Forecasting Short-term Mood for Depressive Patients in Therapy

Vincent Jansz

A research paper presented for the Master Business Analytics

Supervised by Dr. Mark Hoogendoorn

Vrije Universiteit Amsterdam

Faculty of Science

Business Analytics

De Boelelaan 1081a

1081 HV Amsterdam

Januari 2019

Preface

This report is written for the master's program Business Analytics at the Vrije Universiteit Amsterdam. The purpose of this course is to carry out research that combines business, mathematics and informatics with a business administration approach. After consulting with my supervisor, I was able to find a relevant topic that lies in the domain of social issues for which advanced machine learning techniques were suitable as a solution. I would like to thank my supervisor Dr. Mark Hoogendoorn, who made this research possible, for his feedback and guidance throughout the completion of this course.

Summary

This paper examines forecasting the mood for depressive patients in therapy on a short-term basis. Specifically, it studies the impact of adding therapy and messaging data, not used in previous research, on the performance of three single-layer models: recurrent neural network (RNN), gated recurrent unit (GRU), long short-term memory network (LSTM). Due to large variance in patients' behavior, clusters of patients are formed using hierarchical agglomerative clustering before training models. In addition, this paper studies the effect of varying the amount of timesteps of historical data as input. Out of the three models, the worst performing model is the RNN, with the GRU and LSTM having near identical performance. No significant positive effect on the performance of the models is observed by adding messaging and therapy data or by adding more timesteps of historical data. Additionally, no model was able to outperform the baseline of the persistence model. Possible causes are the architectures of the models, the manually extracted features, the clustering of the patients or the preprocessing performed on the data.

Contents

1 Introduction

1.1 Goal

With over 300 million people suffering from it worlwide, depression is the number one mental health disorder globally [1]. An increased risk of mortality is associated with this disorder for both major depression and subclinical forms, making it a life-threatening disorder [2]. Alongside these individual effects, there is also a larger social effect, namely the economic burden. In Europe alone, the total annual economic cost of depression and its treatment in 2004 is estimated at €118 billion, 1% of the total GDP of Europe [3]. Therapy for this mental disorder is evolving from the conventional psychiatrist counseling to self-help therapy and a mix of both, such as the E-COMPARED project. The rapid progression in technology leading to smartphones and widespread availability of fast internet connections made this evolution possible. Nowadays, therapists have the option to contact patients through their phones and are able to intervene in real-time. This is precisely how the EMA (Ecological Momentary Assessment) dataset from the E-COMPARED project (European COMPARative Effectiveness Research on blended Depression treatment versus treatment-as-usual) is produced: self-therapy through modules in an app, self-reporting of the mental state and questionnaires of a patient, and real-time intervention from the therapist.

This data can be used to forecast the mood of depressive patients in order to warn the patients themselves, the therapist or even relatives. However, several papers have already demonstrated that this forecasting can be challenging [4, 5, 6, 7, 8]. One of the challenges is the large differences in individual mood reporting. Making a model per individual often leads to shortage of training data whereas creating a single model leads to low performance. A possible solution to this is grouping patients on their rating behavior and then training models on those groups [7, 8]. Another challenge is the limited value of historical mood in predicting future mood. This problem can perhaps be tackled by smart feature engineering such as incorporating extra therapy and message information [5], which are also included in the E-COMPARED dataset. Both of these methods will be tested in this paper alongside using advanced machine learning models capable of capturing information from historical input.

Most papers that try to forecast mood among depressive patients only use the self-reported data from the patients. This is mostly due to an absence of extra data recorded from these depressive patients. On top of that, the papers also tend to use only a long short-term memory network (LSTM) and gated recurrent units (GRU) [6, 8]. The goal of this paper is to predict the mood of depressive patients using historical self-reported data, therapy data and messaging data, on a short-term basis. For this purpose, this paper will compare the effectiveness of a LSTM, GRU and a regular recurrent neural network (RNN). The dataset used in this paper comes from the E-COMPARED project. The research question will be: ”How accurately can we predict a depressive patient’s mood given their self-reported, therapy and messaging data?” In order to properly answer that, two sub-questions will be defined: ”Does including messaging and therapy data improve the performance of forecasting mood, given their self-reported data?” and ”Which of three models LSTM, GRU and RNN, performs the best in predicting a depressive patients’ mood given their self-reported, therapy and messaging data?”.

1.2 Structure

The report is structured as follows: first, relevant literature regarding mood prediction using mobile data and time series prediction in general will be examined in Section 2. Then, the complete methodology of this paper, including modeling techniques and metrics, will be discussed in Section 3. Next, the collection, transformation, clustering and analysis of the dataset will be shown in Section 4. Before the results are discussed, information about the benchmark creation and model setups will be shown in Section 5. After that, the results will be shown in Section 6 and the discussion and conclusion will be presented in Section 7.

2 Literature Review

This section will give an overview of relevant literature on forecasting mood of depressive patients and forecasting timeseries in general. We will start by examining papers discussing time series in general in Section 2.1 and will then move to papers that discuss the problem at hand in more detail in Section 2.2.

2.1 Timeseries forecasting

The papers examined here are papers that employ the same techniques used in this paper regarding timeseries forecasting, namely RNN, GRU and LSTM. These papers study timeseries forecasting in various domains, demonstrating the effectiveness of the chosen models on varying problems.

In the paper written by Weron [9] they examine the state-of-the-art methods used for electricity price forecasting. The authors note that the price of electricity depends on weather and other external influences, much like mood. For the purpose of forecasting the electricity price, they look at five different types of models: multi-agent models, fundamental models, reduced-form models, statistical models and computational intelligence models. For the purpose of our research, we are not interested in the last category as a whole, but rather a sub-category: recurrent neural networks. The authors note that the major strengths of recurrent neural networks are, alongside their ability to handle complexity and non-linearity, their flexibility. In their survey, they find that computational intelligence models in general are better at forecasting the electricity price than statistical models.

Sainath et al. wrote a paper called 'Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks [10]. In their paper, they examine the effectiveness of combining convolutional neural networks (CNN) with LSTM and deep neural networks (DNN). The datasets used in their paper are multiple large vocabulary sets, ranging from 200 to 2000 hours. They compare the individual models and combinations, using various weight initialization methods and feature engineering setups. Their proposed new modeling architecture called CLDNN outperforms the baselines, gaining 4-6% performance in terms of word error rate (WER) over the strongest individual model, LSTM. Their proposed CLDNN is mainly useful when there is an abundance of data, due to the large amount of weights that need to be optimized.

In 2017, Lipton et al. wrote a paper on diagnosing patients using LSTM [11]. The dataset used in their paper contains multivariate timeseries of observations. It includes sensor data and lab results that are obtained each time a patient makes a visit to the intensive care unit. The authors note that, due to the varying length of the sequences and missing data, mining their dataset is challenging, but that RNN are powerful solutions to their problem. Their best performing setup is an LSTM network with a classification performance of 0.8643 in terms of micro AUC compared to 0.7128 of the base rate model. They note that their model, trained solely on raw time series, outperforms all their baselines, which include logistic regression and multilayer perceptrons trained on hand-engineered features.

2.2 Mood forecasting

The papers examined in this section are papers closely aligned with the work in our research. Since only five papers are examined, we choose to examine them thoroughly in this section.

The first work that is examined is a paper by John Pastor and Ward van Breda from 2015 [4], in which they try to accurately predict future mood, while also examining what the most important features in their models are. In their work they use a dataset consisting of 27 users with data collected through a mobile phone app over a period of six weeks, in which mood is rated with a number between one and ten. They apply an ARIMA, linear model, regression tree, random forest and support vector machine (SVM) to their dataset. To gauge performance of the models, they calculate the mean squared error and the accuracy. In order to measure accuracy, they transform the predictions by rounding them to one decimal and then check if the prediction lies within a certain interval around the actual prediction, in order to determine if it is correct or not. The baseline used in their paper is the overall mean of mood. Only the random forest and SVM models outperform the baseline (64.359%, 0.441), with the SVM performing the best in terms of their accuracy (68.880%) and the random forest model in terms of mean squared error (0.402). Despite the variable importance varying significantly between patients, they find that the most important features are statistics about mood itself, such as lagged variables and the mean.

The next paper that is examined is written by Mehrotra et al. in 2016 [5]. In their paper, they create an approach for the monitoring of depressive states and display a relation between smartphone interaction and depressive states. For this, they use multi-modal sensing via smartphones and various interaction features such as location, call, sms and usage logs. Their dataset contains 25 participants with data collected over a period of 30 days. They implement Kendall's Rank correlation coefficient in order to analyze the mood of the depressive patients and the association with notification processing and with phone usage. Features that correspond to notification processing are features such as the acceptance rate of notifications, the average amount of time a notification is left unopened, the average amount of time between opening and deciding on the notification, and the average amount of time between delivery and deciding on a notification. Features included in the phone usage metrics are, for example, the total amount of launched applications, total application usage time, total time the phone was used, and the number of times the phone was unlocked. Their results indicate that there is a modest correlation between the depressive state and notification metrics calculated over 14 days. This also holds when the metrics are calculated over 7 days but becomes far weaker when it is calculated solely on the previous day. The resulting correlations for the phone usage features are weaker. There is only moderate correlation when the metrics are calculated over 14 days, low correlation when calculated over 7 days and no significant correlation when calculated over 1 day. From these results they conclude that forecasting performance benefits from using historical phone interaction data, to a certain extent.

Suhara et al. wrote a paper in 2017 on forecasting severly depressed days and do so by developing a recurrent neural network model with embedding layers [6]. The research goal of their paper is to predict whether the user has at least one severely depressed day in the upcoming *n* days given information from the previous *k* days. Their definition of a severely depressed day is when a user *"has negative feelings all day and exhibits inactive behavior in which he or she avoids leaving home"*. In this paper they use a relatively large dataset containing almost 2400 patients, who are all self-declared depressive patients. It contains measurements of mood, behavior and sleep, from which they then create two datasets: a dataset containing only mood and a dataset containing all features. They compare both datasets on two different models, namely a SVM and a long short-term memory network (LSTM). They start by testing on three scenarios: fixing *k* at 14, due to the nature of most of the questions in their data, while setting *n* at 1, 3, or 7. From this experiment they conclude that predicting one day ahead $(n = 1)$ is the best setting, as the AUC-ROC decreases as *n* increases. This follows the intuition that the further in the future you try to forecast, the harder it gets. They also find that the LSTM model outperforms the SVM model in each case and continue forward with just the LSTM model. Some features in their model show no positive influence on the performance whereas they did in previous studies. They conclude that this results from fitting one large model over all users instead of splitting up the users as individuals or in groups. Next, they fix *n* at 1 and vary *k* from 1 to 21. From the results they conclude that using two weeks of historical information is enough to predict severely depressed days. Another conclusion drawn is that the most recent day contains the most predictive power, but the previous 13 days before that also have significant predictive power, with the same day of the week having extra predictive power. They examine this further and show that Fridays, Saturdays and Sundays are generally the happier days, whereas Monday through Thursday are the overall sadder days.

In 2018 Jaques et al. also performed research on the subject of forecasting mood[7]. In their paper they propose a deep neural network (DNN) and a Gaussian process (GP) to predict a person's mood tomorrow. They compare the generic models to a more sophisticated counterpart. For the DNN they add multitask learning and for the GP they add domain adaptation. The dataset used in their paper contains data of 206 undergraduate students, with 30 days of data per participant. The dataset contains a vast amount of information, from which they engineer 343 features in total. These features can be split up into six categories, namely physiology, location, phone, surveys, weather and mood. They then scale down their dataset to only the participants with at least 25 days of data from all sources. The resulting dataset then contains 69 participants, a significant reduction in size. To this data they fit the four models and compare them. Their results show that both improved models outperform their generic counterparts in terms of mean absolute error, but only the improved DNN having a significant performance boost. On top of that, they investigate if the improved models are better at capturing the underlying mood of participants. They again find that both improved models outperform the generic models in terms of intraclass correlation coefficients, with both improved models outperforming significantly. From these results they conclude that personalizing these models to participants significantly improves the performance of said models, if it is done in a principled way.

The final paper that is examined is written by Mikus et. al in 2018 [8]. Their objective is to create a model that predicts short term mood developments using adherence and usage data as additional predictors. For this purpose, they use two different modeling techniques, namely LSTM and Gated Recurrent Unit (GRU), and vary both model's architectures for a total of six different setups (single layer, multiple layers, added recurrent project layer). In addition, they create a benchmark using support vector regression with a radial basis function kernel. The dataset used in their paper consist of data between 9 and 425 days of 143 patients with a major depressive disorder. They split the dataset in two, one containing just the mood ratings and basic information such as nationality, one containing additional features they created through feature engineering. Before fitting the models, they create clusters from the patients, ultimately choosing 12 from the hierarchical clustering dendrogram they create. Next, they create three scenarios: train a model on all patients, train a model per cluster of patients, train a model per individual patient. They then train 18 models: both datasets, on all three three scenarios, using 3 different architectures and use the RMSE as metric. The obtained results indicate that there is little difference in the best variants of the models. Interesting to see is that the clustered model performs worse on the extended dataset (0.075) than on the original (0.066). Overall, they note that forming clusters of the patients slightly improves the performance (0.066) in comparison to the single model (0.070) and the individual model (0.086). They also note that the mood ratings themselves have the most predictive power and that a 7 day historical time window provides the best fit.

3 Methodology

In this section, the general methodology of this paper is explained. We will start by giving a short overview of the complete process, followed by theoretical background on the methods used in this paper.

3.1 Overview

First, the data is transformed from a timestamp format to a sequence of discrete timesteps per patient format and extra features are created. This results in missing values, as there are days on which a patient does not perform any activity or does not fill in his or her ratings. To fix this, missing value imputation is performed. After the missing value imputation is performed, the dataset is split up into clusters of patients. This is one of the ways, as explained in the literature study in Section 2, to balance the available data size and variance between users. After clustering is performed, the dataset is analyzed on correlation per cluster. The complete dataset is used to create two datasets: one containing all the features, which include the therapy and messaging data, one containing only the basic information of the patients and their self-reported data. Both datasets are then again split up into 50% training data and 50% testing data, per user sequence, using the first half of each sequence as training data and the second half as testing data. The data is then scaled using a so called 'MinMax Scaler', which linearly scales the data between 0 and 1. The scaler is fitted on only the training data and then used to scale both training and testing data, as scaling the entire dataset at the same time includes information about the test set in the training set [12]. Both datasets are then used for training multiple RNN, GRU and LSTM models.

3.2 Clustering

The two most common approaches for clustering are hierarchical clustering and centroid-based clustering. This section will shortly discuss the differences between these two categories, such as their parameters to be chosen and their advantages and disadvantages.

```
Data: set of $N$ points
Result: set of $k$ clusters
create a cluster for each separate point $x_i$;
create a proximity matrix;
while more than 1 cluster do
    merge the two least dissimilar clusters based on similarity metric;
    update the proximity matrix;
end
```

Algorithm 1: Hierarchical agglomerative clustering

Hierarchical clustering

Hierarchical clustering algorithms, in short, are methods that uses the idea that *distance* of objects to each other determines their likeliness: objects that are further away have less in common with objects that are closer by. These methods start with a seperate cluster for each datapoint and then merge the two closest clusters, iteratively, until there is one cluster containing all datapoints (agglomerative), or the other way around (divisive). Pseudo-code for the agglomerative version are displayed in Algorithm 1. The user will have to decide the threshold for which the clusters are no longer merged or split up or the maximum amount of resulting clusters. The algorithms distinguish from one another based on the type of distance metric and linkage criteria. Advantages are, but not limited to, the flexibility of choosing a distance or similarity metric and performance on different types of clusters in the data [13, 14]. One of the major disadvantages however is the time complexity ($\mathcal{O}(n^2)$) of the algorithm [13, 14].

```
Data: set of N points
Result: set of k clusters
choose the amount of clusters k;
initialize the center of each cluster $c_j$ randomly;
while change in centers or iterations left do
    (re)assign each point $x_i$ to the closest cluster based on cluster centers;
    update the centers of each cluster $c_j$
end
```

Algorithm 2: Centroid-based clustering

Centroid-based clustering

Centroid-based clustering algorithms try to find *k* cluster centers and assign every datapoint to one cluster by representing a cluster center with a single mean vector [13, 14]. They do so by starting with k cluster centers and moving these by minimizing the distance in the clusters, or sometimes maximizing the distance between clusters [13, 14]. In this case, the user has to determine how many resulting clusters there have to be upfront. Pseudo-code for a general centroid-based clustering algorithm are shown in Algorithm 2. Different algorithms exist with different initialization methods and different objective functions. These algorithms are generally faster than hierarchical algorithms ($\mathcal{O}(n \log n)$) but as a drawback have less flexibility and generally have low performance when the data has non-spherical clusters [13, 14].

Choice of clustering techniques

In this paper, we will try one hierarchical clustering technique, agglomerative clustering, and one centroid-based technique, k-means clustering, as both methods are the most common in their respective domain [13, 14]. We will then compare both results on the total number of clusters, the size of the clusters and the correlation between the feature and target variable per cluster. We will then choose one clustering method and use those to split up the patients in the dataset.

For agglomerative clustering, several linkage functions are available. The most common ones are the *single linkage*, *complete linkage* and *mean linkage* function, and *Ward's criterion* or *Ward's method*. Ward's method minimizes the total variance within clusters, which is exactly what we try to accomplish. We want to create clusters in which the there is as little variance possible in the habits of the patients regarding mood rating. Ward's method is formulated as follows:

$$d(u,v) = \sqrt{\frac{|v|+|s|}{T}d(v,s)^2 + \frac{|v|+|t|}{T}d(v,t)^2 - \frac{|v|}{T}d(s,t)^2}$$

, where $T = |v| + |s| + |t|$, u is the cluster merged from clusters s and t, v is an unused cluster and $| \cdot |$ is the cardinality of the given cluster. In essence, Ward's method calculates the resulting total variance within clusters for every possible merger and chooses the best merger, in a greedy manner. There are also many distance metrics available for hierarchical clustering. The most common distance metric is the *Euclidean distance*. It measures the distance between two vectors as the square root of the summed quadratic differences of the individual elements. Alternatives are the *squared Euclidean distance*, *Manhattan distance* or *maximum distance*. The choice of metric in this paper is the Euclidean distance, as it is the only metric suited for use with Ward's criterion.

For k-means clustering, there are also two parameters that needs to be chosen, namely the amount of clusters*k* and the distance metric. It is possible to also define the starting location of these clusters, but we will refrain from doing this. Instead, we will run several iterations of k-means with the same *k* and see which one results in the 'best' clusters: the clusters which result in the smallest *distortion*, which is defined as the sum of the squared distances between each observation and its corresponding centroid, and use the Euclidean distance as distance metric.

3.3 Time series forecasting

In this section the three types of model used in this paper are discussed. A brief conceptual explanation per model is given alongside their advantages and disadvantages regarding timeseries forecasting. For more information on all three models, there are more comprehensive papers and guides that epxlain the process in more detail, such as [15, 16, 17]. Including this information would increase the size of this paper significantly without benifit. All three models are versions of recurrent neural networks. In comparison to regular feed-forward neural networks, a connection is established between successive units along a sequence. This in turn allows the network to capture dynamic temporal behavior from a sequence, making them suited for timeseries forecasting. In order to 'learn', these models employ backpropagation through time, a more complex version of backpropagation found in feed-forward neural networks. Discussing this is outside the scope of this paper and we refer to external resources for more information, such as [15].

Basic RNN
The basic variant of recurrent neural networks is the basis on which the advanced models are built. The mathematical formulation is given in the set of equations below.

$$a^{<t>} = g(W_a \cdot [a^{<t-1>}, x^{<t>}] + b_a)$$
$$\hat{y}^{<t>} = g(W_y \cdot a^{<t>} + b_y)$$

Here, $x^{<t>}$ is the input vector at timestep t, W_i is a weight in the matrix of weights W, $a^{<t>}$ is the output at timestep t, $y^{<t>}$ is the prediction at timestep t, b_j are the biases and $g(x)$ is the activation function. The information that is passed along between the units is the output $a^{<t>}$ from timestep $t-1$ to t, allowing temporal information to travel from prediction to prediction. A common problem with this type of network is the 'vanishing gradient problem', which occurs when the gradient of the error approaches zero as the length of the input sequence increases. This is a result of multiplication of the activation outputs, as these typically take values between 0 and 1. The performance of this model type typically worsens as the length of the dependencies in the data increases.

GRU

A more complex variant of the basic recurrent neural network is the Gated Recurrent Unit (GRU) which is able to combat the vanishing gradient problem. In this model, there are two 'gates', the update gate Γ_u and reset gate Γ_r, that both have influence on the output and indirect influence on the prediction of the model. The set of equations below describes the model mathematically.

$$\begin{aligned}
\tilde{c}^{<t>} &= \tanh(W_c \cdot [\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c) \\
\Gamma_u &= \sigma(W_u \cdot [c^{<t-1>}, x^{<t>}] + b_u) \\
\Gamma_r &= \sigma(W_r \cdot [c^{<t-1>}, x^{<t>}] + b_r) \\
c^{<t>} &= \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>} \\
y^{<t>} &= g(W_y \cdot c^{<t>} + b_y)
\end{aligned}$$

Here, $x^{<t>}$ is the input vector at timestep t, W_i is a weight in the matrix of weights W, $c^{<t>}$ is the output at timestep t, $\tilde{c}^{<t>}$ is the candidate output at timestep t, $y^{<t>}$ is the prediction at timestep t and b_j are the biases. The increase in complexity allows the model to learn longer time dependencies better than the previously mentioned RNN, but comes at the cost of higher computational requirements.

LSTM

The last variant we will discuss is the Long Short-Term Memory network (LSTM). It is very similar to the previously mentioned GRU, but differs in two aspects. First, the reset gate has been split up into two gates: the forget gate Γ_f and the output gate Γ_o, and second, the addition of a memory cell $a^{<t>}$. The equations below formulate the model mathematically.

$$\begin{aligned}
\tilde{c}^{<t>} &= \tanh(W_c \cdot [a^{<t-1>}, x^{<t>}] + b_c) \\
\Gamma_u &= \sigma(W_u \cdot [a^{<t-1>}, x^{<t>}] + b_u) \\
\Gamma_f &= \sigma(W_f \cdot [a^{<t-1>}, x^{<t>}] + b_f) \\
\Gamma_o &= \sigma(W_o \cdot [a^{<t-1>}, x^{<t>}] + b_o) \\
c^{<t>} &= \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>} \\
a^{<t>} &= \Gamma_o * \tanh(c^{<t>}) \\
y^{<t>} &= g(W_y \cdot a^{<t>} + b_y)
\end{aligned}$$

Here, $x^{<t>}$ is the input vector at timestep t, W_i is a weight in the matrix of weights W, $c^{<t>}$ is the output at timestep t, $\tilde{c}^{<t>}$ is the candidate output at timestep t, $a^{<t>}$ is the memory cell at timestep t,

$y^{<t>}$ is the prediction at timestep t and b_j are the biases. The addition of the third gate and memory cell allow the LSTM to model even more complex patterns and typically outperforms the GRU, at the cost of being less computationally efficient.

Loss functions

The previously mentioned models have several loss functions available for regressive objectives, such as the *mean squared error* (MSE), *root mean squared error* (RMSE), *mean absolute error* (MAE). The MAE calculates the loss as the total sum of absolute errors between the predictions and true target values, but gives no additional weight to larger errors in the way that the MSE and RMSE do. The MSE calculates the loss as the total sum of quadratic errors and the RMSE is simply the square root taken over this loss. Due to the quadratic term in the loss function, both the MSE and RMSE penalize larger errors more, which is useful when the data exhibits spiky behavior. We choose to use the RMSE as the loss function over the MSE due to the RMSE being in the same unit size as the predicted value due to the square root factor.

4 Dataset

In this section, the collection, transformation, clustering and analysis of the dataset will be discussed. We will start by giving an overview of the origin of the data, followed by a brief insight into the raw dataset and then discuss what data is filtered out. Then, we will show which transformations are applied, followed by how the missing values are imputed. Next, the clustering of the patients is explained and the dataset is briefly analyzed for insights.

4.1 Collection

The dataset used in this paper stems from the E-COMPARED project [18]. The E-COMPARED project is a study in which patients received blended treatment for their depression, where blended treatment stands for a mix of online and offline contact with a therapist. It contains data from 8 research sites, from which 5 have additional log file data from the Moodbuster system. This system automatically logs system usage from the patients, such as which module is being looked at and at which specific page from said module. For this paper, we look at the data of the countries which do include Moodbuster data, as we're investigating if including therapy data significantly improves performance. We will also filter out patients that have less than 8 days of data as we want to vary the historical window between 1 and 7 days for input. This filtering results in a dataset with records of 8-888 days of 202 patients.

4.2 Insight

In this section, we will show the mood sequence from three random patients, which are shown in Figure 1, 2 and 3. Pay attention to the scale of the x-axis as it varies between the figures. Figure 1, the first of the three figures, shows a patient who's mood rating fluctuates between roughly 8 and 9, with some extreme values towards 6.5. The patient's mood does not show a clear up- or downwards trend over a period of 170 days. In contrast to that is the second patient's graph. Figure 2 shows a relatively fast upwards trend, starting at around 4 and ending around 7, with bigger fluctuations than the first patient. The third graph is also completely different from the first two: An upwards 'peak' from roughly 1.25 to around 2.5, then slowly decaying back to 1, without any fluctuation. These three figures illustrate the significant amount of variance between different patients.

4.3 Transformation

First, a dataframe describing the rating habits of patients is created. This information can be used further in this paper to create the clusters of patients to train the models on. For this purpose, we calculated several statistics, which are shown in Table 1.

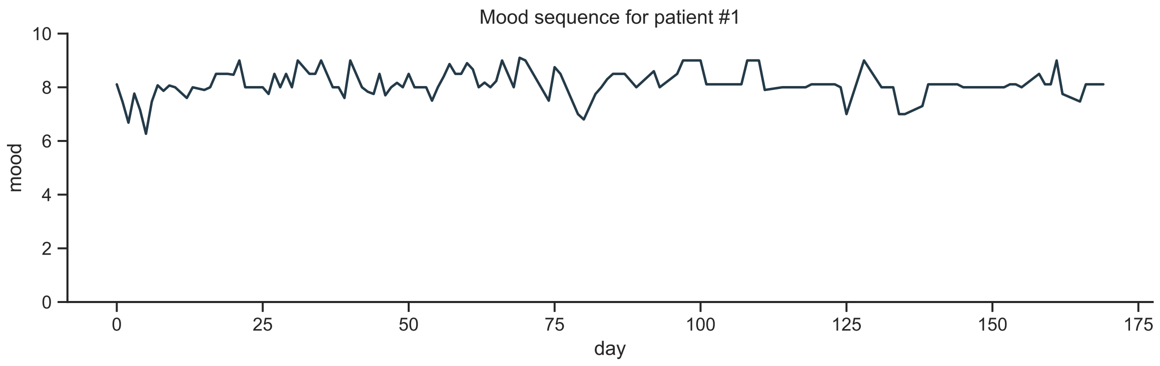


Figure 1: Mood sequence for random patient 1, spanning 170 days

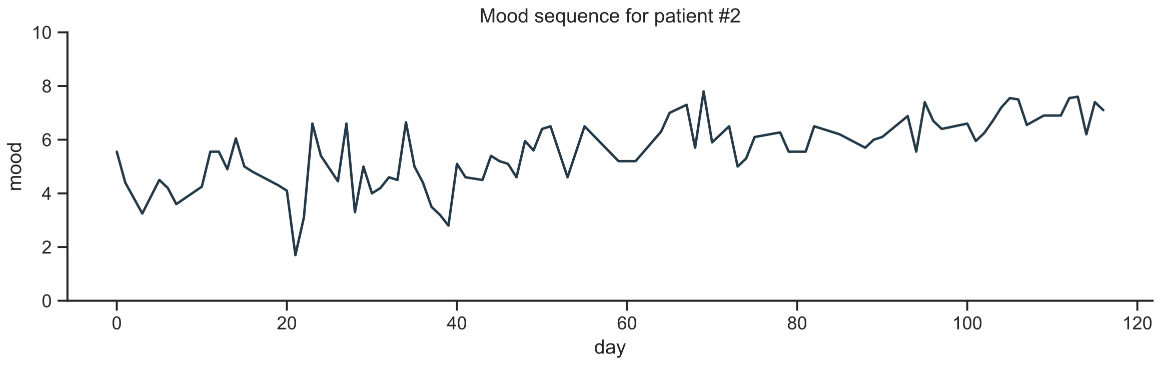


Figure 2: Mood sequence for random patient 2, spanning 120 days

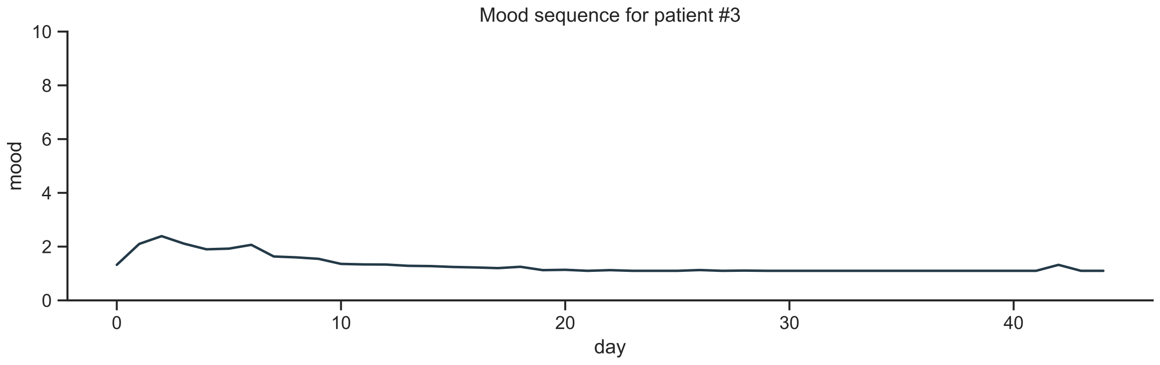


Figure 3: Mood sequence for random patient 3, spanning 45 days

Table 1: Description of statistics of mood ratings behavior

Statistic	Description
Count	Total number of days with daily average mood ratings
Maximum	Maximum of daily average mood ratings
Minimum	Minimum of daily average mood ratings
Mean	Mean of daily average mood ratings
Median	Median of daily average mood ratings
Standard Deviation	Standard deviation of daily average mood ratings
First quantile	First quantile of daily average mood ratings
Third quantile	Third quantile of daily average mood ratings
Percentage	The percentage of days with daily average mood ratings
Max separate	Maximum consecutive amount of days without daily average mood ratings
Max consecutive	Maximum consecutive amount of days with daily average mood ratings

Table 2: Description of features in basic dataset

Feature	Type	Description
Day	Integer	Number of the therapy day for a patient
Weekday i	Boolean	Denotes if the current day is weekday i $(i \in \{1, 2, ..., 7\})$
Rating i	Float	Average daily rating of a patient for question i $(i \in \{1, 2, ..., 7\})$
Country i	Boolean	Denotes if the data is recorded in country i $(i \in \{1, 3, 4, 6, 8\})$
Gender	Integer	Gender of the patient Male $(i = 1)$, female $(i = 2)$
Status i	Boolean	Denotes if marital status of patient is i $(i \in \{1, 2, ..., 5\})$
Education i	Boolean	Denotes if education of patient is i $(i \in \{1, 2, 3\})$

Next, an input dataframe has to be created for the models to be trained on. This dataframe should consist of sequences of dates with information per patient. A row consists of a patient id and date followed by all the features in the files. The list of features is shown in Table 2. For the rating, country, status and education features and their corresponding values, see Appendix A.1 to A.4.

In order to see if the message and therapy data improve the performance of the models, we create a second dataset including all data available, shown in Table 3. The features are calculated on a daily basis and as a running total. The messaging features, with prefix message and char, are calculated separately for *sent by the patient* and *received by the patient*. This means that these features are calculated twice: once for the messages from patient to therapist, once for the other way around.

Table 3: Description of features additionally in complete dataset

Feature	Type	Description
Pages total day	Integer	Daily total number of pages read
Pages total		Total number of pages read
Time total day	Integer	Daily total time spent by reading
Time total		Total time spent by reading
Time average day	Float	Daily average time per page spent reading
Time average		Average time per page spent reading
Message total day	Integer	Daily total number of messages sent/received
Message total		Total number of messages sent/received
Char total day	Integer	Daily total number of characters sent/received
Char total		Total number of characters sent/received
Char average day	Float	Daily average number of characters per message sent/received
Char average		Average number of characters per message sent/received

The therapy features (top half in Table 3) are provide information about the quantity and quality of the therapy sessions of the patients. We hypothesize that when the amount of pages read or time spent reading increases, the patient's mood would increase. This also count for the running total: we expect that when a patient has progressed through more therapy modules, the average mood of the patient increases, as that would be the intended purpose of the modules.

The messaging features (bottom half in Table 3 summarize the amount of contact that has taken place between the patient and their therapist via text messaging. We would expect that the mood of a patient should increase when he/she has had contact with their therapist, or atleast that the decline in mood is dampened a bit. This would be a result of counseling or guidance provided by the therapist. Longer messages would probably contain a more meaningful conversation, which we expect would have more influence on the mood of the patient.

4.4 Missing value imputation

Due to the transformation from a long format to multiple sequences, a lot of gaps in the data are created. These can result from, for example, dates on which a patient does not fill in his mood, not message their therapist or not read one of the modules. Filling in this missing information is done differently per feature. The descriptive statistics for the behavior of rating mood contain no missing values, as all these statistics are calculated on mood. If a patient has not rated mood at all, he or she we will not be present in te dataset in the first place. For the basic features in Table 1, no imputation is needed as there are no missing values. The missing values in the *ratings i* have been filled as follows: if the gap is at most three days, the values are interpolated linearly. If the gap is larger, the values are filled with the mean over the originally computed daily averages. This follows the same protocol as stated in the paper from Mikus et. al [8]. The missing values in the daily features included in the complete set are all imputed with 0, as an absence of these values means an absence of that event. The running totals contain no missing values, as they are simple cumulative sums over the daily features.

4.5 Patient clustering

The clustering of the patients is done by agglomerative hierarchical clustering, using *Ward's criterion* as linkage function with the *Euclidean distance*. This is a result from comparing agglomerative clustering and k-means clustering, as discussed in Section 3.2. Both methods produce similar results with a slight advantage to hierarchical clustering, which led us to prefer agglomerative clustering, as the results of this method are also deterministic, whereas the results from k-means clustering can vary. The features used to perform the clustering are listed in Table 1.

The goal is to create several clusters of patients that exhibit similar rating behavior while maintaining enough patients per cluster to have a sizable training and test set. This resulted in 13 clusters, with most clusters containing 10 to 20 patients, the smallest cluster containing 5 and the largest cluster containing 41. This is based on visual inspection of the dendrograms, which are shown and analyzed in Section 6.1, and balancing the distribution of patients per cluster.

4.6 Analysis

Before we start with the results, we will take a look at the data. Specifically, we will analyze the correlation between the features and the target variables and show a sample sequence from a patient. First, let us look at the correlation plots. These are computed using Pearson's R between the target variable, tomorrow's mood, and the features, per cluster of patients. The correlation heatmap resulting from agglomerative clustering is shown in Figure 4. For the features 'rating_1' to 'rating_7', a table with the corresponding questions is given in Appendix A.1. The question regarding mood is 'rating_2'.

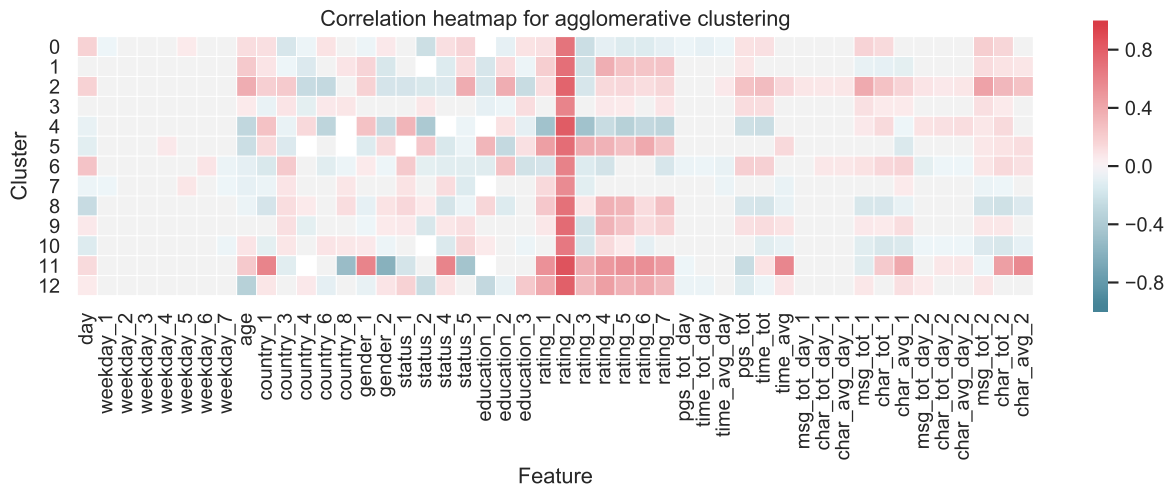


Figure 4: Correlation coefficients for features versus y, per cluster

Figure 4 shows that the highest correlation can be found between the mood of today (*rating 2*) and the mood of tomorrow. The figure also shows that there is a significant correlation between the other 6 ratings and the mood of tomorrow. There is also significant correlation between the long-term messaging and therapy features and the target variable, whereas the daily features show little to no correlation. This could be an indication that therapy does have an impact in the long run, but less in the short-term. In contrast of this is the low correlation between the day of the week and the mood rating of tomorrow. Several papers examined in Section 2 showed performance increases after adding the day of the week as variable. These coefficients do not prove that they will have low impact on performance as only direct correlation is measured. The general information about patients also seems to correlate with the target variable.

5 Experimental Setup

This section will cover the creation of the benchmark and the experimental setups for the models. A concise overview of the choice of benchmark and an overview of the different parameter settings for the models will be shown. The results of these setups will be discussed in Section 6.

5.1 Benchmark

In order to provide a good comparison between the performance of the models, a benchmark is created. For this purpose, we create a simple persistence model, where the forecast for the next timestep is equal to the previous observed value. We believe this is a better choice than predicting the mean, as there is a lot of fluctuation and variance in the ratings of mood. Predicting the mean would result in a error that is significantly higher than the persistence model.

5.2 Clustering setups

Clustering the patients is done using two methods: hierarchical agglomerative clustering and k-means clustering. For hierarchical clustering, the optimal amount of clusters k will be chosen based on the amount and size of clusters using the resulting dendrogram. Since this clustering method is deterministic, there is no need for repeated experiments. In contrast to that is k-means clustering which is non-deterministic. For this method, several values for k will be tested in repeated experiments to create confidence intervals for the resulting distortion metric. For both clustering methods, a correlation heatmap between the features and the mood rating will be created.

5.3 Model setups

Table 4: List of parameter values to be explored

Parameter	Values
Neurons	[5, 10, 25, 50]
Epochs	[10, 50, 100, 250]
Batch size	[2, 4, 8, 16]
Timesteps	[1, 3, 7]

For all three model variants, we will use the same test setup, where we create a single-layer model and train one variant per cluster. The architecture is a result of the literature review in which no significant performance gain was found when increasing the amount of layers in the network. We will vary the amount of neurons per layer between 5 and 50, vary the amount of epochs between 10 and 250, vary the batch size between 2 and 16 and vary the amount of historical timesteps between 1 and 7 (days). Each model setup will be run 5 times, as all three models are of a stochastic nature. From these 5 runs, we will average the obtained error and calculate the standard deviation among these errors. For a full list of values of the parameters that will be explored, see Table 4.

For each cluster, we reserve the first 50% of a patient's data for training and the last 50% of the data for testing. A training set for a specific cluster will therefore be multiple sequences originating from different patients. In order not to mix the historical data of these patients, we will reset the hidden stats of the models between different sequences.

6 Results

This section will show the results of the clustering methods, the benchmark and the models. We will start by showing the clustering results of the agglomerative clustering and k-means clustering methods and compare their effectiveness using the correlations between the features and the target variable. We will then show the results of the benchmark before comparing the different models on their performance and on the different datasets.

6.1 Clustering results

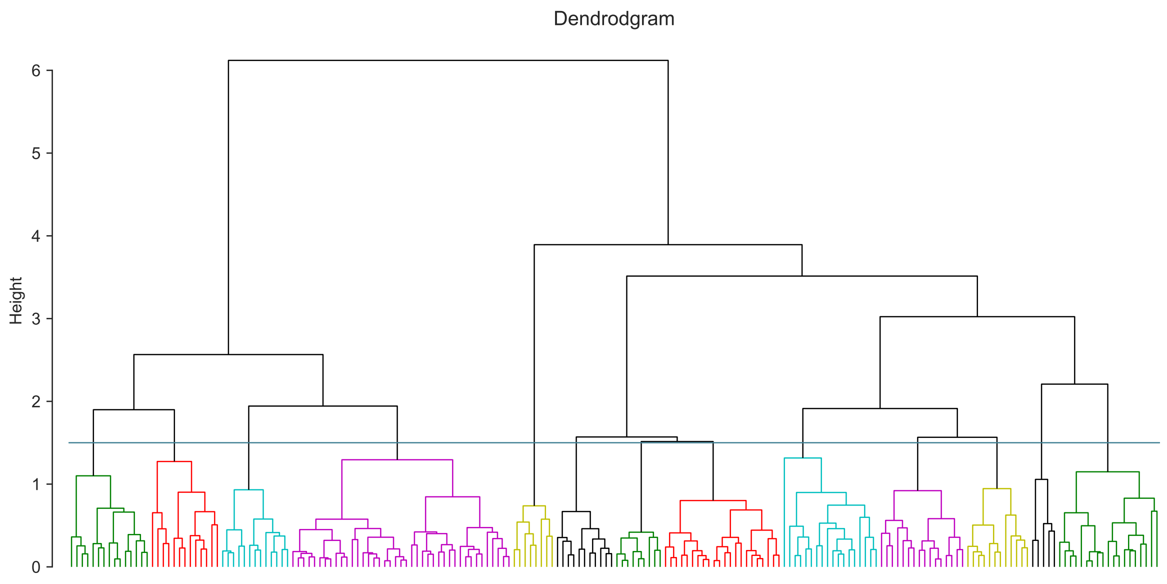


Figure 5: Dendrogram for agglomerative clustering

Figure 5 shows the dendrogram created using the hierarchical agglomerative clustering method. The different colors for the leaves and edges show the distinct clusters. The height of the edges show the relative distance expressed as Ward's criterion. The horizontal line roughly shows the cutoff value chosen, marking the maximum distance for which clusters can be merged. This value is determined based on the visual inspection of the dendrogram: the resulting clusters contain a relatively even number of patients, while keeping the relative distance in the clusters also relatively close. Setting the threshold higher would result in having two extra relatively larger clusters, setting it lower would disperse a lot of clusters into many smaller ones. We determined that this value would balance the amount of clusters and the amount of variance per cluster the best.

In order to get the value for k for the k-means clustering approach, we use the elbow method, shown in Figure 6. This figure is the result of running 5 iterations of the same k value and taking its mean distortion value. Unfortunately, we see that there is no 'hard' elbow in the graph. We do see a small stagnation of the error at $k = 13$, which is the value of k we choose.

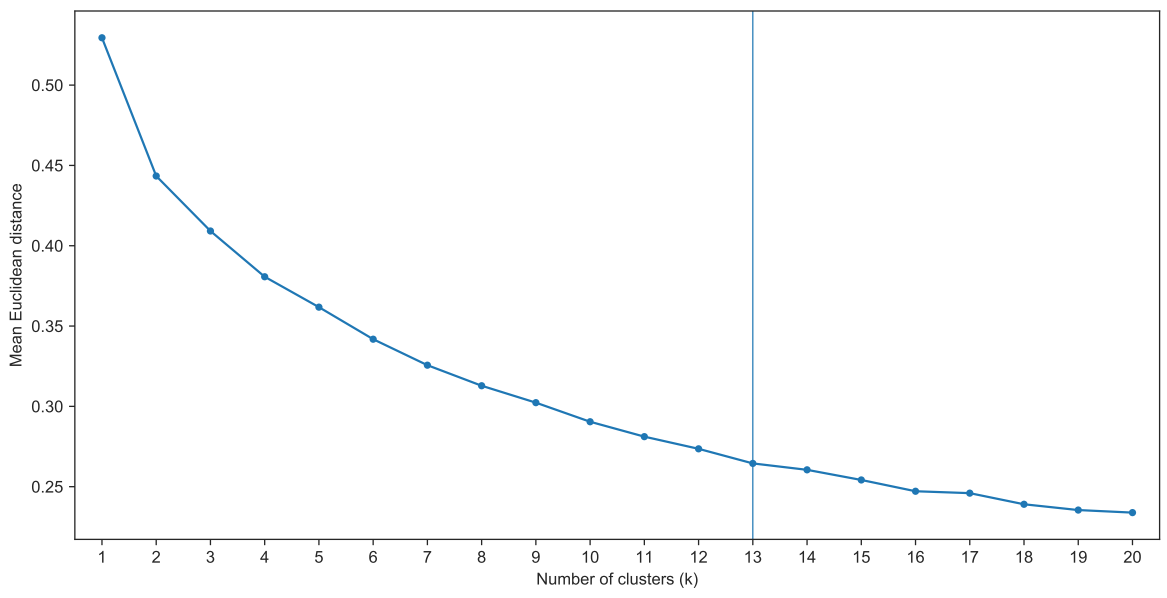


Figure 6: Elbow graph for k-means clustering

The corresponding correlation heatmap between the features and target variables has already been shown in Section 4.6, but we will show it here again for better comparison between k-means and agglomerative clustering, in Figure 7 and 8. It is clear that the strength of the correlations in both heatmaps is about the same for the same features, with the correlations for hierarchical agglomerative clustering being slightly stronger.

Table 5 shows the size of the clusters per clustering method. As can be seen, most clusters for the hierarchical method contain between 8 and 22 patients, with one cluster containing just 5 patients and one cluster containing 41. The clusters sizes resulting from the k-means method are almost as evenly spaced, but with more small clusters and no extra large cluster.

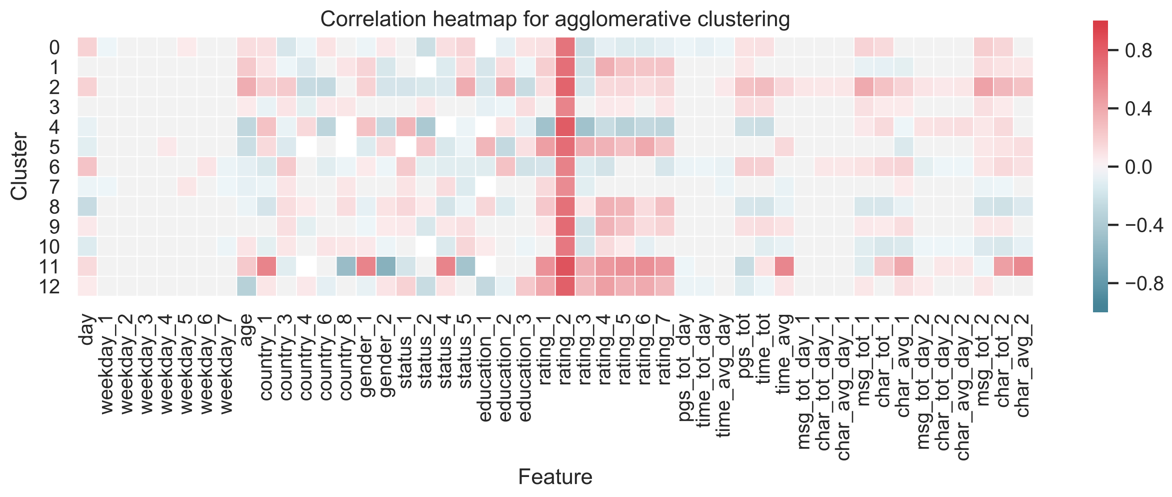


Figure 7: Correlation coefficients for features versus y, per cluster, agglomerative clustering

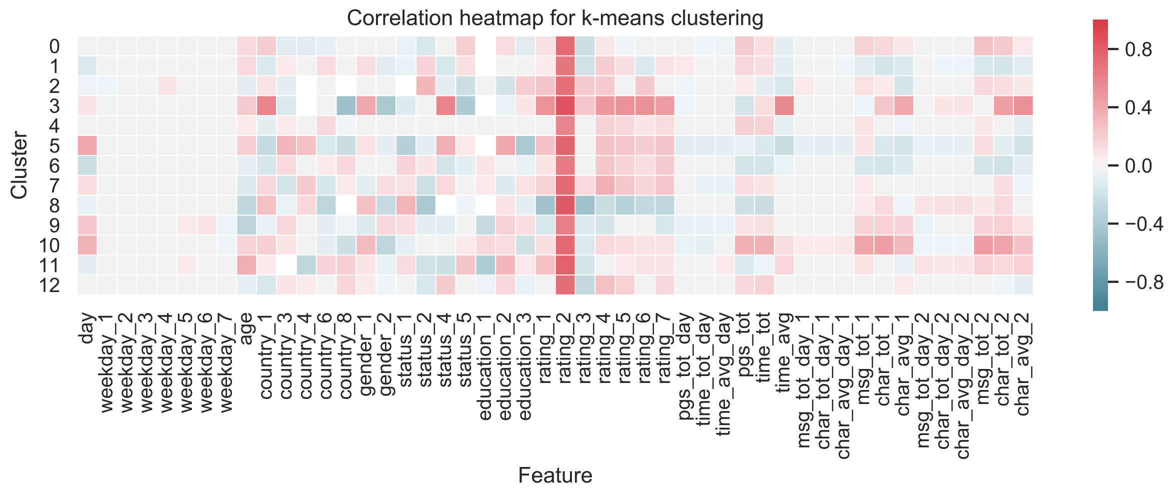


Figure 8: Correlation coefficients for features versus y, per cluster, kmeans clustering

Table 5: Cluster sizes of resulting from agglomerative and k-means clustering

Cluster	1	2	3	4	5	6	7	8	9	10	11	12	13
Size K-means	4	5	6	13	14	14	16	17	17	20	24	24	28
Size Agglomerative	5	8	9	11	12	13	13	15	16	18	19	22	41

6.2 Model results

We start by presenting the results from the benchmark model, the persistence model. This model achieved an RMSE of 0.0978. Next, the results from the model setups are shown in Figure 6, 7, 8. None of the model setups lead to an increased performance compared to the persistence model.

The best test results for all three model variants are: RNN 0.120 (0.003), GRU 0.113 (0.005), LSTM 0.112 (0.002). If we perform t-tests to inspect the difference in performance, we see that only the RNN is statistically worse than both the GRU and LSTM ($p = 0.0277, p = 0.0011$) at the 95% confidence interval.

There is no significant difference in the performance of the GRU and LSTM ($p = 0.689$). Both the GRU and LSTM did not benefit from adding extra timesteps of historical data, achieving their best performance for $t = 1$. The RNN performed consistently worse on the complete dataset than on the base dataset. For the GRU and LSTM this differs between the amount of timesteps t, but showing no consistent improvement. Interesting to see is that for several model setups the test performance is higher than the train performance. After investigating the train and test dataset, we found that test set contained a significant amount more imputed mean values than the training set (43%).

When we inspect the effect of increasing the amount of epochs on the train and test error, we see that the models rapidly start overfitting: after a mere 25 epochs, the training error decreases steadily while the test error increases. A figure of this can be seen in Appendix B, displaying the errors for the three best model setups.

Table 6: RMSE(SD) of the models trained on the base and complete set, 1 timestep,

Model	Base		Complete	
	Train	Test	Train	Test
RNN	0.127 (0.005)	0.120 (0.006)	0.126 (0.020)	0.134 (0.020)
GRU	0.123 (0.006)	**0.113 (0.005)**	0.118 (0.001)	0.116 (0.004)
LSTM	0.124 (0.011)	0.115 (0.013)	0.116 (0.001)	**0.112 (0.002)**

Table 7: RMSE(SD) of the models trained on the base and complete set, 3 timesteps

Model	Base		Complete	
	Train	Test	Train	Test
RNN	0.124 (0.003)	0.122 (0.006)	0.119 (0.003)	0.128 (0.006)
GRU	0.129 (0.013)	0.120 (0.013)	0.118 (0.003)	0.116 (0.003)
LSTM	0.118 (0.004)	0.110 (0.004)	0.116 (0.004)	0.114 (0.005)

Table 8: RMSE(SD) of the models trained on the base and complete set, 7 timesteps

Model	Base		Complete	
	Train	Test	Train	Test
RNN	0.121 (0.003)	**0.120 (0.003)**	0.118 (0.005)	0.122 (0.005)
GRU	0.125 (0.014)	0.121 (0.016)	0.128 (0.018)	0.132 (0.020)
LSTM	0.138 (0.026)	0.136 (0.028)	0.113 (0.003)	0.115 (0.002)

7 Conclusions

From the results, it is clear that the model setups did not perform adequately. None of the models succeeded in outperforming the baseline. Next, we do not observe a consistent performance increase when we extend the basic dataset with messaging and therapy data. Clear from the results is that the GRU and LSTM perform the same in our scenarios, with the basic RNN performance being significantly worse. The worse performance cannot be attributed to the vanishing gradient problem, since the basic RNN performs best with 7 timesteps, whereas both the GRU and LSTM perform best with 1 timestep.

The low performance obtained by all the model setups could have several causes. First, it seems that the single-layer architectures are not appropriate for this dataset. From the wide grid-search performed, even the best performing models started overfitting with few epochs of training, showing significantly worse performance on the test set than the baseline. Second, the handmade features from the messaging and therapy data could carry no predictive power, although the correlation heatmap showed significant direct correlations between the features and target variable, making this less likely. Another reason for inadequate performance of the models could be the clustering performed prior to the modeling. It could be that the clusters obtained do not represent the underlying clusters in the patients, or perhaps, that these clusters do not exist at all. Lastly, due to many missing values in the data, it could be that imputation transformed the dataset in a way that it ceased reflecting the actual mood of the patients.

For future research, it would be interesting to create other handmade features that could have more predictive power. It would also be worthwhile to increase the complexity of the models, such as more recurrent layers or adding dense fully-connected layers, moving more towards Deep Learning. Finally, imputing the mean for missing values could perhaps be replaced by a more sophisticated imputing process, possibly avoiding the test error being smaller than the training error.

Appendices

A Feature legends

A.1 Rating features

The rating features with corresponding EMA question

Feature	Value
rating_1	How well did you sleep last night?
rating_2	How is your mood right now?
rating_3	How much do you worry at the moment?
rating_4	How do you feel about yourself right now?
rating_5	How much did you enjoy activities today?
rating_6	How much were you involved in social interactions today?
rating_7	To what extend did you accomplish pleasant activities today?

A.2 Country features

The country features with corresponding country

Feature	Value
country_1	Poland?
country_3	United Kingdom
country_4	Germany
country_6	The Netherlands
country_8	Switzerland

A.3 Status features

The status features with marital status

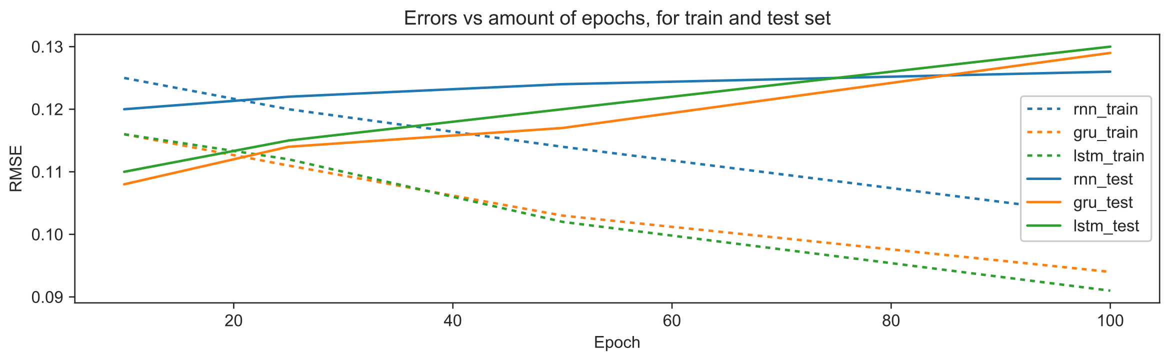
Feature	Value
status_1	Single
status_2	Divorced
status_3	Widowed
status_4	Living together
status_5	Married

A.4 Education features

The education features with corresponding education level

Feature	Value
education_1	Low
education_2	Middle
education_3	high

B Errors vs epochs



References

[1] *Depression and Other Common Mental Disorders: Global Health Estimates*. Geneva: World Health Organization, 2017 (cit. on p. 1).

[2] Pim Cuijpers and Filip Smit. “Excess mortality in depression: a meta-analysis of community studies”. In: *Journal of Affective Disorders* 72 (3 2002), pp. 227–236 (cit. on p. 1).

[3] Patrik Sobocki et al. “Cost of Depression in Europe”. In: *Journal of Mental Health Policy and Economics* 9 (2 2006), pp. 87–98 (cit. on p. 1).

[4] John Pastor and Ward van Breda. “Analyzing and Predicting Mood of Depression Patients”. In: (2015) (cit. on pp. 1, 4).

[5] Abhinav Mehrotra, Robert Hendley, and Mirco Musolesi. “Towards Multi-modal Anticipatory Monitoring of Depressive States Through the Analysis of Human-smartphone Interaction”. In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. UbiComp ’16. ACM, 2016, pp. 1132–1138 (cit. on pp. 1, 4).

[6] Yoshihiko Suhara, Yinzhan Xu, and Alex ’Sandy’ Pentland. “DeepMood: Forecasting Depressed Mood Based on Self-Reported Histories via Recurrent Neural Networks”. In: *Proceedings of the 26th International Conference on World Wide Web*. WWW ’17. International World Wide Web Conferences Steering Committee, 2017, pp. 715–724 (cit. on pp. 1, 4).

[7] Natasha Jaques et al. “Predicting Tomorrow’s Mood, Health and Stress Level using Personalized Multitask Learning and Domain Adaptation”. In: (2017) (cit. on pp. 1, 5).

[8] Adam Mikus et al. “Predicting short term mood developments among depressed patients using adherence and ecological momentary assesment data”. In: *Internet Interventions* 12 (1 2018), pp. 105–110 (cit. on pp. 1, 5, 16).

[9] Rafal Weron. “Electricty price forecasting: A review of the state-of-the-art with a look into the future”. In: *International Journal of Forecasting* 30 (2014), pp. 1030–1081 (cit. on p. 3).

[10] Tara N. Sainath et al. “Convolutional, Long-Short Term Memory, fully connected Deep Neural Networks”. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015, pp. 4580–4584 (cit. on p. 3).

[11] Zachary Chase Lipton et al. “Learning to Diagnose with LSTM Recurrent Neural Networks”. In: *CoRR* abs/1511.03677 (2015) (cit. on p. 3).

[12] Sebastian Raschka. *Why do we need to re-use training parameters to transform test data?* Dec. 14, 2018. URL: https://sebastianraschka.com/faq/docs/scale-training-test.html#why-do-we-need-to-re-use-training-parameters-to-transform-test-data (cit. on p. 7).

[13] Pradeep Rai and Shubha Singh. “A Survery of Clustering Techniques”. In: *International Journal of Computer Applications* 7 (12 2010), pp. 1–5 (cit. on p. 8).

[14] Pavel Berkhin. “A Survey of Clustering Data Mining Techniques”. In: *Grouping Multidminesional Data: Recent Advances in Clustering*. Ed. by Jacob Kogan, Charles Nicholas, and Marc Teboulle. 2006, pp. 25–71 (cit. on p. 8).

[15] Fernando J. Pineda. “Generalization of Back-Propagation to Recurrent Neural Networks”. In: *Physical Review Letters* 59 (19 1987), p. 2229 (cit. on p. 9).

[16] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. In: *CoRR* (2014) (cit. on p. 9).

[17] Sepp Hochretier and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9 (8 1997), pp. 1735–1780 (cit. on p. 9).

[18] E-COMPARED. *European Comparative Effectiveness Research on Internet-based Depression Treatment*. Dec. 14, 2018. URL: https://www.e-compared.eu (cit. on p. 13).