Machine Learning Methods for Mood Prediction Using Data from Smartphones and Wearables

Mengjie Shi
Supervisor: Ambuj Tewari

Department of Statistics
University of Michigan, Ann Arbor

June 15, 2019

1 Abstract

Medical interns tend to have a higher rate of depression than the general population due to factors such as high workload, irregular lifestyle, insufficient sleep, and lack of physical activity. We used mobile health data collected from medical interns and used it to forecast their mood with the ultimate goal of tailoring interventions to improve mental health. Our data sets come from the Intern Health Study (IHS) and include data on mood, physical activity, sleep, heart rate, PHQ-9 responses, geo-location, and baseline information. This time series data was collected from medical interns throughout their entire internship period. Existing studies on mood prediction in mobile health mostly classify an individual's mood using a discrete scale, such as high mood vs. low mood, by training on the same individual's data. In contrast, we predicted mood on a continuous scale. We also predicted mood by training on the same individual's data as well as predicted mood of a new individual by training on other individuals' data. We used linear methods and nonlinear methods, including lasso and neural nets, and compared the performance of these methods. There are signs that a high degree of missingness affects prediction capability and evaluation of prediction capability. We did not see any benefits of using nonlinear models so far, but this may be due to the high missing rate in our data.

2 Introduction

Depression is a pervasive mental disease, with more than 300 million people suffering from it [1]. Sleep deprivation and physical inactivity are important factors for major depression [2] [3]. Self-reported happiness is a primary interest of outcome in research on mental health

[4]. Thus, it is beneficial to be able to predict self-reported moods using sleep, physical activity and other data collected via smartphones and wearable devices. Accurate mood predictions can be used to offer interventions in order to prevent the onset of depression. Several advances in technology and statistical methodology have made the progress in mood prediction possible.

With smartphone apps and wearable sensors, a large amount of data can be collected passively in real time. Various health related quantities such as daily activity, sleep, and heart rate are recorded unobtrusively, providing more available data for analysis. These passively collected data, in addition to self-reported mood scores, give rise to a time-series data set containing a large count of frequently collected variables for each individual.

There are several elements involved in the prediction pipeline. Here we are going to talk about prediction problems, prediction algorithms, multitask learning, missing value imputation and cross-validation for time series.

The prediction problem most studies focus on is mood prediction at the individual level. Also, the prediction problem is to classify future mood using a discrete scale. The algorithms include both linear and nonlinear ones. Works from Rosalind Picard's lab at MIT split mood into binary classification labels (high/low), and applied classification algorithms - neural nets, logistic regression and SVM with modifications on the kernel function [5] [6]. In another paper, mood was classified into four categories, and was predicted on individual's history with behavioural and sleep data using long short-term memory (LSTM) neural network models [7]. Their results showed that the prediction capability of LSTM outperforms SVM, and LSTM is able to extract useful information on lag importance. A later work by Picard showed that LSTM outperforms logistic regression and SVM on classifying mood based on features including sleep, activity and mobility [5]. There is one work that applied regression to predict mood with smart phone sensor logs, which included screen time, phone camera events, etc [8]. However, they found that the forward stepwise regression, which had positive results before, is inferior to the mean prediction or the lag-2 prediction on their data set.

Noticeably, one work from Picard's lab used multitask learning, which allows models to learn different tasks at the same time while still sharing information through similarity constraints [6]. Specifically, they clustered users based on their personality and gender by K-means, and then treated prediction for one given cluster as one task. The models include sharing information across all clusters and unique training on each cluster. In addition to clustering baseline information, another method could be to cluster on regression coefficients. In functional data applications, where each data point is a curve with regression coefficient estimates, K-means could be applied to cluster the data points by plugging in the regression coefficient estimates [9]. In IHS, we could first fit a linear model, and then extract the coefficient estimates as data points used in K-means clustering.

Missing value imputation is vital to mood prediction when the degree of missingness is high. The above study on regression analysis mentioned that individuals with a high degree of missingness were excluded from the analysis [8]. On the other hand, Picard's lab proposed an imputation algorithm to incorporate deep learning called the Multimodal Autoencoder (MMAE) [10]. The study showed that MMAE is more accurate in reconstructing data from

a missing module than discarding missing values, mean imputation or PCA reconstruction method, which applies the inverse transformation learned by PCA.

There are some limitations regarding the currently existing studies. First, according to the discussions on cross-validation, vanilla cross-validation was frequently applied to evaluate models on time-series data [6, 8]. It could be problematic in time-series data because a random split of the data set does not preserve the time-ordering quality. In IHS, we applied block cross-validation, which splits the train and test data sequentially. Additionally, studies so far focuses on prediction at the individual level with few positive results on regression analysis. For IHS, we would apply regression on not only prediction at the individual level, but also prediction on a new individual's mood by training other individuals' data.

3 Data

Our motivating data sets are from the Intern Health Study (IHS) [11]. The study seeks to predict moods of the 2015-2016 cohort of medical interns from the University of Michigan and of the 2017-2018 cohort of medical interns from multiple institutions by following each cohort through their entire internship year. Medical interns have increasing possibility of depression because of high workload, irregular lifestyle, insufficient sleep time, and lack of physical activity [12].

3.1 2015-2016 Data

The study includes 23 medical interns from the 2015-2016 cohort. Usually, the start date of internship is 7/1. The study started to collect data from 3/31/2015 and ended on 7/1/2016. Data on mood, activity, sleep, heart rate, responses to the PHQ-9 questionnaire and a survey, and baseline information were collected for every intern. Mood was recorded daily on a 1–10 scale by an app, where 10 means the best mood. In addition, physical activity and sleep data were recorded every minute by wristbands or apps, and heart rate was recorded every second. A Personal Health Questionnaire (PHQ-9) and a survey on sleep/working hours and chronic depression were delivered once every three months for a year. Baseline information (e.g., pre-intern PHQ-9 score, demographic information, medical specialty) was collected before 7/1/2015.

After cleaning and aggregating the data to get daily level information, we picked 11 variables summarizing data about mood, sleep, and physical activity. Since there is a considerable amount of missing values in our data set, we added "missing" variables which are indicators of whether data for a specific variable was missing on day t. See Table 1 for more details.

The cleaned-up data composed of the variables in Table 1 were used to train models. There are three noticeable characteristics of the cleaned data. 1) The rate of missingness is high. There are up to 72% of mood data, 95% of sleep data, and 81% of activity data missing at the individual level. The serious missingness led to the selection and evaluation of imputation methods. 2) From Table 1, the ranges of variables vary significantly, so standardization at

the individual level was applied on sleep and distance variables. 3) Since this data is a time series, when fitting models, we added a trend term and treated each day as a lag. Because mood, sleep, and activity data were collected at various times during a day, we need to find out where we should start as lag 1. On day t, mood was collected at 8 p.m. Sleep data was based on any sleep period that ended on day t, even if the subject fell asleep on day t-1. Activity data was between 12 a.m. of day t and 12 a.m. of day t+1. Thus, if we were to predict mood on day t, we would treat mood on day t-1, and sleep and activity on day t as lag 1.

variables	Type	Values	Range
Mood	Ordinal	1,2...,10	1-10
Total Minutes Asleep	Count	0,1,...	0-914
Total Time in Bed (in mins)			10-969
Total Distance	Continuous	Positive Real	0.01-31.10
Very Active Distance			0.00-21.25
Moderately Active Distance			0.00-13.77
Light Active Distance			0.00-13.78
Mood Missing	Binary	0,1	0,1
Sleep Missing			
Activity Missing			
Total Sleep Records	Count	0,1,2,3,...	1-5

Table 1: Description of variables of the 2015-2016 cohort.

3.2 2017-2018 Data

From the 2017-2018 cohort, data on similar modules were collected for 536 interns starting from 4/1/2017. The internship period was 7/1/2017-6/30/2018. The 2017-2018 data is also a time series with even higher missingness. There are up to 100% missingness in mood, sleep and activity data during internship at the individual level. 26 interns who have no mood during internship were removed from the entire analysis. 3 more interns were removed from analysis including baseline variables because their baseline variables are missing entirely.

There were some changes on data collection and calculation. First, we additionally collected geo-location data but incorporating it in the prediction task is left for future work. Also, activity data were calculated in terms of step counts and active minutes instead of distances. Lastly, sleep data were calculated to get information for different sleep stages. We picked 13 sleep and activity variables. In addition, we included heart rate data and 21 baseline variables in analysis. See Table 2 for details on variables used in prediction.

Variables	Type	Values	Range
Mood	Ordinal	1,2...,10	1-10
Inbed (in mins)	Count	0,1,...	1-1400
Restless (in mins)			0-534
Deep Minutes			0-210
Rem Minutes			0-310
Light Minutes			58-943
Wake Minutes			9-228
Wake Count			3-93
Light Count			3-94
Deep Count			0-13
Rem Count			0-30
Total Minutes			58-1116
Daily Step Total	Count	0,1,...	0-327365
Active Minutes			0-868
Resting Heart Rate	Count	0,1,...	39-96
Baseline Variables	Various	Demographic Information Medical Specialty Survey Scores PHQ-9 Scores Recent Work/Sleep Hours	Various

Table 2: Description of variables of the 2017-2018 cohort.

4 Scientific Questions

Mood prediction for an individual who has been observed for a period of time can help inform individualized depression intervention timing.

In this study, there are two prediction tasks that we will focus on. 1) Predicting an individual's future mood by training that same individual's previous data (T+1). 2) Predicting a new individual's mood by training other individuals' data (N+1).

$X_{i,t}$ is the predictors on day t for individual i.

$\hat{y}_{i,t}$ is the predicted mood on day t for individual i .

$y_{i,t}$ is the outcome (mood) on day t for individual i.

$\bar{y}_i$ is average mood of outcomes $y_{i,t}$ for individual i.

Prediction Problem 1: Predicting the Same User's Mood (T+1):

For each user $i \in$ data set:

For each day $t \in training$:

Train a model f_i such that $f_i(X_{i,t}) = \hat{y}_{i,t}$

For each day $t \in test$:

Calculate the $mse_{i,t} = (f_i(X_{i,t}) - y_{i,t})^2$

Calculate $R_i^2 = 1 - \frac{\sum_{t \in test} mse_{i,t}}{\sum_{t \in test} (y_{i,t} - \bar{y}_i)^2}$

Prediction Problem 2: Predicting a New User's Mood (N+1):

For each user $i \in training$:

For each day $t \in [1, 365]$:

Train a model f such that $f(X_{i,t}) = \hat{y}_{i,t}$

For each user $j \in test$:

For each day $t \in [1, 365]$:

Calculate the $mse_{j,t} = (f(X_{j,t}) - y_{j,t})^2$

Calculate $R_j^2 = 1 - \frac{\sum_{t=1}^{365} mse_{j,t}}{\sum_{t=1}^{365} (y_{j,t} - \bar{y}_j)^2}$

5 Methods

The pipeline of solving prediction problems can be constructed from several elements. In IHS, the elements include imputation methods in training and test data sets, cross-validation methods, the number of lags, and the prediction algorithms. See Table 3 for details.

Given the high degree of missingness in our data sets, the selection of imputation method is vital to the prediction effectiveness and the evaluation of the prediction effectiveness. In IHS, we applied two imputation methods - mean imputation and carry-forward imputation. Both will be explained in later paragraphs. In the training process, we always used the complete training data. In the test process, we could either test on the complete test data where some moods y were imputed (test on imputed), or on a reduced complete test data where data points with mood y missing originally were excluded (test on observed).

Some elements in the pipeline relate to time series. We selected and tuned the number of lags d to include as predictors. Also, in addition to vanilla cross-validation, we applied block cross-validation which takes the time-ordering between time series data points into consideration. Block cross-validation will be explained in a later section.

The algorithms include linear and nonlinear ones. In terms of linear algorithms, we started with lasso. Then we added multitask learning to lasso in setting 1 by clustering lasso coefficients. Another direction is nonlinear methods because they capture a wider range of relationships between mood and other variables. Vanilla neural networks were a nonlinear method to start with. Then recurrent neural networks were applied to the time series data in this study, which aims to capture the temporal behaviors in the data [13].

	T+1		N+1
	Setting 1	Setting 2	Setting 3
Imputation	Mean Imputation	Carry-forward Imputation	Carry-forward Imputation
Cross-Validation	5-fold Vanilla CV (RNN uses 1-fold block cv)	5-fold Block CV	10-fold Vanilla CV
Lags	1,3,5,10,15	7	7
Test	on imputed	on observed	on observed
Algorithms	Lasso(no baseline) Lasso with Multitask Learning(no baseline) Vanilla NN RNN	Lasso(no baseline) RNN	Lasso(no baseline) Lasso(with baseline)
Data Sets	2015	2015 2017	2017

Table 3: Summary of methods.

5.1 Missing Value Imputation

5.1.1 Mean Imputation

In this method, the mean of all observed values within the variable x is used to impute the missing data entries in x at the individual level.

For individial i:

$$\text{All missing entries} = \frac{1}{\sum_{t=1}^{365} I(x_{i,t}^O)} \sum_{t=1}^{365} x_{i,t} \times I(x_{i,t}^O) \,,$$

$$where\ I(x_{i,t}^O) = \begin{cases} 1, & if\ x_{i,t}\ is\ observed \\ 0, & if\ x_{i,t}\ is\ missing \end{cases}$$

5.1.2 Carry Forward Imputation

Another imputation method we tried is carry forward imputation, which uses an aggregation of previous values in variable x to impute the missing entry on day t.

Specifically, if x is the mood variable, we used yesterday's mood to impute today's mood.

For individual i:

If $I(x_{i,1}^O) = 0$:

$$x_{i,1} = \frac{1}{\sum_{t=1}^{365} I(x_{i,t}^O)} \sum_{t=1}^{365} x_{i,t} \times I(x_{i,t}^O)$$

For t from 2 to 365:

If $I(x_{i,t}^O) = 0$:

$$x_{i,t} = x_{i,t-1}$$

If x is one of the non-mood variables, we used the mean of the previous 7 days to impute. When there are up to 100% missingness in x, we borrowed information across individuals to impute.

For individual i:

If $\sum_{t=1}^{365} I(x_{i,t}^O) = 365$:

All entries $= \frac{1}{\sum_{j \in users} \sum_{t=1}^{365} I(x_{j,t}^O)} \sum_{j \in users} \sum_{t=1}^{365} x_{j,t} \times I(x_{j,t}^O)$

else:

For t from 1 to 7:

If $I(x_{i,t}^O) = 0$:

$x_{i,t} = \frac{1}{\sum_{t'=1}^{365} I(x_{i,t'}^O)} \sum_{t'=1}^{365} x_{i,t'} \times I(x_{i,t'}^O)$

For t from 8 to 365:

If $I(x_{i,t}^O) = 0$:

$x_{i,t} = \frac{1}{7} \sum_{d=1}^{7} x_{i,t-d}$

Here the set of *users* depends on the prediction problem. For T+1, the set is all users. For N+1, the set corresponds to whether individual i belong to the training set or the test set.

5.2 Block Cross-Validation

Vanilla k-fold cross-validation is a technique to assess test error by randomly splitting the data set into k folds, and holding out one fold of observations when training the model[14]. However, when it comes to time series, vanilla cross-validation can be problematic because if the data points are randomly split, we may train on future data points and test on past data points. Instead, we split the data set sequentially into k equal folds to preserve time-ordering, train on the previous folds and test on the next. See Figure 1 [15].

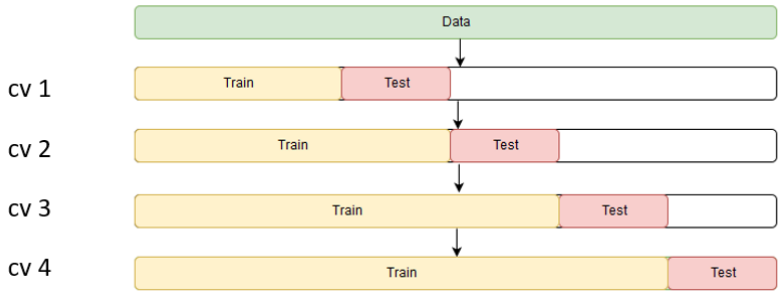

Figure 1: Overview of block cross-validation.

5.3 Prediction Algorithms

5.3.1 Lasso

Lasso is a linear model that can shrink the coefficient estimates towards zero by putting constraints on the sum of their absolute values, i.e., the L_1 norm of the coefficient vector [16, p. 69]. It can help pick out significant predictors. Given our data, if we were to predict mood with d lags, the lasso coefficients $\hat{\beta}_\lambda^L$ minimizes the quantity in:

T+1:

For each individual i,

$$\sum_{t \in training} \left(y_{i,t} - \beta_{i,0} - \beta_{i,d*p+1} t - \sum_{j=1}^{d*p} \beta_{i,j} x_{i,t,j} \right)^2 + \lambda \left(|\beta_{i,d*p+1}| + \sum_{j=1}^{d*p} |\beta_{i,j}| \right)$$

where t is day, i is individual, p is the number of predictors, d is the number of lags, $y_{i,t}$ is mood of individual i on day t, and $x_{i,t,j}$ is the j-th predictor of individual i on day t. $\lambda \geq 0$ is a tuning parameter that decides how much to shrink: the larger λ is, the greater the shrinkage is [16, p. 69].

N+1 (no baseline):

$$\sum_{i \in training} \sum_{t=d+1}^{365} \left(y_{i,t} - \beta_0 - \beta_{d*p+1} t - \sum_{j=1}^{d*p} \beta_j x_{i,t,j} \right)^2 + \lambda \left(|\beta_{d*p+1}| + \sum_{j=1}^{d*p} |\beta_j| \right)$$

Note that here all users are split into a training set and a test set.

N+1 (baseline):

$$\sum_{i \in training} \sum_{t=d+1}^{365} \left(y_{i,t} - \beta_0 - \beta_{d*p+1} t - \sum_{j=1}^{d*p} \beta_j x_{i,t,j} - \sum_{j' \in baseline} \beta_{j'} x_{i,j'} \right)^2 + \lambda \left(|\beta_{d*p+1}| + \sum_{j=1}^{d*p} |\beta_j| + \sum_{j' \in baseline} |\beta_{j'}| \right)$$

where j' is index for baseline variables. $x_{i,j'}$ is the j'-th baseline variable for individual i. Note that baseline variables do not vary with time t.

The coefficient estimates generated by lasso can serve as a sign of feature importance. It is reasonable to conjecture that there might be similarities in the estimated coefficients for different interns. As a first step towards testing this conjecture, we can try to see how clustered the estimated coefficient vectors are. If we discover that there is some evidence

for clustering, we can use multitask learning to learn parameters for the individuals jointly rather than separately.

5.3.2 Multitask Learning

Multitask learning is a technique which allows us to share information across similar individuals. In Hastie, Tibshirani and Friedman's book, K-means is a cluster analysis that can partition observations into K clusters where each cluster contains observations sharing the same closest center [16, p. 460].

We clustered 23 interns in the 2015-2016 data set based on the 167 coefficient estimates generated from a lasso model trained with 15 lags at the individual level. Given 23 data points $\beta_{1:23}$ where each β_i is a $1*167$ vector, we maintain two more variables - cluster means u_k and cluster assignments $Z_i^k \in \{0,1\}$, where $i = 1, ..., 23$ and $k = 1, ..., K$. The value of K is assumed to be known. Then two steps are performed:

1) Initialize u_k arbitrarily, $k = 1, ..., K$

2) Repeat (a) and (b) until convergence:

(a) update for all $i = 1, ..., 23$:

$$Z_i^k = \begin{cases} 1, & if\ k = \text{argmin}_{j \le K} ||\beta_i - u_j|| \\ 0, & \text{otherwise} \end{cases}$$

(b) update for all $k = 1, ..., K$:

$$u_k = \frac{\sum_{i=1}^{23} z_i^k \beta_i}{\sum_{i=1}^{23} z_i^k}$$

To determine the number of clusters K, we looked at the within-cluster sum of squares (WCSS), which is calculated by

$$\sum_{k=1}^{K} \sum_{\beta_i \in S_k} ||\beta_i - u_k||^2$$

where S_k is a cluster and u_k is the mean of the data points in cluster S_k.

The more clusters we have, the smaller WCSS is. The optimal number of clusters is the K from which the decrease in WCSS is not significant anymore. After picking out K, we treated each cluster as an individual and fit lasso for each cluster.

T+1 (Multitask Learning):

For each $k \in K$, the lasso coefficients $\hat{\beta}_\lambda^L$ minimizes,

$$\sum_{\{i,t\} \in \text{training}} \left(y_{i,t} - \beta_{k,0} - \beta_{k,d*p+1} t - \sum_{j=1}^{d*p} \beta_{k,j} x_{i,t,j} \right)^2 + \lambda \left(|\beta_{k,d*p+1}| + \sum_{j=1}^{d*p} |\beta_{k,j}| \right)$$

where $\{i,t\}$ is an individual-time combination $\big\{\{i,t\} : \beta_i \in S_k \text{ and } t \in [d+1, 365]\big\}$.

5.3.3 Deep Learning

Deep learning involves a large class of nonlinear models [16, pp. 392-397]. In this study, we used vanilla neural networks and recurrent neural networks. Vanilla neural networks are the most widely used neural networks, flattening lags in time series data into one long vector which is trained as a data point. On the other hand, recurrent neural networks iterate through the lags in time series data and keep a state that contains information about previous lags [13].

5.3.3.1 Vanilla Neural Networks

Vanilla neural networks revolve around three objects: layers, neurons, and activation functions. The activation function, the number of layers and neurons are determined before training. In Hastie, Tibshirani and Friedman's book, vanilla neural networks are described as a model that takes in an input vector, applies a nonlinear function on the inputs (which is called an activation function), and gets a representation of the inputs [16, pp. 392-397]. This representation is called a neuron. Several such neurons make up a layer and can be treated as inputs for the next layer. In a regression setting, the final layer usually includes one neuron which gives the prediction results.

Given our data, if we were to predict an individual's mood on day t with d lags and p variables per lag, vanilla neural networks take in vectors including mood $y_{i,t}$, trend variable t, and $d \cdot p$ predictors $x_{i,t,1:d\cdot p}$. Despite having numerous combination choices, since we only have about 365 observations per individual, to avoid overfitting, we picked the number of layers to be 1 and tuned neurons in a set of 1,5,10,15,20. The activation function was chosen as the sigmoid function, which is widely used. See Figure 2 for an overview of the model.

T+1:

For each individual i:

For $t \in training$, a neuron $z_{i,t,k}$ in layer 1 can be formulated by:

$$z_{i,t,k} = \frac{1}{1 + e^{-\left(w^{(1)}_{i,k,0} + w^{(1)}_{i,k,d\cdot p+1} \cdot t + \sum_{j=1}^{d\cdot p} w^{(1)}_{i,k,j} \cdot x_{i,t,j}\right)}}$$

where $w^{(1)}_{i,k,j}$ is the weight on the j-th predictor to construct the k-th neuron in the first layer for individual i. We seek to find weights that would make the model fit the data well. Thus, all weights in the model are updated via the gradient descent process of minimizing the loss function, which in the setting of this study, can be formulated by

For each individual i:

$$Loss = \sum_{t \in \text{ training}} \left(y_{i,t} - w_{i,0}^{(2)} - \sum_{k=1}^{m} w_{i,k}^{(2)} z_{i,t,k}\right)^2$$

where $w_{i,k}^{(2)}$ is the weight on k-th neuron in the first layer for individual i and m is the number of neurons [16, pp. 392-397]. Note that on the second layer, we only applied the linear function.

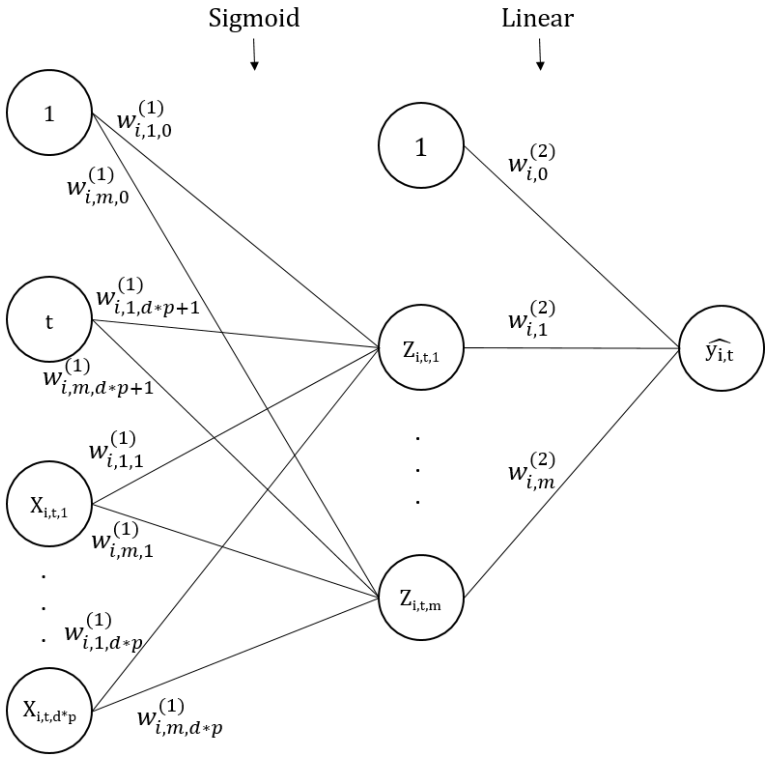

Figure 2: Overview of the vanilla neural network model.

5.3.3.2 Recurrent Neural Netowrks (RNN)

Vanilla neural networks treat time series data not much differently from regular data, except for the addition of a trend variable. On the other hand, in Chollet and Allaire's book, they described a new nonlinear method called recurrent neural networks which can learn the temporal behavior for a time sequence [13]. It takes in a data point composed of mood y_t and a $d \cdot p$ matrix of inputs where d is the number of lags, and there are p variables per lag. After taking in the data, recurrent neural networks train from one lag to the next, carrying information from previously trained lags. There are different types of RNNs, and in this study, we trained with three of them – simple RNN, long short-term memory (LSTM), and gated recurrent unit (GRU).

In the context of IHS, we trained RNN with two layers - an RNN layer and a dense layer. The RNN layer transforms the matrix input into a 32×1 vector, which is transformed into a

scalar through a linear activation function. Then the parameters are optimized by rmsprop. See Table 4 for details of the two layers.

	Layer1	Layer2
Activation Function f	tanh	linear
Input Dimension	$d \times p$	32×1
Output Dimension	32×1	1×1
Recurrent Function f_r	Sigmoid	NA
Optimizer	rmsprop	

Table 4: Description of layers in RNN models.

In layer 1, when simple RNN trains over lags, at lag j, it generates output by combining the input on lag j and a state, which is the output of the lag j+1 via an activation function. The output on lag j will be forwarded to the training on lag j-1 as a state which includes information from previous lags. The algorithm could be written out as:

T+1:

For each individial i:

For $t \in training$:

1) Initialize $state_{i,t,d} = 0$

2) For lag j from d to 1

a. Generate $output_{i,t,j} = f_1(W_{i,1} \cdot x_{i,t,j} + U_{i,1} \cdot state_{i,t,j} + b_{i,1})$

b. Update $state_{i,t,j-1} = output_{i,t,j}$

3) $\hat{y}_{i,t} = f_2(output_{i,t,1})$

$x_{i,t,j}$ is the $p \times 1$ vector from lag j of day t for individual i. f_1 and f_2 are the activation functions in the two layers respectively. Activation functions are applied element-wise. $W_{i,1}$ and $U_{i,1}$ are parameterized matrices and $b_{i,1}$ is a bias vector in layer 1.

Although theoretically simple RNN can retain information from many lags ago, the long-term dependency may not be accounted for [13]. According to Chollet and Allaire, LSTM is designed to solve this problem by adding a term into the activation function [13]. The term, called carry, aims to keep information from the past lags intact and enable later lags to retrieve previous information with ease. With this additional term, the algorithm of LSTM can be written as:

T+1:

For each individial i:

For $t \in training$:

1) Initialize $state_{i,t,d} = 0$, $carry_{i,t,d} = 0$

2) For lag j from d to 1

a. Generate $output_{i,t,j} = f_1(W_{i,1} \cdot x_{i,t,j} + U_{i,1} \cdot state_{i,t,j} + V_{i,1} \cdot carry_{i,t,j} + b_{i,1})$

b. Generate three terms which will make up $carry_{i,t,j-1}$

$m_{i,t,j} = f_r(U_{i,m} \cdot state_{i,t,j} + W_{i,m} \cdot x_{i,t,j} + b_{i,m})$

$n_{i,t,j} = f_r(U_{i,n} \cdot state_{i,t,j} + W_{i,n} \cdot x_{i,t,j} + b_{i,n})$

$k_{i,t,j} = f_r(U_{i,k} \cdot state_{i,t,j} + W_{i,k} \cdot x_{i,t,j} + b_{i,k})$

c. Update $state_{i,t,j-1} = output_{i,t,j}, \quad carry_{i,t,j-1} = m_{i,t,j} * k_{i,t,j} + carry_{i,t,j} * n_{i,t,j}$

3) $\hat{y}_{i,t} = f_2(output_{i,t,1})$

where f_r is the recurrent function. f_r is also applied element-wise. $*$ is element-wise multiplication.

Compared to LSTM, GRU is cheaper to run [13]. It combines the carry term into the state term [17]. The algorithm can be written as:

T+1:

For each individial i:

For $t \in training$:

For lag j from d to 1,

$r_{i,t,j} = f_r(W_{i,r} \cdot x_{i,t,j} + U_{i,r} \cdot output_{i,t,j+1})$

$z_{i,t,j} = f_r(W_{i,z} \cdot x_{i,t,j} + U_{i,z} \cdot output_{i,t,j+1})$

$state_{i,t,j} = f_1(W_{i,1} \cdot x_{i,t,j} + U_{i,1} \cdot (r_{i,t,j} * output_{i,t,j+1}))$

$output_{i,t,j} = z_{i,t,j} * output_{i,t,j+1} + (1 - z_{i,t,j}) * state_{i,t,j}))$

$\hat{y}_{i,t} = f_2(output_{i,t,1})$

6 Results and Discussion

6.1 Setting 1

The discussion in this section focuses on comparing different algorithms in setting 1, which are noted in Table 3. The analysis is trying to solve prediction problem T+1 on the 2015-2016 data.

6.1.1 Multitask Learning

After clustering the 167 lasso coefficient estimates of 23 individuals by K-means, we can see from Figure 3 that WCSS does not decrease significantly after 4. This suggests that there may be 4 clusters of individuals with similar coefficient estimates.

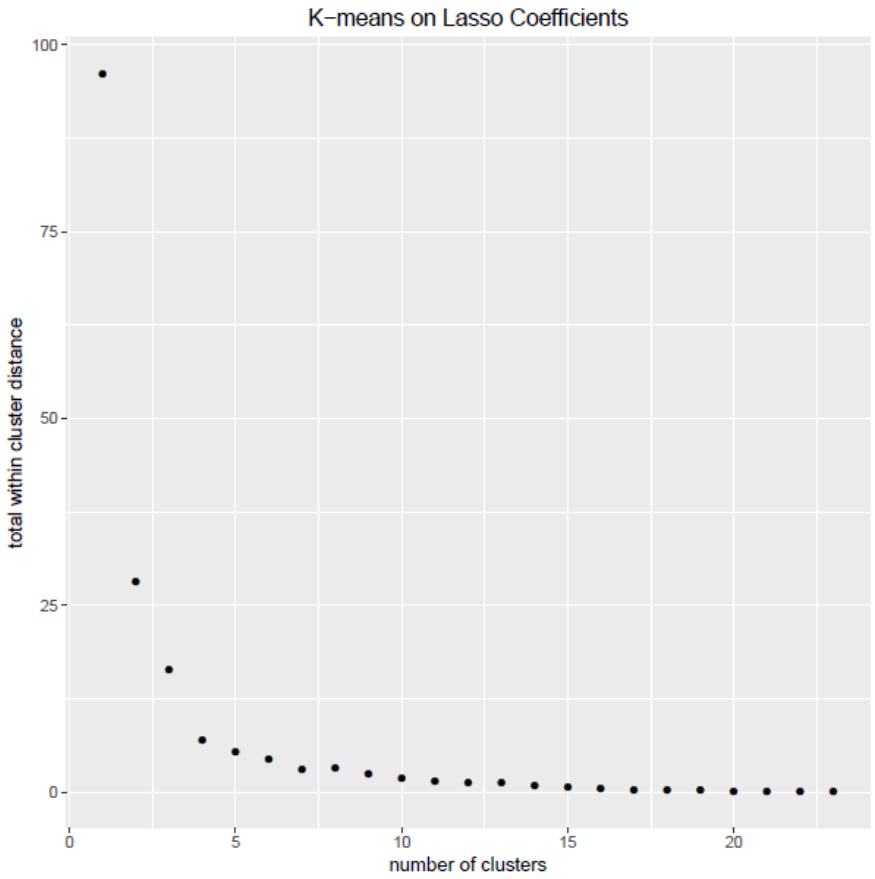

Figure 3: WCSS vs number of clusters from K-means.

Then we were curious whether there was any trend in prediction accuracy regarding clustering. Thus, we clustered the 23 individuals into different numbers of clusters ranging from 1 to 23. For every number of clusters, we fit a lasso model on each cluster and calculated each model's MSE. The results are in Figure 4. We can see that despite seeing 4 clusters in Figure 3, test MSE does not have a significant drop at 4 clusters. The general trend is decreasing in a fluctuating fashion.

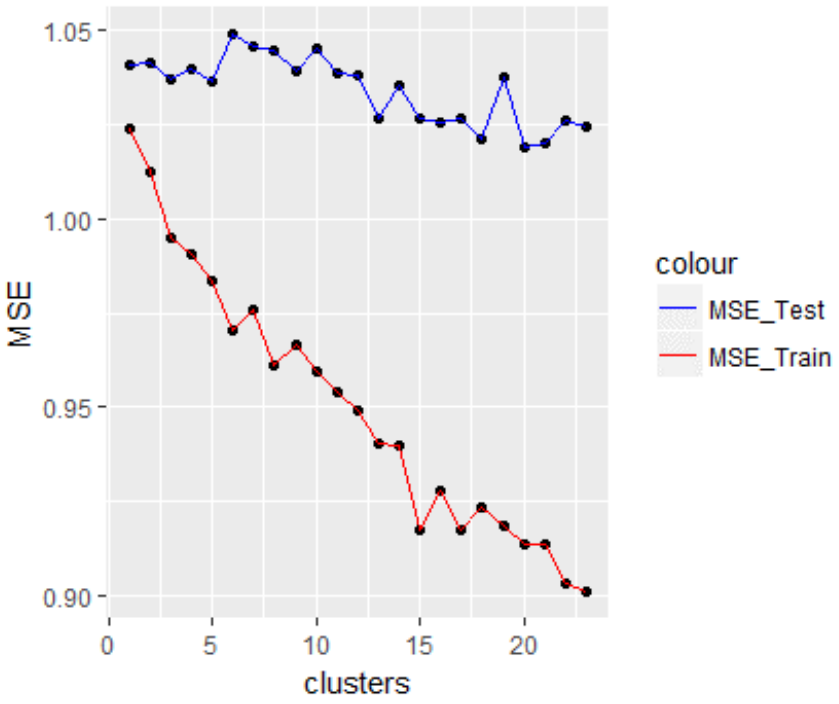

Figure 4: MSE of different cluster.

6.1.2 Model Comparison

Linear models trained include lasso on the individual level (23 cluster), lasso on the cluster level(4 clusters), and lasso on all individuals (1 cluster). Nonlinear models include vanilla neural networks and RNNs. See Figure 5 for the boxplots of the R^2 values for each model. Overall, lasso trained at the individual level performed the best.

The medians of linear and nonlinear models are similar, while the spread of nonlinear models is larger with more negative outliers. This suggests that there may not be a lot of evidence of nonlinear relationships between mood and other variables. Among three linear algorithms, we can learn that lasso on the individual level has a higher median, while the other two models look similar. Among nonlinear algorithms, GRU and LSTM have higher medians than simple RNN or vanilla neural networks, and GRU has less negative values than LSTM. Thus, GRU is the best model in nonlinear methods.

It is noticeable that R^2 values of the same individual do not show consistency between methods. For example, the R^2 values of some individuals are around 0.2 in linear but negative in nonlinear.

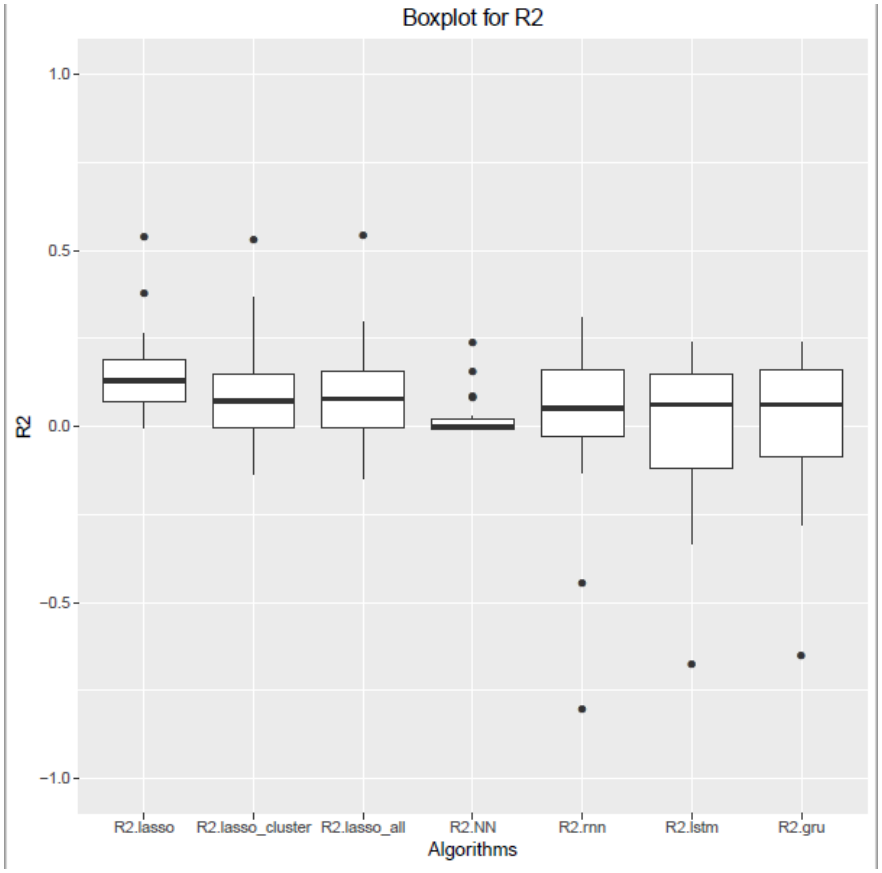

Figure 5: Side-by-side boxplots for R^2 of each algorithm in setting 1.

6.2 Setting 2

The discussion in this section focuses on comparing different algorithms in setting 2, which are noted in Table 3. The analysis is trying to solve prediction problem T+1 on the 2015-2016 data and the 2017-2018 data.

6.2.1 Prediction with Lasso

Table 5 displays statistics for R^2 in each cross-validation split from lasso to solve T+1 on both data sets. -Inf happens when every observed mood in the test set of an individual are the same. NA occurs when there is no observed mood for an individual in the test set. The summary statistics in Table 5 were calculated by excluding the -Inf and NA entries.

From Table 5, Figure 6 and Figure 7, we have a couple observations. 1) With the increase of cv, -Inf and NA becomes more prevalent in the 2017-2018 data, which indicates that the degree of missingness in mood increases over the course of the study. 2) In both data sets, the centers of R^2 do not change much with the increase in cv. It shows a counter-intuitive point that an increase in training data sample size does not improve prediction. A conjecture

may be that the current evaluation of prediction effectiveness is affected by missingness. The sample size of the test data decreased over time. Thus, when it comes to the later cv splits where there are only a few test data points, the variance becomes low and the prediction with the mean will be sufficient. In this case, R^2 may not increase because R^2 uses mean prediction as a benchmark. Another conjecture could be with the increase of cv, there are more imputed training data, which does not help with prediction. 3) R^2 in the 2015-2016 data set is much less negative than that in the 2017-2018 data set. Since 2015-2016 data set has less missingness, it may indicate that a better imputation method could help with prediction.

		Total	-Inf	NA	Min.	1st Qu.	Median	Mean	3rd Qu.	Max
cv 1	2015	23	0	0	-1.24	-0.09	0.00	-0.04	0.09	0.62
	2017	510	28	91	-49.10	-0.28	-0.07	-0.40	0.00	0.86
cv 2	2015	23	0	0	-0.67	-0.13	-0.04	-0.06	0.06	0.42
	2017	510	40	120	-41.35	-0.31	-0.08	-0.50	0.00	0.47
cv 3	2015	23	0	0	-0.64	-0.08	-0.01	-0.03	0.11	0.21
	2017	510	48	158	-49.68	-0.39	-0.08	-0.53	0.02	0.52
cv 4	2015	23	0	0	-1.25	-0.21	0.02	-0.15	0.07	0.17
	2017	510	54	206	-11.12	-0.30	-0.08	-0.36	-0.01	0.50

Table 5: Summary statistics of R^2 from lasso model on 2015 and 2017 data for T+1 in setting 2.

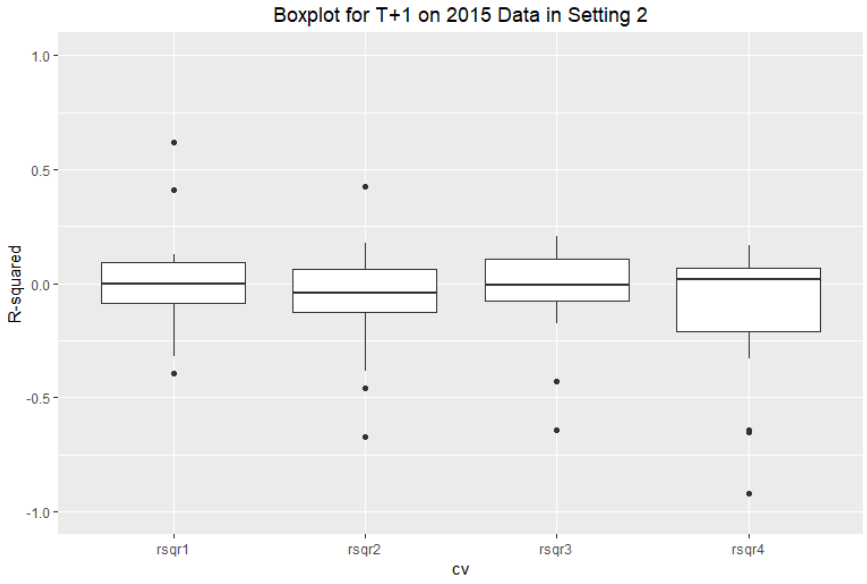

Figure 6: Side-by-side boxplots for R^2 of block cross validation from lasso on 2015 data in setting 2.

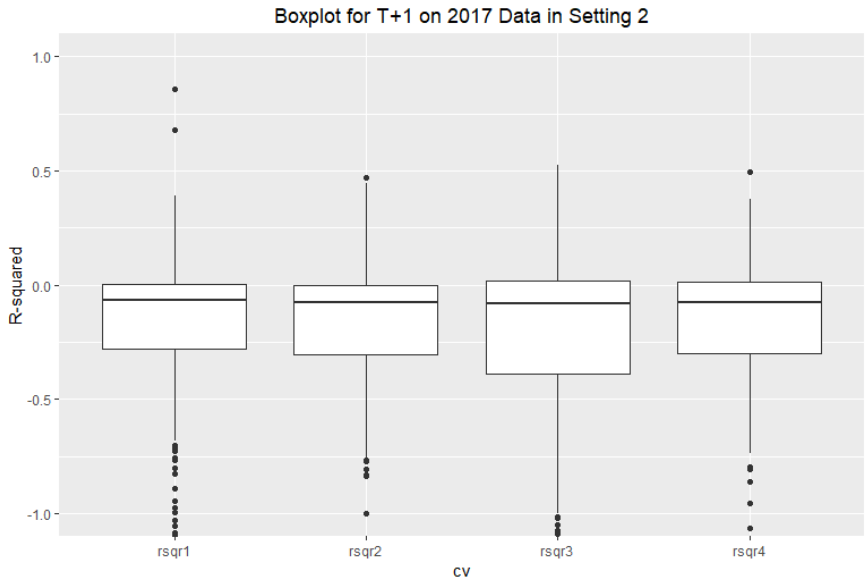

Figure 7: Side-by-side boxplots for R^2 of block cross validation from lasso on 2017 data in setting 2.

6.2.2 Prediction with GRU

In setting 2, we only applied GRU on the 2017-2018 data set. Note that only in this section, we used yesterday's sleep and activity as lag 1 for sleep and activity predictors.

Training GRU includes tuning parameters. The parameters we focused on are epoch, step epoch and batch size. In GRU, the training data is trained for multiple iterations which are called as epochs. An epoch is constructed from several step epochs. In each step epoch, the model trains a batch size of the total training data points. In this section, we only tuned epoch from 1 to 50 with step epoch and batch size fixed. Table 6 show details of the parameter values used.

Table 7 shows the results after fitting GRU. The statistics show the results are much worse than lasso in Section 7.2.1. A conjecture may be that the current GRU model is overfitting. In block cross-validation, the sample size in the training data is always less than 300, which is not large. However, we have 32 units in the hidden layer, which results in multiple $32 * p$ and $32 * 32$ matrices. The next step is to reduce the hidden units to a smaller number.

Name	Explanation	cv 1	cv 2	cv 3	cv 4
Epoch (Tuned)	Number of iterations over the entire training data provided	50	50	50	50
Step Epoch	Total number of steps before declaring one epoch finished	4	3	5	5
Batch Size	Number of data points in each Step Epoch	20	50	50	60

Table 6: Parameters in GRU in setting 2.

	Total	-Inf	NA	Min.	1st Qu.	Median	Mean	3rd Qu.	Max
cv 1	510	35	72	-428.56	-13.45	-5.38	-15.05	-1.40	0.15
cv 2	510	39	118	-263.97	-19.64	-9.05	-18.51	-3.35	0.24
cv 3	510	49	157	-128.13	-5.18	-1.65	-7.90	-0.07	0.19
cv 4	510	55	205	-401.59	-6.21	-1.20	-10.18	-0.06	0.16

Table 7: Summary statistics of R^2 from GRU on 2017 data for T+1 in setting 2.

6.3 Setting 3

The discussion in this section focuses on comparing different algorithms in setting 3, which is described in Table 3. The analysis is trying to solve prediction problem N+1 on the 2017-2018 data. The interns were split into a training set and a test set. The train/test ratio is 3:1.

Table 8 displays statistics for R^2 from lasso with baseline variables and lasso without baseline variables. The total number of individuals in the test sets are different because we excluded 3 individuals in the lasso with baseline model because their baseline variables are missing entirely. -Inf happens when every observed mood for an individual is the same. NA occurs when there is no observed mood for an individual from day $d + 1$ to 365. The following summary statistics were calculated excluding -Inf and NA entries.

From Table 8 and Figure 8, there is not enough evidence showing baseline variables improve the prediction capability of lasso for N+1. Adding baseline variables to lasso generates more negative R^2, and a more negative center as well.

	Total	-Inf	NA	Min.	1st Qu.	Median	Mean	3rd Qu.	Max
Baseline	125	4	0	-6.28	-0.41	-0.22	-0.31	-0.05	0.69
No Baseline	128	2	2	-8.17	-0.48	-0.18	-0.33	0.08	0.80

Table 8: Summary statistics of R^2 from lasso models on 2017 data for N+1 in setting 3.

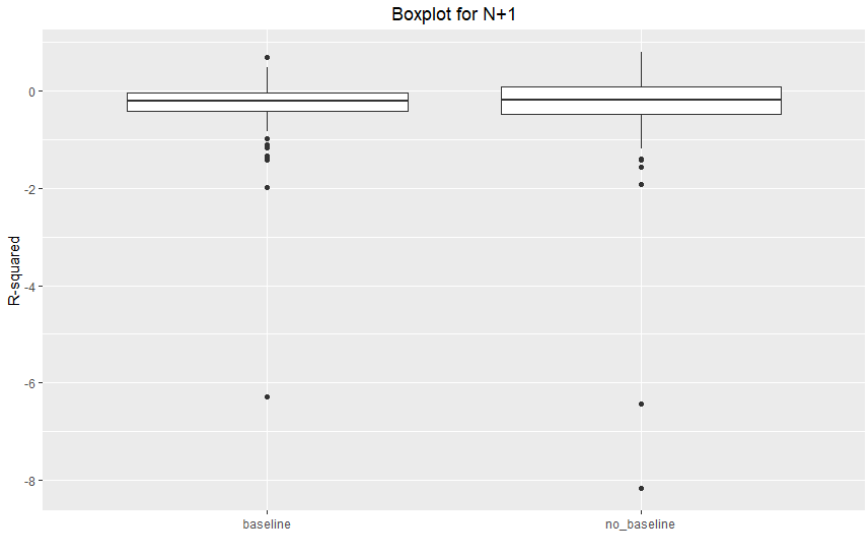

Figure 8: Side-by-side boxplots for R^2 of each algorithm on 2017 data in setting 3.

Since lasso without baseline performs slightly better, we looked at the important variables selected by the model. Table 9 shows the variable names and their coefficient estimates. mood.l1-mood.l7 are lags 1-7 of the mood variable. deepCount.l6 and deepCount.l2 are lag 6 and lag 2 of variable deep sleep counts, respectively. remCount.l1 is lag 1 of variable rem sleep count.

In general, mood from the past is important for future mood prediction. Yesterday's mood is the most important predictor. An increase in previous mood will bring an increase in future mood. Besides mood, Table 9 shows evidence that the number of cycles of deep sleep and rem sleep may help with future mood prediction.

Order	1	2	3	4	5	6	7	8	9	10
Variable	mood.l1	mood.l2	mood.l7	mood.l3	mood.l4	mood.l5	mood.l6	deepCount.l6	deepCount.l2	remCount.l1
Coefficient Est.	0.5650	0.1091	0.0793	0.0575	0.0501	0.0493	0.0391	-0.0050	0.0039	0.0021

Table 9: Variable importance from lasso without baseline on 2017 data for N+1 in setting 3.

7 Conclusion and Future Work

This work has compared the performances of prediction algorithms on predicting self-reported mood based on data from mobile phones and wearables. Firstly, we specified two prediction problems - T+1 and N+1. Next, in order to take the high degree of missingness and the time-ordering quality of our data into consideration, we investigated and applied carry-forward imputation methods and block cross-validation in addition to mean imputation and vanilla cross-validation. With these methods, we built up three different settings where prediction algorithms were applied. Then we extended the currently existing linear models by introducing multitask learning and explored nonlinear methods - neural network

models, including vanilla and recurrent neural networks. There is no major improvement by applying nonlinear algorithms, but this may be due to some limitations of the study. 1) The 2015-2016 data set is small. 2) Another limitation is the high rate of missingness in both data sets. All of the code for the project, which is written in R, is available at https://github.com/smengjie/mood_prediction.

Some future work includes:

1. Multitask learning can be applied on the 2017-2018 data set. Since the 2017-2018 data set has more individuals, clustering will probably be more accurate.

2. More imputation methods could be tried and evaluated. In the study, we saw that under the same setting, algorithms generally gave more accurate prediction on the 2015-2016 data set, which has less missing values. This could be a sign that more accurate imputation can improve prediction accuracy.

3. More cross-validation methods for time series can be discussed and applied.

4. In setting 2, GRU model can be modified. 1) Lags of sleep and activity in GRU can be modified to be more consistent with the rest of the study. 2) Tune learning rate and epoch. 3) Tune the number of units in the hidden layer in a set of numbers smaller than 32.

5. Try LSTM and simple RNN in setting 2.

There are also some midterm goals:

1. Try classification prediction problem. Based on current literature review, most studies focused on predicting using a discrete scale. Previous studies on mood classification can also be used as a benchmark.

2. Include geo-location data in the analysis of the 2017-2018 data set.

3. Analyze the 2017-2018 data set with interventions.

4. Understand the reliability of evaluation of mood prediction capability when there is a high degree of missingness in the test set.

8 Acknowledgements

I would like to thank Prof. Ambuj Tewari for his guidance. Also, I am grateful to Tim NeCamp for his help and insights. Next, I would like to thank Jessica Liu for her involvement in the analysis. Lastly, this data is supported by the Sen Lab at the University of Michigan. I would like to record my appreciation to Dr. Srijan Sen and his team for their dedication in depression research.

References

[1] Depression. http://www.who.int/news-room/fact-sheets/detail/depression, 2018. Accessed: 2019-06-14.

[2] Chiara Baglionia, Gemma Battaglieseb, Bernd Feigea, Kai Spiegelhaldera, Christoph Nissena, Ulrich Voderholzerac, Caterina Lombardob, and Dieter Riemanna. Insomnia as a predictor of depression: a meta-analytic evaluation of longitudinal epidemiological studies. *Journal of Affective Disorders*, 135(1-3):10–19, 2011.

[3] Andreas Ströhle. Physical activity, exercise, depression and anxiety disorders. *Journal of neural transmission*, 116(6):777, 2009.

[4] Darius A Rohani, Maria Faurholt-Jepsen, Lars Vedel Kessing, and Jakob E Bardram. Correlations between objective behavioral features collected from mobile and wearable devices and depressive mood symptoms in patients with affective disorders: Systematic review. *JMIR mHealth and uHealth*, 6(8):e165, 2018.

[5] Terumi Umematsu, Akane Sano, Sara Taylor, and Rosalind Picard. Improving students' daily life stress forecasting using lstm neural network. *IEEE International Conference on Biomedical and Health Informatics*, 2019.

[6] Sarah Taylor, Natasha Jaques, Ehimwenma Nosakhare, Akane Sano, and Rosalind Picard. Personalized multitask learning for predicting tomorrow's mood, stress, and health. *IEEE Transactions on Affective Computing*, PP(99), 2017.

[7] 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*, pages 715–724. International World Wide Web Conferences Steering Committee, 2017.

[8] Joost Asselbergs, Jeroen Ruwaard, Michal Ejdys, Niels Schrader, Marit Sijbrandij, and Heleen Riper. Mobile phone-based unobtrusive ecological momentary assessment of day-to-day mood: an explorative study. *Journal of medical Internet research*, 18(3):e72, 2016.

[9] Thaddeus Tarpey. Linear transformations and the k-means clustering algorithm: applications to clustering curves. *The American Statistician*, 61(1):34–40, 2007.

[10] Natasha Jaques, Sara Taylor, Akane Sano, and Rosalind Picard. Multimodal autoencoder: A deep learning approach to filling in missing sensor data and enabling better mood prediction. In *2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII)*, pages 202–208. IEEE, 2017.

[11] David A. Kalmbach, Yu Fang, J. Todd Arnedt, Amy L. Cochran, Patricia J. Deldin, Adam I Kaplin, and Srijan Sen. Effects of sleep, physical activity, and shift work on daily mood: a prospective mobile monitoring study of medical interns. *Journal of General Internal Medicine*, 33(6):1–7, 2018.

[12] Srijen Sen, Henry R. Kranzler, KrystalJohn H., Heather Speller, Grace Chan, Joel Gelernter, and Constance Guille. A prospective cohort study investigating factors associated with depression during medical internship. *Archives of General Psychiatry*, 67(6):557–565, 2010.

[13] Francois Chollet and J. J. Allaire. *In Deep Learning with R.* Manning Publications, 2018. pages 180-208.

[14] Trevor Hastie Robert Tibshirani Gareth James, Daniela Witten. *An Introduction to Statistical Learning with Applications in R.* Springer, 2013.

[15] Jatin Garg. Using k-fold cross-validation for time-series model selection. https://stats.stackexchange.com/questions/14099/using-k-fold-cross-validation-for-time-series-model-selection, 2017. Accessed: 2019-06-14.

[16] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning Data Mining, Inference, and Prediction.* Springer, 2013.

[17] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, et al. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.