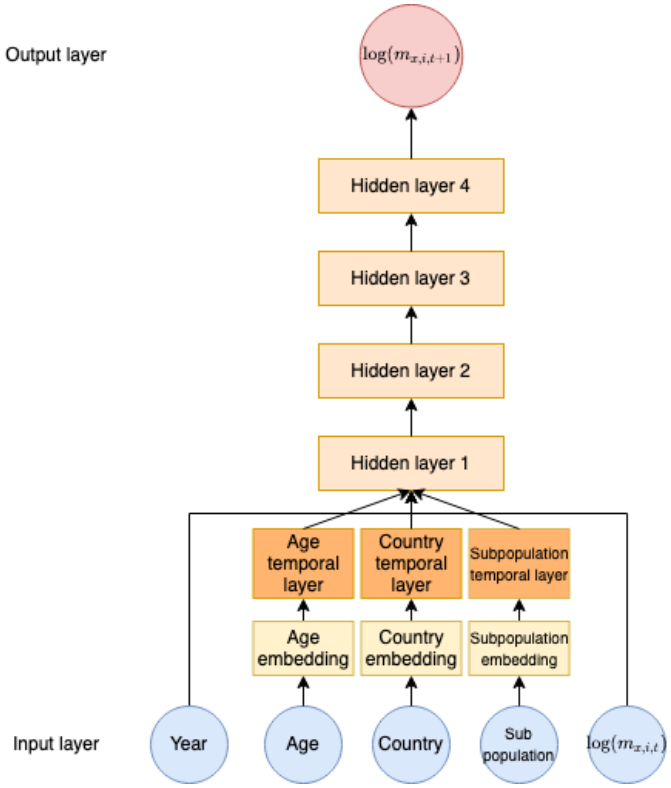

Figure 14: Proposed deep recurrent neural network model structure.

D Generalized Linear Models (GLM)

Currie (2016) explains in great detail how the single population models proposed in section 3 can be fitted using the GLM framework. Here GLMs are explained to understand how they work and how they can be used. It can provide a broader understanding of the single-population models.

GLMs can be defined as any model where the conditional mean of the outcome variable Y is transformable to a linear combination of the X variables (using a link function). Furthermore, the variance of Y is proportional to a function of the mean. The set of functions that use the same error distribution is called a family of generalized linear models (Newson, 2001). The family that is used in this thesis is the Poisson distributed family of GLMs. A GLM consists of three main parts

- Random component: The probability distribution of the response variable.
- Linear predictors: The set of X and β parameters.
- Link function: The link function g applied to each component of $E[y]$ such that $g(E[y]) = X\beta$.

When y_i has a Poisson distribution, then the probability mass function is given as

$$f(y_i;\mu_i) = \frac{\exp(-\mu_i)\,\mu_i^{y_i}}{y_i!} = \exp[y_i \cdot \log(\mu_i) - \mu_i - \log(y_i!)]$$

The log-likelihood function is found as,

$$L(\theta) = \sum_{i=1}^{n} \log[f(y_i;\theta_i)] = \sum_{i=1}^{n} [y_i \cdot \log(\mu_i) - \mu_i - \log(y_i!)] \tag{77}$$

which is an ideal distribution for count data (Agresti, 2015). In the mortality forecasting setting with a Poisson distribution, the log-likelihood is given as

$$L\left(d_{xt}, \hat{d}_{xt}\right) = \sum_t \sum_x \left(d_{xt} \log\left(\hat{d}_{xt}\right) - \hat{d}_{xt} - log\left(d_{xt}!\right)\right), \tag{78}$$

where $\hat{d}_{xt} = E_{xt} g^{-1}\left(\alpha_x + \sum_{i=1}^{n} \beta_x^{(i)} k_t^{(i)} + \beta_x^{(0)} \gamma_{t-x}\right)$ is the given model specification, and g^{-1} denotes the inverse of the link function (Villegas et al., 2015). Here d_{xt} is the observed number of deaths, and E_{xt} is the exposure to risk.

Generalized linear models are applicable in situations when data is non-linear and perhaps also non-normal, which means standard linear models are unsuitable for direct application (Crosbie and Hinch, 1985). In GLMs, the linear model is molded to fit the data characteristics. This makes GLMs ideal in mortality modeling, where death counts are assumed to be Poisson distributed.

E Singular Value Decomposition

A singular value decomposition (SVD) of an $m \times n$ matrix A is given as

$$A = UDV', \tag{79}$$

where it holds that

1. U is an $m \times m$ orthonormal matrix.
2. V is an $n \times n$ orthonormal matrix.
3. D is an $m \times n$ diagonal matrix with non-negative elements. Furthermore, the diagonal elements a sorted, starting with the highest value first.

The columns of U are the left singular vectors of A. The columns of V are the right singular vectors, and the diagonal elements of D are the singular values of A. The first singular vector refers to the one associated with the largest singular value. The SVD expresses the matrix A as a non-negative linear combination of $\min(m, n)$ rank-1 matrices, with the singular value being the multiplier and the outer product of the left and right singular vectors being the rank-1 matrices. Rank-1 matrices are matrices where the columns are multiples of each other, and the same for the rows.

One can produce a low-rank approximation of matrices using SVD, keeping only the most essential information, which can be expressed by the k first singular values and singular vectors.

The right singular vectors of A are the same as the eigenvectors of $A'A$. The eigenvalues of $A'A$ are the squares of the singular values. The principal components analysis (PCA) reduces to computing the SVD of A. The SVD $A = UDV'$ gives the eigenvectors, which are used in PCA. The first k eigenvectors are the first k rows of V'. The goal of PCA and SVD is often different. PCA seeks to approximate data points as a linear combination of a small set of vectors, whereas SVD is a general matrix operation defined on all matrices. SVD provides more information in cases where the rows of A are also of importance. In the mortality forecasting problem, both rows and vectors are essential. The rows provide information about ages and columns provide information about time.

F RNN vs FFNN model for 50 - 59 age group

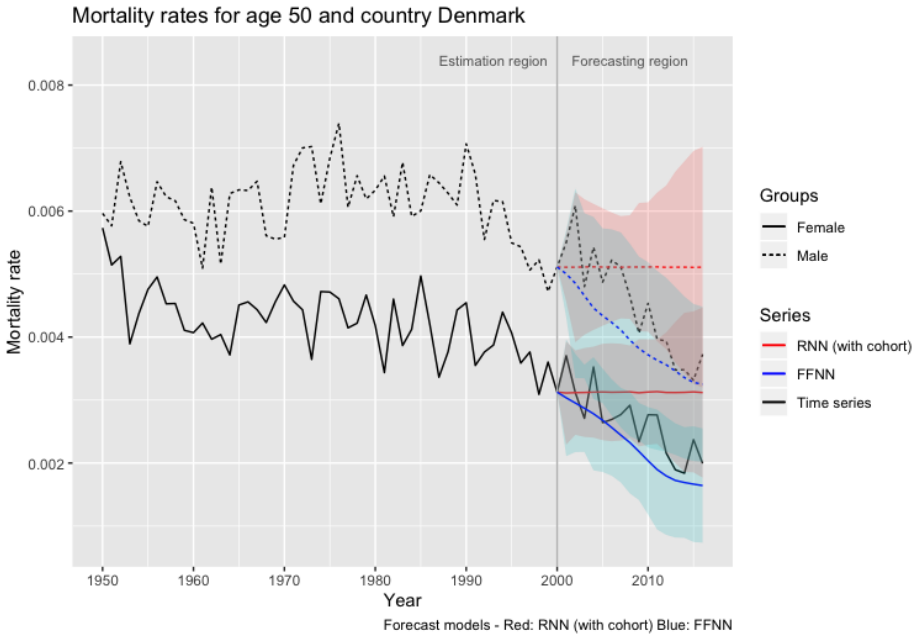

Figure 15: Mortality rates for Danish (DNK) males and females for age 50, together with forecasts from the FFNN (blue) and RNN model (red). The forecasts are made for the years 2000-2016.

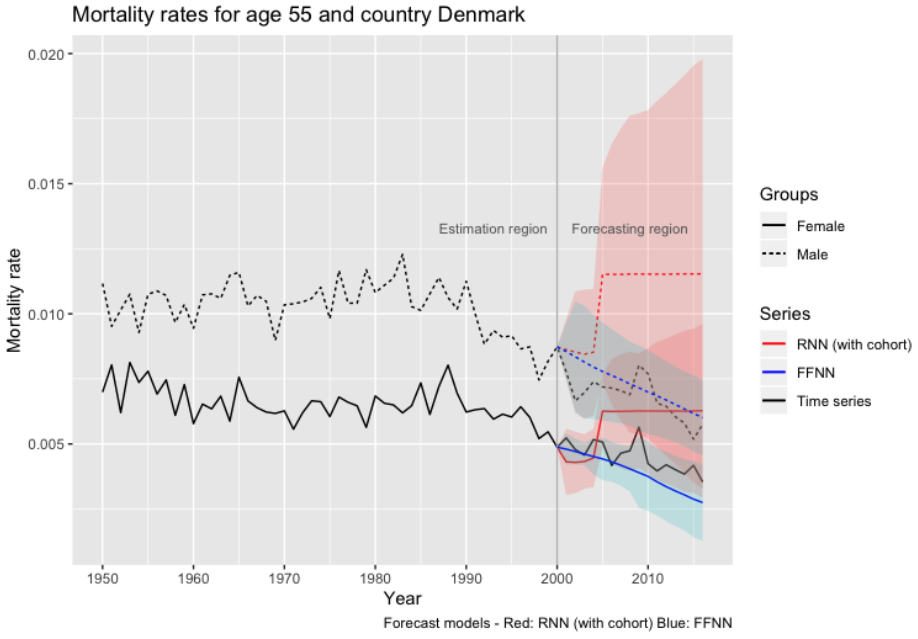

Figure 16: Mortality rates for Danish (DNK) males and females for age 55, together with forecasts from the FFNN (blue) and RNN model (red). The forecasts are made for the years 2000-2016.