The Alan Turing Institute

Data Study Group Final Report: UK Dementia Research Institute

6 – 24 Sep 2021

Using machine learning to improve sleep habits in Dementia patients
# Contents

1 Executive summary .......................... 3
   1.1 Challenge overview ........................ 3
   1.2 Data overview .............................. 4
   1.3 Main objectives ............................ 4
   1.4 Approach .................................. 5
      1.4.1 Predicting sleep parameters (Question 1a) .... 5
      1.4.2 Marginal effects of covariates to sleep measures (Question 1b) .... 6
      1.4.3 Synthetic data generation (Question 2) ........ 6
      1.4.4 Studying intervention (Question 3) ............ 8
         Guaranteeing the existence of counterfactuals ....... 9
         Generating counterfactuals .................... 10
   1.5 Main conclusions .......................... 11
   1.6 Limitations ........................-------- 12
   1.7 Future work ........................-------- 12

2 Data overview .............................. 12
   2.1 Dataset description ........................ 12
   2.2 Data quality issues ........................ 21

3 Experiments ................................. 21
   3.1 Predicting sleep parameters ................. 21
      3.1.1 Predicting time-in-bed using NGBoost .... 21
      3.1.2 Predicting number of wakeups and time-in-bed using time series forecasting .... 24
      3.1.3 Predicting activity ..................... 35
   3.2 Estimating marginal effects .................. 37
      3.2.1 The causal structure .................... 39
      3.2.2 The back-door criterion .................. 39
   3.3 Generating synthetic data in Question 2 ........ 41
      3.3.1 Variational autoencoder to model sleep patterns .... 41
      3.3.2 Model ................................. 43

4 Future work and research avenues ............. 48
   4.1 Exploring data and application .............. 48
      4.1.1 Dataset collation ...................... 48
4.1.2 Feature importance ........................................ 48
4.1.3 Diagnosis decline detection ............................... 49
4.2 Exploring model evaluation strategies ....................... 49
  4.2.1 Measuring prediction uncertainty ....................... 49
  4.2.2 Performance metrics ..................................... 50
4.3 Exploring models ............................................. 50
  4.3.1 Uncertainty-based prediction calibration ............... 50
  4.3.2 Modelling the physical process using Deep Koopmann 51
  4.3.3 Models and hyperparameter optimisation ............... 55
  4.3.4 Generalisation ........................................... 55
  4.3.5 Multiple-multivariate time series forecasting ......... 55

5 Team members ................................................... 56

References .......................................................... 57
1 Executive summary

1.1 Challenge overview

Dementia is a collection of several types of cascading loss of brain functions that start with mild symptoms eventually leading to severe impairment which becomes terminal within a decade after diagnosis. In the UK, there are currently around 850,000 people living with dementia. There are several types of dementia, including Alzheimer’s disease, which accounts for 60-80% of all dementia cases [Ass22].

An estimated 80% dementia patients have sleep problems. Recent research has found sleep disturbances are a risk factor for deteriorating prognosis. Sleep disorders adversely impact the dementia patients’ physical, behavioural, and cognitive functions. Their sleeping problems often distress their caregivers as well, since patients with dementia are usually awake requiring care for uncomfortably long periods.

Studies suggest that to improve sleep in dementia patients, pharmaceutical interventions are less effective. In comparison, non-pharmaceutical interventions have proved effective; light therapy has been successfully used in several pilot studies as means to regulate the circadian rest activity cycle.

Designing such an intervention strategy is challenging due to the multi-dimensional nature of the problem. Multiple factors contribute to sleep problems, for example changes in the patient’s environment, their physical, cognitive and psychiatric conditions, neurodegenerative changes in the brain, medication used for chronic illnesses, and other dementia-related symptoms, among others.

This challenge aims to predict the efficacies of the intervention markers and find the most effective ones which will allow the medical practitioners to take more informed suggestions.

1https://www.england.nhs.uk/mental-health/dementia/
1.2 Data overview

The UK DRI Care and Research Center has compiled a multi-dimensional dataset produced by tracking 38 dementia patients for multiple months. The challenge dataset includes age, type of dementia diagnosed, data from various indoor sensors, outdoor weather, and scores from two clinical tests to quantify the severity of the patients' dementia. The tests are the neuropsychiatric inventory questionnaire (NPI-Q) [KCK+00] and the Mini-Mental State Exam (MMSE).

During the study, a participant's bed was instrumented with a Withing's Sleep Analyzer which is a thin mat equipped with pneumatic sensors that slips under the mattress. It tracks bed occupancy, heart rate, respiration rate, and snoring at the temporal resolution of a minute. The data also contains bathroom and bedroom occupancy, indoor light intensity, and temperature tracked using sensors placed in each household. Outdoor environmental parameters also affect sleep habits, and thus we include a dataset of outdoor temperature, humidity, and wind speed from the local area of the cohort.

1.3 Main objectives

The aim is to build a system that will provide sleep experts with the ability to explore the effect of potential interventions to improve sleep parameters. Following are the directions proposed by the challenge owners.

1. a. Can we predict the effect of different attributes on the sleep parameters? b. Can we estimate the marginal effect of a single feature, for example by fixing all other features except that one?

2. Can we build a synthetic data generator to augment the data to construct a better predictor?

3. Can we determine the optimum behavioral and environmental cues to help a patient reach their target sleep cycle?
1.4 Approach

This section describes the approaches we used for each challenge starting with an abstract formalisation followed by how we fit it to the given dataset and finally the learning tools used.

1.4.1 Predicting sleep parameters (Question 1a)

We formalise the setup in the language of causal inference. A random variable $Z$ is said to be an instrument (or the target) for another random variable $Y$ if $Z = f(Y, U)$, where $f$ is a deterministic function, and $U$ represents unobserved random variable, independent of $Y$.

The instruments (or target variables) are bed ($X^B_t$) and number of wake ups ($X^W_t$). We use the features (or parents, in causal terms, see Section 1.4.2) sensing data of the vitality measures and demographic information available in the dataset: heart rate, breathing rate, number rooms visited, lights, sex, age, and dementia diagnosis.

Let’s represent the dependence of the current state of a person, $X^i_t$ on the past using the following parent function $PA(X^i_t) = \{X^B_{past(t)}, X^W_{past(t)}, X^H_{past(t)}, X^B_{past(t)}, X^R_{past(t)}, X^L_{past(t)}, X^S_{past(t)}, X^A_{past(t)}, X^D_{past(t)}\}$, where $X^i_{past(t)}$ represents the values of feature $i$ in the time window $past(t) = (t - 1, \ldots, t - \tau)$ with window size $\tau \in \mathbb{Z}$.

Having organised the data this way, we can fit the multivariate time series as

$$X_t = F_\theta(PA(X_t), N_{X_t}) \quad (1)$$

where the random vector $F_\theta = (F^i_\theta)$ denotes a machine learning model for each feature $i$ such as linear model, non-parametric regression, neural network, etc, with $PA(X_t) = PA((X^i_t)_{i \in \mathcal{F}})$, for fixed $t$, the parents’ vector with $\mathcal{F}$ being the set of features, and $N_{X_t}$ a jointly independent noise term.

We have used the daily time series from dataset $D_{05}$ for this prediction task. All the analyses starts with preprocessing steps to handle the periodicity in the data and missing values. Statistical tests for stationarity and causality provided essential input to choose the learning model.
our experiments we have used VAR, VARMAX, multi layer perceptrons, and NGBoost methods to estimate the $F_\theta$.

1.4.2 Marginal effects of covariates to sleep measures (Question 1b)

Marginal effects of single features can be measured using the above model as a structural causal model (SCM) for a multivariate time series [ED10, PJS17].

$$X^i_t = F^i_\theta(\text{PA}(X^i_{t-1}), \ldots, \text{PA}(X^i_{t-\tau}), N_{X_t}).$$  

(2)

Note that Equation (2) is a discretisation of Equation (1) for each feature. Equation (2) associates with full time graph (the causal graph $\mathcal{G}$ for a time series model) [PJS17] as a directed acyclic graph (DAG) having $X^i_t$ for $(i,t) \in \{1, \ldots, d\} \times \mathbb{Z}$ as nodes, see Figure 1.

The importance of SCMs and the graphical architectures is the possibility to formalise the effect interventions and counterfactuals beyond observations in the data, see Figure 3. An intervention corresponds to modifying a subset of the structural assignments [ED10]. For example, setting $X_j^t$ only at one specific time instant $t$ to certain value or intervene on all values $\{X_j^t\}_{t \in \mathbb{Z}}$ for some feature $j$ to certain values. This last case is relevant, since in many applications, the sampling process may be slower than or even irrelevant to the time scale of the causal processes. In this context, the concept of summary graph is important. The summary graph is the directed graph with nodes $\{X^1, \ldots, X^d\}$ containing an arrow from $X^j$ to $X^k$ for $j \neq k$ whenever there is an arrow from $X^j_t$ to $X^k_s$ for some $t \leq s \in \mathbb{Z}$, see Fig.(2).

1.4.3 Synthetic data generation (Question 2)

This question can be divided in two separate points: either to augment data in order to build a robust predictor created as part of Question 1; Or to generate time series for heart rate and respiration rate as a proxy for “virtual patients” with a window of time-matched temporal data. For the purpose of the Data Study Group, we focus on the latter.

Possible techniques that can be found in the literature for this kind of problem include using Variational Autoencoders (VAEs) and Generative
Figure 1: Example of a full time graph. Source: [PJS17].

Figure 2: Summary graph of the full time graph above. Source: [PJS17].

Figure 3: SCMs do not only model an observational distribution $P$ but also intervention distributions and counterfactuals. Source: [PJS17].
Adversarial Networks (GANs).

We decided to start this challenge by implementing a “naïve” VAE to create a baseline model and, if time permitted, proceed to more complex architectures, such as Deep Koopman model [LKB18].

A VAE allows us to estimate the probability density function in a low dimensional latent space of the training data, which can then be used to generate new “realistic” samples. A VAE is made of two components: an encoder that maps the input samples to a bottleneck layer, where the number of neurons is the smallest; and a decoder that takes the encoded value back to the original input shape. The latent space is the space in which the data lies in the bottleneck layer.

The latent space contains the information the decoder will use to reconstruct the input as accurately as possible. To check that if the network is learning relevant features we will visualise the latent space with dimensionality reduction techniques such as t-SNE or t-distributed Stochastic Neighbour Embedding. These methods reduce a high dimensional point cloud to a low-dimensional one preserving the distance between the points. This dimensionality reduction step is for visualisation.

Note that here the model is not leveraging the temporal structure of the data. The inputs we tried to use in this experiment are: PatientID, day of the week, month, time in bed, time out of bed, heart rate, and respiration rate. However, for the moment we simplified our inputs and problem formulation, in order to set up a working framework that can be further expanded in future. More details about the implementation of our experiments can be found in Section 3.

1.4.4 Studying intervention (Question 3)

While we discuss the following approach, we did not manage to implement this part in the challenge due to time constraint.

We propose a path to answer this question using the idea of counterfactuals. Corresponding to an actual data sample, $X$ counterfactuals are synthetic samples within data distribution, that flips the model prediction from the prediction on $X$. With this, we can
understand how input changes affect the model and conduct reasoning under the what-if circumstances. To illustrate, consider a loan applicant who was rejected. Correlation-based explanations may simply indicate the most contributed features (e.g., income and credit) for the rejection, while counterfactuals are capable of showing how the application could be accepted with certain changes (e.g., increase the monthly income from $5,000 to $7,000) [YLDH21].

The research community has used counterfactuals to explain model predictions [YLDH21, JKKG18]. According to the theory proposed by J. Pearl [PM18], three distinct levels of cognitive ability are needed to fully master the behaviors of a particular model, i.e., seeing, doing and imagining from the easiest to the hardest cases. In fact, counterfactual explanation is just raised to meet the imagining-level cognition for model interpretation.

Despite the existing efforts, generating valid counterfactuals for raw data instances is still challenging due to the following reasons:

1. Effective counterfactuals for certain label are not guaranteed to be existed in training set, so the selected prototypes and criticisms are not always sufficient for counterfactual analysis [KKK16];

2. Efficient feature replacement for raw data instances could be very hard and time-consuming [GWE+19].

**Guaranteeing the existence of counterfactuals.** Since they are typically generated under the what-if circumstances which may not necessarily exist. To guarantee the counterfactuals existence, we can discuss them under the assumption under the assumption of "closest possible world", where desired outcomes can be obtained through the smallest changes to the world [WMR17]. To clarify better let’s follow the idea in [YLDH21].

Consider, without loss of generality, a binary classification model \( f_\theta : \mathbb{R}^d \rightarrow \{0, 1\} \), where 0 and 1 respectively indicate the undesired and desired output. The model input \( x \in \mathbb{R}^d \) is further assumed to be sampled from observational distribution \( P_X \). Then, given a query instance \( x_0 \) with the undesired model output (i.e., \( f_\theta(x_0) = 0 \)), the corresponding
counterfactual $x^*$ can be mathematically represented as:

$$x^* = \arg \min_{x \mid P_X > \epsilon} d(x, x_0) \quad s.t. \ f_\theta(x^*) = 1,$$

(3)

where $d$ is a distance measure in the input space, and $\epsilon > 0$ is the threshold which quantifies how likely the sample $x$ is under the distribution $P_X$. The obtained counterfactual $x^*$ is said to be valid if it can effectively flip the target classifier $f_\theta$ to the desired class.

Finding counterfactuals is similar to generating adversarial examples in the sense that they both aim to flip the model decision by minimally perturbing the input instance, however, they are essentially different in nature. Following the previous settings, the adversarial sample $x^{adv}$ for model $f_\theta$, with query $x_0$, can be indicated by:

$$x^{adv} = \arg \min_{x = x_0 + \delta} \|\delta\|_p \quad s.t. \ f_\theta(x^{adv}) \neq f_\theta(x_0),$$

(4)

where $\delta$ denotes the adversarial perturbation on the query and $\|\cdot\|$ represents the norm operation with $p \in \{\infty, 1, 2, \ldots\}$.

Note that counterfactual example has two significant differences from adversarial sample. First, counterfactual generation process is subject to the original data distribution, while adversarial samples are not constrained by the distribution. This difference brings about the fact that counterfactuals are all in-distribution samples, but adversarial examples are mostly out-of-distribution (OOD) samples. Second, counterfactual changes on the query need to be human-perceptible, while adversarial perturbations are usually inconspicuous. Therefore, the key problem of counterfactual explanation actually lies in how to generate such in-distribution sample, with human-perceptible changes on the query, to flip the model decision as desired.

**Generating counterfactuals** To handle the aforementioned challenges of counterfactual generation for raw instances, the high-dimension data space and non-semantic raw features are the two obstacles ahead. To deal with this, Ref.[YLDH21] design a framework to generate counterfactuals specifically for raw data instances with their proposed Attribute-Informed Perturbation (AIP) method. By using generative
models, they obtained useful hypothetical instances within the data distribution for counterfactual analysis. Essentially, the AIP method guide a well-trained generative model to generate valid counterfactuals by updating its parameters in a attribute informed latent space, which is a joint embedding space for both raw features and data attributes. Compared with the original input space, attribute-informed latent space has two significant merits for counterfactual generation: (1) raw features are embedded as low dimension ones which are more robust and efficient for generation; (2) data attributes are modelled as joint latent features which are more semantic for conditional generation.

Generative modelling aims to summarize the data distribution of input variables and further create new samples that plausibly fit into that distribution. Emerging families of generative modelling, such as Generative Adversarial Networks (GANs) [10] and Variational Auto-Encoders (VAEs) have been attracting lots of attentions due to their full advantage of their power on raw data with high dimensionality and has been used in Ref.[YLDH21]. Here, we could update their techniques by using the idea of Robust Variational Auto-Encoders (RVAEs). By exploiting the concept of Koopman Operators, this method concentrate in discover the dynamical law behind the unknown data-generating system, see Sec.4.3.2. This path could give new interesting results from both theoretical and practical point of views.

1.5 Main conclusions

A multivariate time series forecasting model is implemented to forecast the sleep duration and the number of wakeups for each patient from the Daily Summary dataset using time series approaches such as Vector Autoregressive (VAR), Vector Autoregressive Moving Average (VARMA), and Neural Networks (LSTM). The results show that VAR and VARMA models performs better than neural networks, possibly due to small training dataset. We evaluate the performance using mean absolute error, mean percentage error, root mean square error, and mean absolute percentage error. The results can be improved by using the approaches mentioned in future work.

The trained VAE is able to generate visually similar sleep patterns to the
data. The latent space representation of the sleep patterns form patients form a giant cluster with the patients except 10 and 11. The cluster suggests that there is some structure in the sleep patterns. However, the structure cannot be explained by any of the available covariates.

1.6 Limitations

- The prediction models experimented in this challenge only use the dataset D_05 and D_01. We believe that using other datasets could improve the accuracy, but is not tested as part of the challenge due to time constraints.
- The time series prediction experiments consider each patient independently. Learning from data accumulated across similar patients can potentially yield more accurate predictions.
- We have not investigated effects of unmeasured confounding variables [Sjö19] (condition known by causal sufficiency) to estimate the ACEs, in Equation (6).
- The dataset contains a small number of patients (38). As the patients are heterogeneous in terms of their sleeping habits, we believe, data from more patients would likely improve the accuracy of the results (see Section 2.2 for details).

1.7 Future work

For a full description of possible future work, see Section 4. Three main broad directions are i) exploring the data and application, ii) exploring different model evaluation strategies, and iii) exploring different modelling strategies.

2 Data overview

2.1 Dataset description

The dataset contains longitudinal multi-dimensional data from 38 dementia patients. Most of the patients in the dataset are diagnosed with
Alzheimer’s disease ($N = 25$), while there are 5 patients diagnosed with dementia in Parkinson’s disease, 4 Vascular dementia patients, 3 patients diagnosed with fronto-temporal dementia and one patient with a mixed diagnosis (Figure 5, left side). The mean patients’ age is 82.61 (range: 71 – 94, Figure 5, right side). Cognitive functioning was assessed with the Neuropsychiatric Inventory–Questionnaire (hereafter, NPI, Figure 6, left side) and the Mini-Mental State Examination (hereafter, MMSE, Figure 6, right side).

The data was obtained using multiple passive sensors that recorded patients’ activity in their home. A Withing’s Sleep Analyzer, which is a thin mat equipped with sensors that slips under the mattress, tracked bed occupancy (time spent in bed, time out of bed), respiration and snoring measures on an hourly basis (i.e., 24 datapoints per day). In addition, the dataset contains information about temperature and light conditions in the patient’s home. Finally, the dataset contains information about external conditions (temperature, wind speed, and humidity).

The data is divided into five datasets that are linked by patient identifiers (pseudo-randomized). The datasets have 18,192 recordings in total, with the majority of patients contributing around 700 recordings (Figure 4).

The dataset 2 contains hourly mean heart and respiration rates, snoring activity sums (of minutes) and bed occupancy sums (minutes). Dataset 5 contains per patient daily summary of the same parameters, specifically the following. Number of times a person woke up (nb_awakenings); time in the bed in minutes (in-bed); time out of bed (out-of-bed); maximum, minimum, and average heart rate (hr-max, hr-min, hr-mean); maximum, minimum, and average respiration rate (rr-max, rr-min, rr-mean).

Attributes in the dataset 5 are shown from different perspectives in Figure 10, 11, 12, and 13. We only included the detailed visualization for dataset 5, as our experiments are restricted to this dataset due to time constraint of the Data Study Group.

Heart rate, respiration rate, snoring, light intensity, and indoor temperature are included as hourly averages. The time a participant spent in bed is measured between 6pm to next day 12pm.

We also had access to a dataset containing daily summaries, with the following features time when a patient went to bed, time when a patient
Figure 4: Distribution of the sleep features: the number of nights patients contributed to the dataset, number of awakenings, time spent in bed and time spend out of bed.

got out of bed, minutes spent in bed, minutes spent out of bed, number of awakenings, minimum heart rate, maximum heart rate, mean heart rate, minimum respiration rate, maximum respiration rate, mean respiration rate.
Figure 5: Number of patients in each diagnosis group and age distribution in the patient sample.

Figure 6: Summary of patients’ scores on the Neuropsychiatric Inventory (NPI) and the Mini-Mental State Examination (MMSE).

Figure 7: Visualisation of patients’ heart rate data. The x-axis depicts time in hours during one day (starting with noon), y-axis depicts time in days with each line representing one recording (i.e. one day). Note that the heart rate is recorded by the mattress sensors and therefore only recorded when a patient is in bed (time points with no data, i.e. when a patient was not in bed, are depicted in grey).
Figure 8: Visualisation of patients’ respiration data. The x-axis depicts time in hours during one day (starting with noon), y-axis depicts time in days with each line representing one recording (i.e. one day). Note that the respiration rate is recorded by the mattress sensors and therefore only recorded when a patient is in bed (time points with no data, i.e. when a patient was not in bed, are depicted in grey).

Figure 9: Visualisation of three patients’ snoring data. The x-axis depicts time in hours during one day (starting with noon), y-axis depicts time in days with each line representing one recording (i.e. one day). Note that the snoring rate is recorded by the mattress sensors and therefore only recorded when a patient is in bed. This shows heterogeneity of the patients in terms of their snoring habit.
Figure 10: Time-series plot over all patients for dataset 5.
Figure 11: Time-series plot of dataset 5 colored by diagnosis. The sequence of colors represent the type of dementia: Alzheimer’s - red; Parkinson’s(P) - orange; Fronto-temporal(FT) - green; Vascular(V) - blue.
Figure 12: Weekly averages for dataset 5.
Figure 13: Monthly averages for dataset 5.
2.2 Data quality issues

We describe some of the issues of the dataset provided. The small number of patients limits the success of machine learning models. Moreover, analysis is hard due to the fact that patients differ in terms of contributed numbers of recordings (e.g. patients that have withdrawn from the study during the period of data collection).

Another issue identified in the dataset is missing data. One source of it are days with no records, another source is more structural, meaning that when a patient is in bed, the indoors movement sensors are not receiving signals. Both these sources of missing data pose a problem in modelling the time-series. These issues are treated in different ways in the approaches that we propose.

3 Experiments

This section starts with predicting sleep parameters and then proceeds to estimating the marginal effects of the predictors, and finally evaluates synthetic data generators.

3.1 Predicting sleep parameters

In this part, we predict different sleep parameters: time in bed, number of awakenings, and activity. Different machine learning methods are evaluated ranging from classical time series forecasting methods to NGBoost and neural networks. Methods are also varied in terms of considering the temporal dependency in the data. All experiments here use dataset D_05 and D_01.

3.1.1 Predicting time-in-bed using NGBoost

Uncertainty in prediction often is crucial to the systems that take decisions based on predicted values. For example in the current context, uncertainty is important along with the predicted effect of interventions on the patient’s sleep quality. The uncertainty shall provide the clinical practitioners a measure of reliability on the prediction when suggesting
the interventions to the patients. Hence, here we predict a probability density function (PDF) of time-in-bed for every input sample.

We decompose the sources of predictive uncertainties into two distinct categories: aleatoric and epistemic. Aleatoric uncertainty captures the uncertainty inherent to the data generating process. To analogue using an everyday object, this is the entropy associated with an independent toss of a fair coin. Epistemic uncertainty, on the other hand captures the uncertainty associated with improper model-fitting. In contrast to its aleatoric counterpart, given a sufficiently large dataset epistemic uncertainty can theoretically be reduced to zero [Gal16, GDF+21].

Aleatoric uncertainty is thus sometimes referred to as irreducible uncertainty, while epistemic as the reducible uncertainty. High aleatoric uncertainty can be indicative of noisy measurements or missing informative features, while high epistemic uncertainty for a prediction could be a pointer to the outlier status of the associated input vector. Here, we predict just the aleatoric uncertainty, and leave the derivation of the epistemic uncertainty for future work.

**Metrics for Deterministic Predictions.** For each input sample $X$ we derive the predicted mean time-in-bed and the associated aleatoric uncertainty as $\mu$ and $\sigma$, respectively. We present three measures to quantify the quality of the predicted time-in-bed values: root-mean-square error (RMSE), mean absolute error (MAE), and bias error (BE). Respectively, these three measures are defined as

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\mu_i - y_i)^2},$$

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |\mu_i - y_i|,$$

$$\text{BE} = \frac{1}{N} \sum_{i=1}^{N} (\mu_i - y_i).$$

In the above definitions, $y_i$ and $\mu_i$ are the true and predicted time-in-bed values respectively corresponding to an input sample $i$ and $N$ is the number of samples.
Methodology.

1. For modelling, we use NGBoost [DAD+19] with hyperparameters chosen via trial-and-error to minimize loss on the validation set, and use MSE with Gaussian likelihood as the loss function. This enables us to predict both the mean and standard deviation for each sample.

2. Datasets. We use two datasets, D_05 and D_01. First, we only use the daily summary dataset, D_05 for training and testing in Figure 14. Next, we combine it with D_01 in Figure 15. We note that results using both datasets are more accurate, as can be deduced from the RMSE, MAE, and BE values in both these figures. For both experiments, we used the same set of hyperparameters in NGBoost for a fair comparison.

3. Feature engineering for time: NGBoost cannot, at least yet, handle time series data. To adequately leverage the temporal information in D_01 and D_05, we convert the column in each with time-stamps to three different features: hour-of-day, day-of-week, and week-of-year. All three range from 0 to 1. To account for the cyclic nature of these quantities, we then further convert them to a sin and a cos term each, thus converting the original single column with time-stamps to six separate columns.

4. One-hot encoding of patient IDs: We have 37 patients, each with
a unique patient ID. We one-hot encode these, thus converting the patient-ID column in both datasets $D_01$ and $D_05$ to 37 columns.

5. Split into training, validation, and test sets based on date: For each patient, we first order the samples chronologically. Next we pick the first 70% of the samples for training, the next 15% for validation, and the last 15% for testing. Finally, we collate all data points for each patient into three bins for training, validation, and testing.

Figure 15: Same as Figure 14, but utilising both datasets $D_05$ and $D_01$. By comparing the metrics here with those in Figure 14, we can see that the additional information in dataset $D_01$ is of predictive value.

### 3.1.2 Predicting number of wakeups and time-in-bed using time series forecasting

Machine learning methods on time series learn structures from past observations to forecast the future values. Depending on the nature of the data, such modelling can be univariate or multivariate. In a multivariate time series there are more than one time-dependent variable. Each variable can also depend on other variables and not just its own previous values.

The given dataset gives patients' sleep patterns and wakeups as daily time series and thus is amenable to time series forecasting techniques. As heart rate and respiration rate are causally related to sleep duration and wakeups, predicting sleep duration and the number of wakeups only based on their historical data is insufficient. Therefore, we use a multivariate approach to build a predictor.
We use features from a week (or lags) to predict the target on the next day. We use heart rate and respiration as predictors to predict the number of awakenings events per night and total time spent in bed, respectively. Daily summary dataset ($D_{05}$) is used for this experiment. The experiment first considers each patient's data independently and then generalized to all patients. We used Vector Autoregressive (VAR), Vector Autoregressive Moving Average (VARMA), and Long Short-Term Memory (LSTM) models to conduct comparative experiments.

The detail of this experiment for using the time-series method includes data processing, statistical test, model description, and analysis of the results.

**Data Preprocessing.** The dataset $D_{05}$ contains PID (Patient ID), night (date), bedtime start and end, number of awakenings events per night, total time spent in and out of bed during the bedtime opportunity period (end-start), and the heart and respiration max, min and means values.

The same preprocessing steps are followed for all the patients. Data preprocessing steps include missing value treatment, feature selections, and converting the night(date) column into date-time format. Missing dates (the dates on which there is no recordings) were first added. Then in-bed, out-of-bed, heart rate and respiration rates are added using linear interpolation. The PID column is dropped since this technique forecasts for an individual patient. The start and end time are also dropped. The night(date) column is then set as the index of the data frame to convert data into a time-series format.

Prior to building the models, we perform two statistical tests: stationarity test and causality test to select the right model.

**Stationarity test: Augmented Dickey-Fuller Test (ADF).** The statistical properties of a process generating a stationary time series do not change over time. Thus, such a time series is amenable to accurate long term prediction, for example a sine wave. As expected stationarity has a crucial influence on the choice of model.

Two popular methods for the stationarity test of time series are i) a qualitative assessment of the time series using autocorrelation plots, and ii) a method based on statistical test. Here we use Augmented
Dickey-Fuller test from the second category.

The Augmented Dickey-Fuller test is also known as a unit root test and quantifies how strongly a time series is defined by a trend. The null hypothesis of the test is that the time series is not stationary and the alternate hypothesis is that the time series is stationary. We consider threshold $p$-value to be 0.05.

In this experiment, we carried out ADF tests on all variables. The result is that all variable except the in-bed is stationary. Therefore, we take the first-order differentiation of in-bed to make it stationary. The time-series data after the conversion are shown in Figure 16.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>$p$ - value</th>
<th>Stationarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>nb-awakenings</td>
<td>0.012</td>
<td>Series is Stationary</td>
</tr>
<tr>
<td>in-bed</td>
<td>0.029</td>
<td>Series is Stationary</td>
</tr>
<tr>
<td>out-of-bed</td>
<td>0.0</td>
<td>Series is Stationary</td>
</tr>
<tr>
<td>hr-max</td>
<td>1.08e-17</td>
<td>Series is Stationary</td>
</tr>
<tr>
<td>hr-min</td>
<td>0.002</td>
<td>Series is Stationary</td>
</tr>
<tr>
<td>hr-mean</td>
<td>5.47e-05</td>
<td>Series is Stationary</td>
</tr>
<tr>
<td>rr-max</td>
<td>5.852e-12</td>
<td>Series is Stationary</td>
</tr>
<tr>
<td>rr-min</td>
<td>8.567e-10</td>
<td>Series is Stationary</td>
</tr>
<tr>
<td>rr-mean</td>
<td>2.495e-19</td>
<td>Series is Stationary</td>
</tr>
</tbody>
</table>

Table 1: p-values from ADF Test after making the time series stationary.
Figure 16: Time series of a randomly selected patient for the summary dataset (D05) after it has been made stationary.

Causality test: Granger Causality Test.
Granger causality is a method for determining whether or not variables in
A time series are causally related. A probabilistic explanation of causation is used in this technique. It is a popular statistical test used in economics. It uses one variable to predict another in a multivariate time series with a given lag. Granger causality is defined as whether previous values of $X_{t-1}$ help in the prediction of $Y_t$, assuming that the effects of past values of $Y_{t-1}$ on $y_t$ have already been accounted for (and perhaps of past values of other variables). If they do, $X$ is said to be the “cause” of $Y$. VAR is based on the idea that each time series in the system impacts the others. The data must be stationary in order to run the Granger Causality test, which means it must have a constant mean, constant variance, and no seasonal component. Thus our previous step to ensure that the time series are stationary enables the Granger Causality Test.

Here the null Hypothesis ($H_0$) is that $X_t$ does not granger causes $Y_t$, and the alternate Hypothesis($H_1$) is that $X_t$ granger causes $Y_t$. We use the threshold for $p$-value as 0.05.

In this experiment, Granger Causality Test is carried out for the number of awakenings events per night. The table of the Causality Test is shown as follows.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>$\max(p\text{-value})$</th>
<th>$\text{Null Hypothesis}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-bed</td>
<td>0.0002</td>
<td>Reject</td>
</tr>
<tr>
<td>out-of-bed</td>
<td>0.0412</td>
<td>Reject</td>
</tr>
<tr>
<td>hr_max</td>
<td>0.9733</td>
<td>Fail to reject</td>
</tr>
<tr>
<td>hr_min</td>
<td>0.9257</td>
<td>Fail to reject</td>
</tr>
<tr>
<td>hr_mean</td>
<td>0.1013</td>
<td>Fail to Reject</td>
</tr>
<tr>
<td>rr_max</td>
<td>0.1726</td>
<td>Fail to reject</td>
</tr>
<tr>
<td>rr_min</td>
<td>0.0001</td>
<td>Reject</td>
</tr>
<tr>
<td>rr_mean</td>
<td>0.8607</td>
<td>Fail to reject</td>
</tr>
</tbody>
</table>

Table 2: Maximum of p-values from Granger Causality test for attribute nb-awakenings.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>(max(p-value))</th>
<th>(NullHypothesis)</th>
</tr>
</thead>
<tbody>
<tr>
<td>out-of-bed</td>
<td>0.747</td>
<td>Fail to Reject</td>
</tr>
<tr>
<td>hr_max</td>
<td>0.9056</td>
<td>Fail to reject</td>
</tr>
<tr>
<td>hr_min</td>
<td>0.0382</td>
<td>Reject</td>
</tr>
<tr>
<td>hr_mean</td>
<td>0.0178</td>
<td>Reject</td>
</tr>
<tr>
<td>rr_max</td>
<td>0.6775</td>
<td>Fail to reject</td>
</tr>
<tr>
<td>rr_min</td>
<td>0.8763</td>
<td>Fail to Reject</td>
</tr>
<tr>
<td>rr_mean</td>
<td>0.9687</td>
<td>Fail to reject</td>
</tr>
</tbody>
</table>

Table 3: Maximum of p-values from Granger Causality test for attribute in-bed.

**Prediction models.**
A multivariate time series has more than one time-dependent variable. Each variable is not just dependent on its previous value, but also on other variables. Multivariate time series models are intended to capture the dynamics of many time series at the same time while using dependencies between them to make more accurate predictions.

The D_05 dataset includes nb-awakenings, in-bed, out-of-bed, hr_min, hr-max, hr-mean, rr-min, rr-max, rr-mean for approximately the past two years. In this case, there are multiple variables can be used to predict nb-awakenings and in-bed. Vector auto-regressive moving average models (VARMA) is a vector variant of popular Autoregressive integrated moving average (ARIMA) that can be used to analyse the connections among many variables in multivariate time series analysis. We use VAR, VARMA, and LSTM to forecast sleep duration and number of wakeups.

The choice of history window length (or the lag) to consider is an important measure that influences accuracy. Following presents the methods we use to select a lag that produces highest accuracy.

**Probabilistic model selection: AIC and BIC.**
Probabilistic model selection is an analytical approach for evaluating and
selecting candidate models. There are several statistical methods for assessing how well a given model fits a dataset and how complicated the model is.

- **Akaike Information Criterion (AIC).** It is defined as
  \[
  AIC = -\frac{2}{N} \times LL + 2 \times \frac{k}{N}
  \]
  Where \( N \) is the number of examples in the training dataset, \( LL \) denotes the log-likelihood of the model on the training dataset, and \( k \) is the number of parameters in the model.

  To utilise AIC for model selection, one needs to choose the model with the lowest AIC from the collection of models evaluated. The penalty for AIC is lower than the penalty for BIC. As a result, AIC selects more complicated models.

- **BIC Derived from Bayesian probability.** It is defined as
  \[
  BIC = -2 \times LL + \log(N) \times k
  \]
  Similar to AIC, the model with the lowest BIC needs to be chosen. The BIC penalises the model for its complexity more.

In our experiment, the number of lags for the VAR and VARMA model are decided based on the AIC Score although one can also use the BIC score. The AIC score was minimum at 11th lag (or 11 days) for the number of wakeups forecasting and the AIC for the sleep duration forecasting had a minimum score at 8th lag (or 8 days). So while modelling VAR and VARMA, these lags were used to forecast the future values.

Next we briefly describe the machine learning models used for prediction.

**VAR.** Vector Autoregression (VAR) is a forecasting algorithm that can be utilised when covariates in a multivariate time series have bidirectional dependence. Each variable in VAR is a linear function of all variable values from the past. Because there are several time series influencing each other, it is treated as a system of equations with one equation for each variable. VAR can comprehend and apply correlations between various variables, which aids in describing the dynamic behaviour of the
data and delivers improved predicted results. It is described as an Autoregressive model since each variable (Time Series) is modelled as a function of previous values. Its major distinction with ARIMA is that while ARIMA is unidirectional, i.e., the predictors affect the Y but not vice versa, VAR is bi-directional, i.e., the factors influence one another.

VAR(p) Model can be written by:

\[ Y_t = c + A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_p y_{t-p} + e_t \]

where, \( c \) is a \( n \times 1 \) constant vector, \( A_i \) is a \( n \times n \) matrix, and \( e_t \) is a \( n \times 1 \) noise vector.

**VARMA.** The Vector Auto Regression Moving-Average (VARMA) technique combines VAR and VMA and extends ARMA to multivariate time series. The VAR structure and the moving average terms for each variable are included in VARMA models. Similar to VAR model, it supports bi-directional dependence between features and the target.

The order of the VARMA model needs to be determined before it is established. One is \( p \) in the vector autoregressive model and \( q \) in the vector moving average model.

VAR(p) Model can be written by:

\[ y_t = c + A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_p y_{t-p} + M_1 \epsilon_{t-1} + \cdots + M_q \epsilon_{t-q} \]

where,

\( y \) is a stack of multiple time series vector (v)
\( y_{t-1}, \cdots \) are the AutoRegressive (AR) terms
\( M_1 \epsilon_{t-1}, \cdots \) are the Moving Average (MA) terms

**Experiment setting.** In both the above models, VAR and VARMA, the last 2 weeks of data is used for testing and the rest for training.

The Granger causality test helped us to employ three endogenous factors to determine sleep duration in our experiment: in-bed, hr-mean, and hr-min. Since the AIC score was lowest at the 8-th lag, the number of lags is taken is same as VAR as 8, i.e. \( p = 8 \).
Due to the Granger causality test, we utilised four endogenous factors to calculate the number of wakeups: nb-awakenings, in-bed, out-of-bed, and rr-min. The number of lags is considered is same as VAR as 11, i.e. \( p = 11 \), because the AIC score was lowest at the 11th lag.

**Neural Networks (LSTM).** LSTM is a specific type of Recurrent Neural Network (RNN) that uses gates to preserve information over long history [GSK+16]. Figure 17 shows the internal of a LSTM cell.

![LSTM structure](ASY20)

We access the data using a sliding window of length 7 days. So, the data of 7 days is used to predict the 8-th day and this window slides one day at a time. All the features in a time series are normalised to have mean 0 and unit variance.

A Time series generator is used which is provided by Keras library to time series data into samples, prerequisite for training deep learning models.

For forecasting of in-bed (sleep duration) the following arguments are passed to Keras TimeseriesGenerator:

1. features: in-bed, hr-min, hr-mean
2. target: in-bed
3. length (It is the window length): 7

For forecasting of nb-awakenings (number of wakeups) the following arguments are passed to Keras TimeseriesGenerator:

1. features: nb-awakenings, in-bed, out-of-bed, rr-min
2. target: nb-awakenings
3. length (It is the window length): 7

Results and Discussions

Summary dataset, D_05 is used for this experiment. We used the data for the last 12 days for testing and the rest for training.

In the forecasting of awakenings events per night, we used {'nb-awakenings','in_bed','out_of_bed','hr_mean','rr_max','rr_min'} as predictors; In the forecasting of total time spent in bed, we use{'in_bed','hr_min','hr_mean'} as predictors.

Four functions were used to evaluate the forecasting results: Mean Error(ME), Mean Absolute Error(MAE), Mean Percentage Error(MPE), Root Mean Square Error(RMSE) and Mean Absolute Percentage Error(MAPE).

\[
ME(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} y_i - \hat{y}_i
\]

\[
MAE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} |y_i - \hat{y}_i|
\]

\[
MPE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} \frac{y_i - \hat{y}_i}{y_i}
\]

\[
RMSE(y, \hat{y}) = \left[ \sum_{i=1}^{n_{samples}} (y_i - \hat{y}_i)^2 / n_{samples} \right]^{1/2}
\]

\[
MAPE(y, \hat{y}) = \frac{100}{n_{samples}} \sum_{i=1}^{n_{samples}} \left| \frac{y_i - \hat{y}_i}{y_i} \right|
\]

where, \( \hat{y} \) and \( y \) are predicted and true values corresponding to an input sample \( i \) and \( n_{samples} \) is the number of samples.
According to the prediction results in Tables 4 and 5, VAR (Vector Autoregressive) performs better when estimating sleep duration, whereas VARMA (Vector Autoregressive Moving Average) performs better when forecasting a patient’s number of wakeups. These findings were obtained by feature selection utilising the Granger Causality test and the AIC - BIC test to determine the delays or window size needed to train the model.

It can be observed that LSTM models didn’t perform well when compared to the VAR and VARMA, as the data per patient was less and neural networks require sufficient data to make good predictions. The implemented time series models such as the VAR, VARMA and LSTM could be improved by using indoor and outdoor features such as light,
temperature and weather along with the demographics for each patient along with their hourly data.

### 3.1.3 Predicting activity

As we have patients’ hourly bed occupancy data (called activity in D\_02 which ranges from 0 to 60 minutes), it is natural to seek patterns in the future activity based on the historical hourly data. On the one hand, we have more data points to train on compare to approaches that rely on summary statistics D\_05; one the other hand, it is a more difficult task to predict higher dimensional time series.

For simplicity, we consider a vector of consecutive $72 = 24 \times 3$ hours values of activity, $X \in \mathbb{R}^{72}$, as predictor, and we aim to predict the next 24 hours values $Y \in \mathbb{R}^{24}$.

**Metric.** Our goal here is to predict $Y$ as accurate as possible, in the sense that the average prediction error is minimised.

$$\text{avg\_pred\_error} = \frac{1}{24} \sum_{i=1}^{24} |Y_i - Y^{\text{pred}}_i| = \frac{1}{24} \|Y - Y^{\text{pred}}\|_1$$

**Data.** We focus on one single patient (patient 11) who has clearer patterns of wake up due to religion, and so it is interesting to see if one can predict the wake up by our models.

We will train our model on training set based on 80% of the available dates. Suppose we define one night by 6 pm to 6 pm the next day, we identify a missing date if activity takes value 0 for all the hours in the 24 hours period. Then we can assign the values to be $\text{NA}$. When we slide the window to select the training set or testing set, we will skip if the window that contains at least one $\text{NA}$ value.

**Model.** In theory, deep neural network achieves ‘almost’\(^2\) optimal minimax rate for non-parametric regression with ReLu activation function. Therefore, we consider

$$Y = F_{\theta}(X) + \epsilon$$

\(^2\text{off by some logarithmic factor } \text{https://arxiv.org/abs/1708.06633} \)
where $F_{\theta}$ belongs to some feed-forward neural network functional space.

We consider the following

- **Naive neural network:** 3 hidden layer feed-forward neural network. This model may produce output out of the range $[0, 60]$, so we simply apply a projection function

  \[ P : \mathbb{R} \mapsto [0, 60] \]

  \[ P(x) := \arg\min_{a \in [0,60]} \{ |x - a| \} \]

  when we make predictions.

- **Improved neural network:** same as Naive neural network except that, instead of applying projection function, we add a non-linear transformation (based on the Sigmoid activation function) to enforce the output range.

To train the neural network, we also need an objective function. We considered

- $\ell_2$ norm: proportionate to Root Mean square error (RMSE), we fit the model in a least square sense.

- $\ell_1$ norm: encourage sparsity (more zero elements) in the error vector $\epsilon$ as $\ell_1$ is the convex relaxation of $\ell_0$ (counts of non-zero elements).

Based on the metric we defined, $\ell_1$ should be a better choice. However, due to the non-convex nature of neural networks, we will reach a sub-optimal fit for the model, therefore we include $\ell_2$ loss function for comparison. Also, we may want to consider elastic-net-typed loss function $\alpha \ell_1 + (1 - \alpha) \ell_2$ with $\alpha \in [0, 1]$ that is more flexible than $\ell_1$ or $\ell_2$ alone.

**Results:** See Figure 18, 19, 20. Each of the figure consists of 3 plots: plot of $Y_{\text{pred}}$, $Y_{\text{truth}}$, $Y_{\text{pred}} - Y_{\text{truth}}$ from left to right. The range of $Y_{\text{pred}}$, $Y_{\text{truth}}$ are $[0, 60]$, so the range of $Y_{\text{pred}} - Y_{\text{truth}}$ is $[-60, 60]$.

We also summarise the average prediction error in Table 6. Here, the baseline is using the current 24 hours values to predict the next 24 hours. The smaller avg_pred_error value the better.

<table>
<thead>
<tr>
<th>avg_pred_error (hr/day)</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
</tr>
</tbody>
</table>
From the experiments, we note that the naive neural network model predicts better when it was trained with $\ell_1$ loss. It makes sense because the vector contains many 0’s and 1’s. The use of $\ell_2$ loss could make the residuals small but not shrink them to zero (sparse). However, it does not pose a problem for the improved neural network where the range is enforced by the output layer.

It is interesting to see that training with $\ell_1$ loss, the model can learn the shifted wake up time presented in the data. It does not have a lower prediction error because in the testing set, the wake up time is not shifted as before.

**Summary**: We inspected in a simple neural network framework for time series prediction which can be easily extended by including more features. It can predict roughly 22 hours of sleep pattern each day for a single patient. With more features and data, the model may be able to predict with more accuracy or even detect the change points. As a result, we can simulate the sleeping pattern data based on the neural network (or adapted it to VAE) making personal recommendation.

### 3.2 Estimating marginal effects

Here we estimate the marginal effects of the summary graph, see Sec. (1.4.1) for the heart and respiration rate as predictors set $PA_{SL} = \{H, R\}$ on total time spent in bed per hour/night as the target.
variable, $SL$. Hour and Daily summary data (datasets 2 and 5) was used for this experiment.

The causal effect of $X$ on $Y$, denoted by $\text{Pr}(Y|\text{do}(X = x))$, it denotes the counterfactually generated distribution of $Y$ where $X$ is set to a given value $x_0$. This is not, in general, the same as the ordinary conditional distribution $\text{Pr}(Y|X = x)$ [Sha16]. Calculating counterfactual is challenging due to possible inter-dependence between variables in $\text{PA}_Y$ which may result in confounders (common causes) between $X \in \text{PA}_Y$ and $Y$. A variable $Z$ is said to be a confounder of $X$ and $Y$ if $X \leftarrow Z \rightarrow Y$. Strategies exist to identify confounders, e.g. front-door adjustment. These methods broadly work in two phases.
Given data about a system, find its causal structure;

2. Given the causal structure of a system, estimate the effects the variables have on each other.

3.2.1 The causal structure

To answer the first problem the graphical structure is needed (summary graph or simply DAG here). For this, Granger Causality Test Sec.(4.2.1) was utilized to identify the direction of the arrow between the heart (H) and respiration (R) variables.

We observed by the test that for daily, weekly and monthly lags (dataset 5) the arrow $H \rightarrow R$; but, for 1h, 6h of lag (using dataset 2) the arrow had the opposite direction. We concentrated on the dataset 5 analysis which gave to us the DAG of Fig.21. Fixed the graphical structure, problem 1 became easier to analyse.

3.2.2 The back-door criterion

When estimating the effect of $R$ on $SL$, a back-door path is an undirected path between $R$ and $SL$ with an arrow into $R$, the variable $H$ here. These are the paths which create confounding, by providing an indirect, non-causal channel along which information can flow. It turns out that $H$ satisfies the back-door criterion [Sha16], in which case, we say that $H$
blocks the non-causal channel and represent this by an box instead of a circle in the DAG, Fig.22. And

\[ \Pr(SL|\text{do}(R = r)) = \sum_h \Pr(SL|R = r, H = h)\Pr(H = h) \]  

Note that all terms on the right-hand side are observational and not interventional. In general, for analysis usually an averaging over the target variable is performed to get averaged causal effect (ACE) [Pea09]:

\[ \mathbb{E}(SL|\text{do}(R = r)) := \sum_s s\Pr(SL = s|\text{do}(R = r)). \]  

**Analysis.** To estimate the ACE we adapt the functions from causality package. In order to clarify the importance of intervening to isolate marginal effects in contrast to only observing, conditional expectations (CEs), we construct the table below:
Table 7: The first column represents the dataset 5 averaged over 7 and 30 days. The second and third, are the ACEs (interventions) and CEs (observations) averaged over all realizations of $R$ after a process of binarization.

By table above, we can see qualitatively that, the $\mathbb{E}(SL|do(R))$ values are smaller than $\mathbb{E}(SL|R)$. This means that the interventional procedure eliminates spurious correlations when compared with conditional operation only. For a quantitative analysis we would need to guarantee the i.i.d. regime of the samples. We did not manage this part due to time constraint.

3.3 Generating synthetic data in Question 2

In this part of the project we focus on generative modelling of the available sleep data.

3.3.1 Variational autoencoder to model sleep patterns

Our initial approach to generative modelling was to focus on the modelling of sleep patterns across patients. To this end, we used a variational autoencoder (VAE) architecture to learn a generative model of the time patients spend in and out of bed.

VAEs are a popular neural network architecture for generative modelling. In the simplest case, VAEs assume that the observed data $x$ can be captured by a probabilistic model $p(x|z)$ that is fully determined by a latent space representation $z$. Together with a generative model for the
latent representation $p(z)$ define a generative model for the data
\[ p(x, z) = p(x|z) p(z). \]  
(7)

The challenge of generative modelling is to identify the conditional distribution $p(x|z)$ which generates realistic new ‘data’ for the latent distribution $p(z)$. VAEs achieve this in two steps: Firstly, an autoencoder-like architecture is used to learn the latent space representation of the data. In particular, an encoder, in form of a neural network, learns the mapping from the data $x$ to the latent space $z$, and a decoder, in form of another neural network, learns to reconstruct the data from the latent space encoding. The latent space representation is typically lower dimensional than the input data. Thus, in order for the decoder to successfully reconstruct the data from the latent space representation, $z$ has to focus on the relevant features of the data. By comparing the input data $x$ to the reconstructed data $\hat{x}$ we can obtain a standard autoencoding architecture, which can be trained by optimising the reconstruction quality. This autoencoding architecture by itself does, however, not learn a generative model because it fails to connect the latent space encoding to the generative distribution $p(z)$. Thus, even if an autoencoder can reduce the input data $x$ to essential the features $z$, we have no means to generate realistic new features to synthesis realistic new data. A variational autoencoder, therefore, does not learn the decoding of the data $x$ in the latent representation $z$ directly, but rather learns a distribution of $z$, $q(z|x)$, for any given realisation of the data $x$. By penalising the distance of the encoding distribution $q(z|x)$ to the generative distribution $p(z)$, we can train the encoder to map the data to a latent space representation that corresponds to the generative process.

As a result, a VAE can be trained with two loss contributions: a reconstruction loss, which compares the input to the reconstructed data, and a generative loss, which compares the encoding distribution to the generative distribution. Note that these contributions to the loss cannot be optimised simultaneously. The generative loss favours encodings that are equal to the generative distribution $p(z)$. However, if all realisations of the data are encoded by the same distribution, the decoder can no longer successfully identify different realisations of the data and always reconstructs the same ‘mean’ data which leads to a suboptimal
reconstruction loss. On the other hand, if the encoder produces different distributions for all realisations of the data, the reconstruction of the data may be easier, but the distance of the encoding distributions to $p(z)$ will be suboptimal. As a result, variational autoencoders learn a trade off encoding that produces distributions which only differ from $p(z)$ to encode unique features that are necessary for a successful reconstruction. This, in turn, has the benefit that data with similar features will be encoded by similar distributions in latent space.

**Data preprocessing**

We focused on modelling the sleep patterns across patients. To this end, we used the hourly measurements of the bed mats to derive a time series that indicates when patients spent time in and out of bed. In particular, we binarised the heart rate data by transcribing NaNs to zeros and any finite heart rate measurements to ones. We then organised the data into daily records of the time in and out of bed for each patient using the patient ID and the date time. As the typical sleep timeframe of a patient lies between 6pm and 6am, we shifted the date time by 12 hours and then used the date to identify the 24 hour sequences.

To identify possible covariates of the sleep patterns, we merged the demographic data with the bed mat data. We also inferred the days of the week and the months of the year from the date as possible seasonal covariates.

This preprocessing led to 22,244 distinct binarised time series across patients where all 24 hour of the day were recorded by the sleep mat. About 100 nights contained fewer than 24 measurements after this preprocessing procedure, which we excluded from the analysis.

**3.3.2 Model**

To learn a generative model of the binarised sleep patterns, we assume that the presence or absence in bed of a patient for each hour can be captured by a Bernoulli distribution

$$p(x_k|z) = \theta_k(z)^{x_k}(1 - \theta_k(z))^{1-x_k},$$

(8)
where \( \theta_k \in [0, 1] \) is the probability of a patient to be in bed at hour \( k \). \( x_k \in \{0, 1\} \) is a random variable, indicating whether a patient was in bed or not. As a result, the daily sleep pattern of a patient is given by

\[
p(x|z) = \prod_{k=1}^{24} p(x_k|z),
\]

where we define \( x = (x_1, x_2, \ldots, x_{24}) \). The dependence of the hourly probabilities \( \theta(z) \) on the latent variables \( z \) is learned by the decoder network. In this implementation we chose to encode the latent space representation in a 10 dimensional space, i.e. \( z \in \mathbb{R}^{10} \).

The generative distribution of the latent space representation \( p(z) \) was chosen to be a multivariate standard normal distribution with covariance matrix \( \Sigma = \text{diag}(1,1,\ldots,1) \).

The encoded distribution of \( z \) for any given data \( x \), \( q(z|x) \), also known as the guide, was modelled by a multivariate normal distribution

\[
q(z|x) = \mathcal{N}(z|\mu(x), \Sigma(x)),
\]

where, similarly to \( p(z) \), the covariance matrix was assumed to be diagonal. The mapping from the data \( x \) to the parameters of the latent space distribution \( \mu \in \mathcal{R}^{10} \) and \( \Sigma \in \mathcal{R}_{>0}^{10 \times 10} \) is learned by the encoder network.

The encoder network was implemented by two fully connected layers which are connected by a ReLu activation function. The first fully connected layer takes the 24 dimensional binary sequence and returns 102 derived features. After passing through a ReLu activation, the second fully connected layer constructs the 10 means and 10 log-variances of the latent variables from the derived features.

The decoder consists of two fully connected layers, which are connected by a ReLu activation function. The final output of the second fully connected layer is transformed by a sigmoid activation function. This ensures that the final output of the decoder is between 0 and 1, and can therefore parametrise the Bernoulli distribution. In particular, the first fully connected layer takes a sample \( z \) from the encoded guide distribution \( q(z|x) \), which is defined by the means and log-variances produced by the
encoder and returns $10^2$ derived features. The second fully connected layer then constructs 24 probabilities to be in bed for the 24 hours from the derived features.

As a result, one pass through the VAE encodes data $x$ in terms of means $\mu$ and variances $\Sigma$ of the guide $q(z|x)$, samples a realisation of the latent space representation from the guide, decodes the latent space representation into probabilities to be in bed for each hour of the day $\theta_k$, and finally reconstructs the input data by sampling from the Bernoulli distributions $p(x|z)$.

The network is trained by minimising the loss

$$\log p(x|z) + KL(q(z|x)||p(z)),$$

where the first term represents the reconstruction loss and the second term the generative loss. The first term evaluates the likelihood of the input data $x$ under the encoded latent space representation $z$, and the second term calculates the Kullback-Leibler divergence of the encoded guide distribution to the generative distribution $p(z)$.

**Implementation & Training**

We implemented the VAE model in pytorch, please find a notebook on GitHub. The following resources were useful.

1. Pyro's tutorial on Variational Autoencoder.
2. New York University’s tutorial on Variational Autoencoders.

We split the available data into 8:2 training and test set split, where we did not randomise the temporal order of the data, in anticipation that consecutive nights might be more correlated than distant nights. Instead we identified a date that splits the data into a training and test set. The most recent data was used as test data.

The model was trained for 50 epochs using the Adam optimiser at a learning rate of $10^{-3}$. The model started with a test loss of 16.6874 and a training loss of 13.3319 and finished with a test loss of 4.8293 and a training loss of 5.9451. The training showed no signs of under- or overfitting.
Results and discussion

The trained VAE is able to generate visually similar sleep patterns to the data. To understand whether the observed sleep patterns can be explained by any covariate, we embedded the 10 dimensional latent space representations of the test data in 2 dimensions using T-SNE, see Figures 23, 24, 25, 26 and 27. We can see that the latent space representation of the sleep patterns form patient 10 and 11 form distinct clusters in the bottom left quadrant of the figures, and one large cluster in the center. These clusters were learned by the VAE and suggest that there is some structure in the sleep patterns. In figures 23 - 27, we labelled the embedded latent space by sex, age, diagnosis, days and months to identify whether any of these covariates explain the sleep patterns.

It appears that the structure in the sleep patterns cannot be explained by any of the available covariates individually. More complex analysis of combining the covariates and considering confounders may constitute natural next steps of exploration, however, we leave it to future work due to the time constraint of Data Study Group.
Figure 24: 2 dim. T-SNE embedding of latent space labelled by age.

Figure 25: 2 dim. T-SNE embedding of latent space labelled by diagnosis.

Figure 26: 2 dim. T-SNE embedding of latent space labelled by days.
4 Future work and research avenues

There are several promising, and arguably necessary, directions and lines of inquiry for future work.

4.1 Exploring data and application

4.1.1 Dataset collation

In solving Question 1.a, we used a master dataset that is a collation of the demographics and summary datasets. While a good start, any deployable model must be trained on a dataset consisting of all the features at our disposal: daily heart-rate and respiration measurements, external light intensity, temperature and wind speed measurements, knowledge of whether or not the subjects were snoring, their medical diagnosis, MMSE and NPI values. As the very first step in future work, we will collate all six datasets made available to us, into one master dataset sampled on a daily cadence.

4.1.2 Feature importance

One of our goals in this work is to understand the physical mechanisms that yield high and low time-in-bed values so that we can propose recommendations to change relevant input features to improve a subject’s sleep quality. To accomplish this, we need to understand the insights that the ML models decision making processes reveal, and ascertain that the
features deemed important by the model make sense physically. To this end, we propose using Shapley values, Integrated Gradients, Expected Gradients, and Expected Hessians \cite{LL17, LEC+20, JSL20, GLN21}. Attribution scores for input features quantify both the magnitude and direction of its contribution to the predictions, whereas interaction scores quantify the inter-dependence of pair-wise features, and their synergistic impact in predicting the output.

### 4.1.3 Diagnosis decline detection

While exploring the variational autoencoder in §3.3 we thought of the idea of learning a latent representation of each patient until the last current status of illness decline. The assumption is that the model which compresses and reconstructs the known data will fail to do so when it will encounter anomalous data. The reconstruction error can therefore be a sign of illness deterioration presence, or at least a sign that something has changed in the patient newly acquired data.

### 4.2 Exploring model evaluation strategies

#### 4.2.1 Measuring prediction uncertainty

In Question 1, we used a single NGB\textsc{OOST} model with a Gaussian likelihood for the output. The uncertainty ($\sigma$) obtained for each prediction there was the \textit{aleatoric} uncertainty. To obtain epistemic uncertainty, we will create multiple such models and feed them different subsets from the training set, and average their predictions thus. For each sample and model, let $\mu_m$ and $(\sigma_m)^2$ respectively denote the predicted mean and variance. To obtain the final mean predicted time-in-bed, we will average the individual predicted means:

$$ \mu = \frac{1}{M} \sum_{m=1}^{M} \mu_m \quad (12) $$

Aleatoric uncertainty is the average of the individual predicted variances
variances, calculated as [CLLO18]:

\[ \sigma_{al}^2 = \frac{1}{M} \sum_{m=1}^{M} \sigma_{m}^2, \]  

(13)

while epistemic uncertainty is the variance of the predicted means:

\[ \sigma_{epis}^2 = \frac{1}{M} \sum_{m=1}^{M} \mu_m^2 - \mu^2 \]  

(14)

The total uncertainty is computed by adding Equations (13) and (14) in quadrature.

4.2.2 Performance metrics

In Question 1.a, Section 3.1.1, we present metrics to quantify the quality of predicted means. However, when working in a probabilistic setting, it is also important to leverage metrics that are specifically geared towards quantifying the quality of predicted uncertainties. We recommend using two additional metrics to this end: average coverage area (ACE) and interval sharpness (IS). ACE is a measure of the how well the confidence interval captures the realised distribution of predictions.

ACE = 0, for 95% confidence intervals, implies that exactly 95% of them cover the true time-in-bed values, ACE < 0 implies incomplete coverage, and ACE > 0 implies over-complete coverage. At the same time, ACE doesn’t give us a sense of the concentration of the error. It’s possible to have infinitely large errorbars and still have ACE = 0. Interval sharpness/interval score (IS), ranging between 0 and 1, judges the sharpness of predicted uncertainties. We thus would like concentrated predictions (IS = 0) that simultaneously offer adequate coverage (ACE = 0).

4.3 Exploring models

4.3.1 Uncertainty-based prediction calibration

While calculation of both aleatoric and epistemic uncertainties is imperative for a complete prediction and hence downstream use in
scientific decision-making, they can at times mislead the practitioner into a false sense of overconfidence [LPB17, ZH20]. As a necessary component of any probabilistic prediction pipeline, it is important to calibrate our uncertainty estimates to more closely match the true distribution of errors. For example, we would want to ensure that that 95% confidence intervals for time-in-bed predictions contain the true time-in-bed values $\sim 95\%$ of times. To accomplish this we recommend using the methodology of [ZH20] to learn how to post-hoc fix the predictions on the predictions on the validation set, and using the learned knowledge to fix the predictions on any future test samples.

4.3.2 Modelling the physical process using Deep Koopmann

Neural networks algorithms provide a powerful framework to understand high dimensional data. Especially when symmetries of the data are exploited, neural networks often provide the current state-of-the-art across many machine learning tasks. For images such symmetries are, for example, translation and scale invariance, which are efficiently exploited by convolutional neural network architectures. For audio and text data, recurrent neural networks have proven to be the most useful architecture, as they exploit the sequential nature of the data. Implementing invariances of the data explicitly in the algorithm enables neural networks to reach good predictive performance with less data and, more importantly, make extrapolations to out-of-sample data more robust. In this section we discuss a recently proposed deep learning architecture which efficiently exploits the ‘symmetries’ of time series data.

Time series data, such as the heart rate measurements of a patient, can be understood as the measurements of a dynamical system

$$x_{k+1} = F(x_k),$$

where the dynamical system reflects the biological processes in a patient’s body. $x_k \in \mathbb{R}^d$ are the dynamical states of the system at time $t_k$, and $F$ is a non-linear map that propagates the states from one time point to the next. We restrict our attention to discrete-time dynamical systems where all time points are equidistant $t_{k+1} = t_k + \Delta t$, as this reflects the nature of the available sleep time series data. $\Delta t$ defines the temporal resolution of the data.
Conceptualising time series data as measurements of a dynamical system illustrates that one ‘symmetry’ of time series data is the way how measurements at one time point are related to the next. The time propagation function $F$ is an invariant of the dynamics. Thus, any efficient neural network architecture for time series data will likely estimate $F$ directly or indirectly. A popular neural network architecture for time series data is, for example, a recurrent neural network. Recurrent neural networks sequentially predict data points based on the preceding data points. In other words, they propagate a data point $x_k$ to the next data point $x_{k+1}$. This illustrates the fact that the most popular architecture for time series does indeed estimate the time propagation function $F$.

However, while the function $F$ may be invariant for any given dynamical system, $F$ is, in general, a non-linear function of the current dynamical state. This makes the learning of $F$ extremely challenging for two reasons. Firstly, many data points are required to learn non-linear mappings and, secondly, non-linear functions do not trivially translate to out-of-sample scenarios. That is, an estimate of $F$ learned by a recurrent neural network may perform well for all dynamical regimes represented in the training data, any extrapolation to other dynamical regimes will, however, likely perform poorly.

This limitation of non-linear dynamics does not apply to linear dynamics

$$x_{k+1} = M x_k,$$

where $M \in \mathbb{R}^{d \times d}$ is a constant matrix. Linear dynamics propagate the states $x_k$ to the next time point $x_{k+1}$ in the same way for all dynamical regimes. This simplifies the estimation of $M$. It also guarantees perfect extrapolation to out-of-sample data as linear dynamics do not leave room for the decay of existing dynamical features or the emergence of new dynamical features in unobserved dynamical regimes.

Koopmann showed in his seminal paper in 1931 that any non-linear dynamical system can be mapped to an infinite dimensional representation in which the dynamics are linear [Koo31]

$$y_{k+1} = K y_k,$$

where $y_k \in \mathbb{R}^\infty$ are the dynamical states in Koopmann space and $K \in \mathbb{R}^{\infty \times \infty}$ is the Koopmann operator that linearly propagates the $y_k$ from one time
point to the next. \( g \) is a non-linear map that translates the original states \( x_k \) to their Koopmann representation \( y_k \). Leveraging the Koopmann representation of dynamical systems, a time series machine learning task can be translated into the estimation of the non-linear map \( g \) and its inverse \( g^{-1} \), as well as the linear time propagation operator \( K \). Then, any prediction can be performed by first mapping the current state \( x_0 \) to Koopmann space \( y_0 \), then exploiting the linear dynamics to propagate the state to any desired time \( y_k \), and a final back transforming to physical space \( x_k \).

It is not obvious that learning the Koopmann representation of a non-linear dynamical system will be more fruitful than learning the non-linear dynamics directly. Especially, since we seem to trade off learning a finite dimensional non-linear map \( F \), just to learn two non-linear maps and an infinite dimensional linear map \((g, g^{-1}, K)\). However, Lusch, Kutz and Brunton demonstrated that an autoencoder-like architecture can successfully learn the dynamics of complicated non-linear systems, such as a high-dimensional non-linear fluid flow, by exploiting its Koopmann representation [LKB18]. This Deep Koopmann architecture also seems to outperform architectures that directly estimate the non-linear dynamics \( F \).

In Lusch et. al.’s approach the mapping to Koopmann space \( g \) and its inverse \( g^{-1} \) are learned by an autoencoder-like architecture by inputting the complete time series at once. An encoder, in form of a neural network, maps the time series \((x_0, x_1, \ldots, x_{n-1})\) to its Koopmann representation \((y_0, y_1, \ldots, y_{n-1})\), where \( n \) is the length of the time series. Here, the Koopmann states are represented in the eigenbasis of the Koopmann operator. The Koopmann operator propagates the Koopmann states to the next time points \((y_1, y_2, \ldots, y_n)\), which are then transformed back to physical space by a decoder network \((x_1, x_2, \ldots, x_n)\). Thus, the inputs and outputs of the architecture is the full time series and the full time series propagated by one time step. The encoder and decoder learn the maps \( g \) and \( g^{-1} \).

The Koopman operator is learned by an additional architecture. In its eigenbasis \( K \) is fully determined by its eigenvalues \( \lambda_{\pm} = \mu \pm i\omega \). Each
pair of eigenvalues gives rise to a block matrix

\[ B(\lambda_+, \lambda_-) = e^{\mu \Delta t} \begin{bmatrix} \cos \omega \Delta t & -\sin \omega \Delta t \\ \sin \omega \Delta t & \cos \omega \Delta t \end{bmatrix}, \]  

which together represent the Koopman operator in its Jordan block form

\[ K = \text{diag}(B_1, B_2, \ldots, B_\infty), \]

where each \( B_i \) corresponds to an eigenvalue pair. In practice, the infinite number of eigenvalues cannot be learned. However, it turns out that for many dynamical systems only few eigenvalues with \( \mu \geq 0 \) exist. Eigenvectors with \( \mu < 0 \) decay exponentially, and therefore one can obtain a good approximation of the Koopman operator by only focussing on the subspace spanned by the eigenvectors with \( \mu \geq 0 \). In other words, the Koopman operator can be represented by a finite matrix \( K = \text{diag}(B_1, B_2, \ldots, B_k) \), such that it becomes feasible to learn the Koopman embedding with the above described network architecture. In Lusch et. al.'s approach the finite number of relevant eigenvalues of the Koopmann operator are learned from the outputs of the encoder with a separate neural network. For a diagrammatic illustration of the network full Deep Koopman architecture, please refer to figure 2 in [LKB18].

We believe that the Deep Koopmann network architecture can be used to efficiently learn the dynamics of the heart rate and respiration across patients. This may provide a framework to generate virtual patients with realistic heart rate and respiration patterns. The translation of the dynamics to a finite number of eigenvalues of the linear Koopmann dynamics may also serve as an efficient method to extract relevant features of the dynamics for other machine learning tasks. Such tasks may involve understanding the influence of inter-individual variability and the disease state, as well as external factors, such as temperature and light on the sleep dynamics of patients. Finally, the Deep Koopmann model may also serve as a basis to optimise sleep conditions for individual patients by first learning the Koopmann representation of an individual patient, then understanding the influence of controllable factors on the eigenvalues of the Koopmann operator, and finally optimising these factors to obtain a desired sleep pattern.
4.3.3 Models and hyperparameter optimisation

In Question 1.a, we use a single NGBoost model [DAD+19] to predict time-in-bed, using the default hyperparameters from xgboost-distribution\(^3\) package. In future, we plan on leveraging other gradient-boosted machine learning algorithms such as catboost [PGV+18], XGBoost [CG16] and LightGBM [KMF+17] which can easily handle missing values. We also plan on leveraging powerful deep learning models which have been shown to perform as well, if not better, on tabular datasets: mixture density network [Bis95, CLLO18, GDF+21], TabNet [AP19], deep ensemble [LPB17], and deep evidential regression network [ASSR19], among others. For obtaining optimal predictions, we also plan on using any of various open-sourced python packages for hyperparameter optimisation (HPO) (e.g., optuna\(^4\)) to explore the parameter space of these various models and pick sets of parameters that most reduce the loss on the validation set.

4.3.4 Generalisation

At its very core, this was a problem of generalisation on unseen, out-of-distribution samples. We plan on leveraging several methods in this area: ensembling predictions from multiple different machine learning and deep learning models [SZA21], sample re-weighing and data augmentations for improving both interpolation and extrapolation [YZC+21, LWL+20], and adversarial data augmentation [QZP20].

4.3.5 Multiple-multivariate time series forecasting

A future work for question 1.a would be to implement a multiple-multivariate time series forecasting models to forecast for each patient by taking the inputs from all the patients. More attributes and hourly data can be added to daily data to improve prediction. The implemented time series model could be improved by using indoor and outdoor features such as light, temperature and weather along with the demographics for each patient. To achieve this, a multiple-multivariate

---

\(^3\)https://github.com/CDonnerer/xgboost-distribution

\(^4\)https://optuna.org/
A time series forecasting model is required which would take all patients historical data along with the other attributes that influence the sleep and wakeups to forecast for each patient. The goal is to infer numerous multivariate connections between co-occurring time-series by assuming that temporal data is influenced by not just by inner factors and intra-temporal relationships, but also by outer variables and inter-temporal relationships [SHB+55].

5 Team members

Xin Yang is a PhD student at School of Engineering and Materials Science, Queen Mary University of London. She contributed to this project by working on Question 1.

Jelena Sucevic is a Postdoctoral Research Scientist at the Department of Experimental Psychology, University of Oxford. She contributed to this project by working on Question 2.

Rohit Sahoo is a Software Engineer with a bachelor’s degree in Computer Engineering from the University of Mumbai. In his academic life, he worked and published several papers on Machine Learning and Artificial Intelligence. His current interests are in Generative Modelling and Time Series Forecasting using Deep Learning. He contributed to this project by working on the multivariate time series forecasting approach in Question 1 and proposing future work in Section 4.

Tiago Martinelli is a physicist and PhD candidate at University of Sao Paulo, where he develops new theoretical informational-based criteria to quantify causal emergence from data-generated complex systems. In his previous academic life he worked with quantum foundations, more specifically in the emergence of quantum information. He contributed to partially solving Question 1 and by proposing a path in Question 3.

Chang Yuan Li is a statistics PhD student from University of California, Santa Barbara working on Question 1.

Sankalp Gilda is a Machine Learning Forecasting Engineer at Fermata Energy, where he utilizes the latest in time series data mining and forecasting to push forward the Vehicle-to-Grid (V2G) space. During his tenure as a PhD candidate in Astrophysics at the University of Florida, he specialized in applications of machine learning techniques to solve problems in galaxy formation and evolution, spectral characterisation, and telescope automation. He contributed to partially solving Question 1, and proposing lines of inquiry for future experimentation in Section 4.
Gabriela-Nadia Domide is an engineer with an MSc in Computer Science, with a specialisation in machine learning at the University of the Basque Country. She is a core member of a collaborative research project that aims to provide support for developed products based on the moral questions raised by artificial intelligence. She contributed to this project by working on Question 2 and proposing future work in Section 4.

Celeste Damiani is a mathematician working as a Postdoctoral Data Scientist at Queen Mary University of London, where she focuses on using AI for risk assessment at mammography screening. In her previous academic life she worked in geometric topology, more specifically in the fields of braid groups generalisations and knot theory. She contributed to this project acting as a facilitator for this group together with David and working on Question 2.

David Augustin is a physicist/mathematician working towards a doctorate in Computer Science at the University of Oxford, where he develops machine learning approaches which help to personalise treatment strategies in clinical practice. He contributed to this project by facilitating the group together with Celeste and by working on Question 2, and by proposing future work in Section 4.

References


[KMF⁺17] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly...


