Predicting mood from time-series data

Jorn Verheggen[2680220], Anastasios Mitsigkolas[2715091], and Zhexin Liu[2689786]

Vrije Universiteit, De Boelelaan 1105, 1081 HV Amsterdam, Netherlands

1 Introduction

An individuals mood often has great effects on their life. A consistently bad mood can lead to sadness, inability to focus, loss of friends, lower job performance. All these negative effects can compound to even greater problems like obesity, depression and even suicide. If we could better understand what makes up a persons mood we could maybe help manage it. This could lead to both great individual as well as societal benefits.[1]
Development like the the smartphone give researchers new opportunities to study mood. Not only can the mood of study participants be more frequently assessed, researchers can also gain some insight on what could cause a good or bad mood for a study participant. During this project we have analyzed a dataset from 27 individuals who all tracked their mood and phone usage throughout their daily life during a period of about three months. From this dataset we have built two predictive models that are able to predict the average mood on a users next day given some prior information about the users phone usage.

2 Dataset pre-processing

Before we can start processing the dataset we first need to understand what we are looking for. In the case of this project that means understanding how a mood value might manifests itself among the list of provided features in the dataset. In order to do this we will first take a look at the psychological literature on mood and emotions. After that we will explore the dataset to find what features are most likely to be informative in predicting an individual's mood. We will then use various statistical approaches to determine what features should be used in the model.

2.1 Psychological analysis

In order to predict a person's mood it would be useful to have a list of factors that determine an individual's mood. However, it is impossible to compile an exhaustive list of all factors that determine a persons mood from a psychological point of view. We however do know that a persons emotions consists of three parts[2].
Subjective component These include primary emotions like fear, sadness, surprise or contentment.

Physiological component The physiological component is associated with changes in the human body. This could be physical pain, exhaustion hunger, feeling warm or cold etc.
Cognitive component The cognitive component is related to a cognitive interpretation of an individuals surrounding and situation. For example, realizing that one's train is going to be delayed does not have a direct impact on one's surroundings. However, it could cause that individual to be in a bad mood.

There are two ways these emotions can manifest themselves in the dataset. Either as causes to some emotion or as reaction to some emotion. For example the cognitive task of checking one's office app regularly could cause stress which could lead to a bad mood. Or using a game/ entertainment app could indicate a physiological state of relaxation which could cause a good mood.
It could also be the case that the dataset shows a reaction to a good or bad mood. It is known that in periods of lower mood people tend to not want to be with other people. A bad mood therefore could manifests itself as a lower amount of calls or text messages then usual.

Furthermore, an individuals personality could also strongly contribute to ones mood[3]. Some people tend to be more optimistic while some people tend to be more pessimistic during daily life. While this likely contributes to their average mood scores it also likely contributes to how they react to certain events. One person could for example react heavier to bad weather while another person might not be as effected.

2.2 Dataset exploration

The dataset consists of one table with 376913 rows and five columns. Each row of the table corresponds to a certain event. The columns hold information about the index of the dataset, the id of the study participants, a timestamp of the event, the variable(type of event) and the value of the event. In total there are 27 unique users, 19 unique variables and 113 unique dates spread out between 17-2-2014 and 9-6-2014. From initial analysis there are a couple of issues that need to be addressed.

Personalities With 27 unique users we can assume that we have a big variety of different personalities and mood responses. This presents a potential problem for our model. We would ideally like to use one model for all people. This way we would not have to tweak hyper-parameters or modal setup per individual user. However, we do want the model to be individualized in some way to make use the individual mood responses.

Seasonality Another problem is that not all users have registered their mood on every day within the time frame. From figure 1 we can see that the first users started the trial almost a month earlier than others. This could create a potential problem as seasonality might interfere. It is conceivable that due to the warmer weather or longer days people on average might be in a better mood in

June compared to March. If this is the case than this would have to be compensated in some way. To test this hypothesis we created a plot with the mean daily mood throughout the trial. From this plot we can see that there are no seasonal changes in mood rating because the line stays close to a mood rating of 7 throughout the entire duration of the trial. We can therefore assume that we have no changes due to seasonality and a mood ratting in May or June is comparable to a mood rating in March.

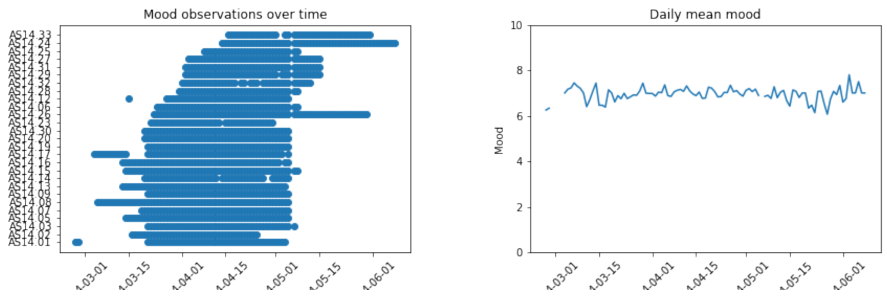


Fig. 1: Seasonal patterns

Weekly routine Although the dataset does not seem to suffer from changes in season we were able to detect another form of repetition. It seems that there is a weekly pattern in the mood ratings. The mood rating between 9h and 12h on Monday seems to be consistently lower than the mood rating on Saturday during the same time period. Intuitively this seems very indicative of a regular weekly routine. This is also intuitive when we think about how most people experience their mood throughout the week. Typically people relax more on the weekends which lifts their mood. Then when they go to work on Monday their mood drops a bit.

7 Centered Another potential problem is that the mood rating scores seem to be concentrated very strongly around the 7. Mood ratings of below a four and above a 9 are exceedingly rare. With so few observations it could be difficult for a model to correctly predict these mood ratings.

Available variables The dataset has 19 unique variables. Most of these variables indicate a clear aspect of the users daily life. For example call, sms, appCat.communication and appCat.social all contribute to knowledge about the social engagement of the user. For our purposes it is important that these variables are both homogeneous and generalize. With homogeneity we intend to say that the app should have only one discernible purpose. If one and the same app is used to do both taxes and call friends for example then a model with this

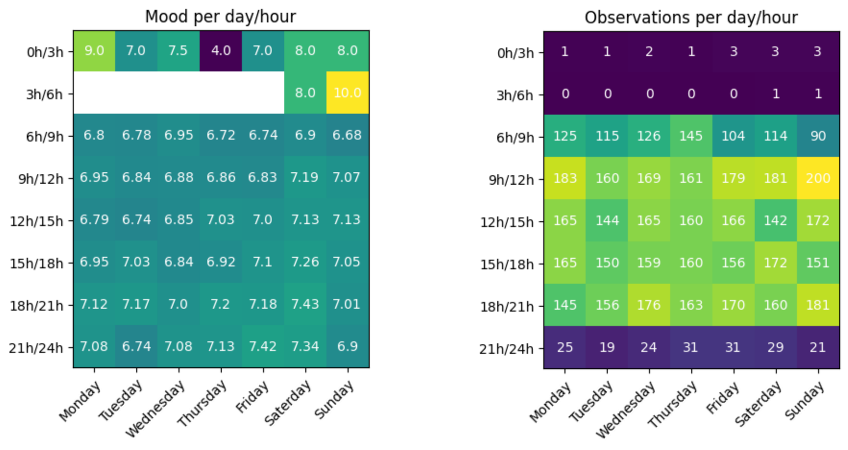


Fig. 2: Weekly mood pattern

app would likely produce very unreliable results. When we say an app should be generalize we mean that the purpose of the app should be the same for every user. If we would like to have a model that can characterize the mood of multiple different users than we cannot have apps that are used for different purposes by different people. Among the 19 variables we would consider two variables that do not meet these two criteria, namely appCat.unknown and appCat.other.

2.3 Feature engineering

For the feature engineering step we decided on five feature engineering measures.

Removing AppCat.unknown and AppCat.other We decided not to include AppCat.unknown and AppCat.other in the feature engineered model. The reason for this is explained before as we consider these variables to not meet our homogeneity and generalizability requirements.
Use relative values Instead of using absolute values are comparing the values to the users daily average. Let's say that a given user averages 3 SMS's a day. We are then going to use the number of SMS's relative to 3. So if on a day a user sends 4 SMS's we will use +1 as our feature. This allows us to include the habits and personality into the model without customizing the model for every person
Use day of week as feature We have seen that the day of the week can have a big impact on the average mood rating. Therefore we will use this day of the week as a feature.
Using a 7 day prediction window We know that there is a weekly periodic change in mood rating. Because of this we would like to consider a full week's worth of data when predicting a new mood value. Conversely if for example we

would only have two days of prior data, and that data would contain the weekend we would not have a good comparative working day rating.
Using mean daily sum as predictor feature There are various ways we can interpret a daily amount of data. We could for example take the amount of events, the median value of events, the mean value of events etc. We chose to use the sum of values as our predictive feature. We chose this because we this is the most indicative of a user's daily activity. For example, a user could open the game app 10 times a day to play for one minute. However, if a user's opens the game app and plays for an entire hour than this is more likely to have a big effect on the mood value.

We also consciously decided not to include some other measures for various reasons. We would also like to mention them here.
Statistical analysis To narrow down the list of possible features we could use statistical analysis like linear regression to filter out the less predictive variables. We however chose not to do this. The list of features is already relatively small. We do not need to reduce the number of features for purposes of increased computing power. Secondly, every effective machine learning algorithm will naturally filter out any unusable data. If a feature does not lead to inconsistency like with AppCat.unknown and AppCat.other it does not need to be removed.
Sub daily data Our research question is about about creating a model to find the mean mood value for a given day. Since this is an entire day rather than a part of a day we don't need to find out the mood at every point in the day. We therefore decided that we can combine the data from an entire day rather than using the specific order.

3 Methods

For this assignment we have built two models. The first one is a feature engineered model that uses an MLP regressor model to with preprocessed data. The second model is an LSTM that is able to take the raw time-series data as input.

3.1 Feature engineering model

Data preparation Since the initial dataset is in melted format (variables in a column and their values in another column), it was completely necessary to find a systematic way to reshape this 2-dimensional table to a format capable to be used by machine learning approaches. Therefore, we grouped the dataset by the id variable in order to repeat the process separately for each patient involved in this study. After that we reshaped the remaining sub-set for each patient using columns as features and rows as instances.

History aggregation Classical machine learning approaches are not always capable to handle features series of temporal data. Therefore we have used the

following workflow to transform our data to be suitable for these types of algorithms. During the trial each study participant has delivered data in non-regular intervals. Our first step was therefore to aggregate this data into 1 day bins. As was explained previously we do this by obtaining the mean mood and sum values of each variable of each person for the duration of one day. This is visualized in Fig.3(A). To make this process clearer, the notation of $d_k : X_{im}$ will be used from now on and it indicates a vector of the sum of the variables $X_{1...i}$ for a corresponding k^{th} day.
Machine learning algorithms are not able to predict the average mood value of the next day based only on the average values of the current day. For that reason, we are going to construct features from the bottom up. There were two available approaches that we are mentioning. One could use rolling mean windows Fig.3(B) to construct new features that include not only the information of the current day but also information of some previous days. This method is suitable (?) to follow the trends of a given period. On the other hand, one could use expanding mean windows Fig.3(C) form the beginning of the timeline or from the end. This method is capable of having much more memory in comparison with the rolling windows of information along the timeline. The method which was followed is the rolling mean windows using a period of 7 days. We are using rolling windows of 7 days based on the assumption that the behavior and thus the mood of a patient is a periodical phenomenon and hence it is more likely for a predictor to predict it correctly by considering the behavior along the passed week.

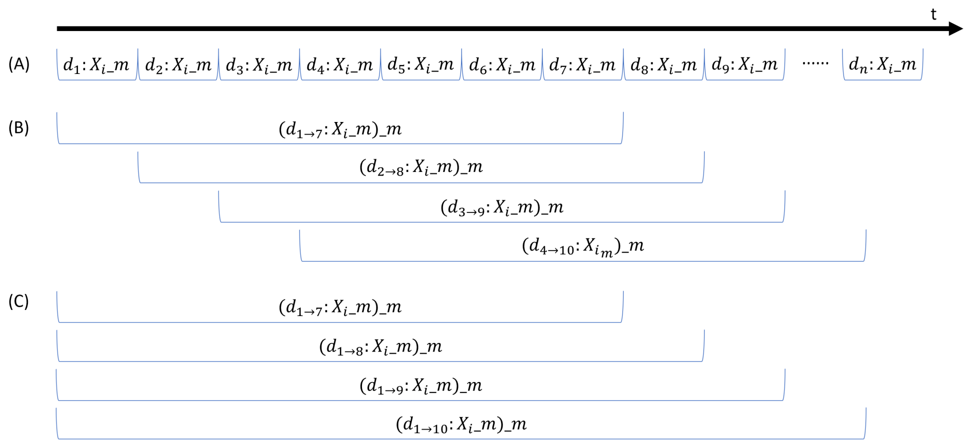


Fig. 3: Bins-Windows

MLPRegressor Model The regression model used was the standard Multilayer Perceptron regressor model in sklearn package in Python and it optimizes the squared-loss using LBFGS or stochastic gradient descent. After grid search hyperparameter tuning we concluded to an optimized Multi-layer Perceptron regressor defined by 100 hidden layers sizes, logistic activation and alpha parameter equal to 5e-05.

Training and validation As for validation scheme a simple 90%-10% random splitting of the modified, after the features engineering, dataset was used: the training set was used during grid search with cross validation for the hyper parameter tuning and for the training of the final optimized model while the testing set was used to evaluate the final model. To evaluate the trained model, we calculate the Mean Square Error (MSE) between the predicted mood values and the actual values. Besides, we set a benchmark model where the prediction was the average mood value from the previous day, and we calculated the MSE between this and the actual values.

Performance As it was mentioned before the test set consists of randomly selected instances over the entire dataset. It does actually consist of fixed periods of 7 days for different patients. The trained model over the training dataset was next used making predictions over the entire randomly obtained training dataset. After the training and the testing procedure we calculated the MSE between the actual mood values and the prediction, and between the actual mood values and benchmark model constructed before. We derived MSE = 0.295 for the regressor, and a lot higher value MSE = 0.6 for the benchmark.

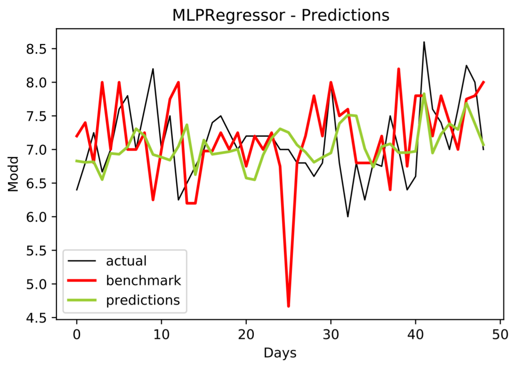


Fig. 4: Mood prediction for 50 days and for different patients using MLPRegressor model and time aggregation

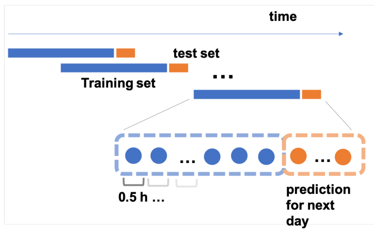
3.2 Temporal model

Data Transformation In the temporal model we merged the data from the different study participants. To increase data density consistency we only used data ranging from 2014-3-20 to 2014-5-01. Fig. 5a shows why these dates were used. The data was then undersampled by 0.5 hour. Missing values were filled by the values from the previous time stamp.
The temporal model used was the standard LSTM model in Keras package in Python[4]. Although the features aggregation was not applied in this model, the Pearson correlation was done on the 19 variables and selected the 5 variables (including the output variable mood) with correlation coefficient over a cut-off 0.3.
Then to make sure the data could be fitted to a ML model, variables were first checked if they are stationary. Then the data was separated into input and output patterns. Observation at the previous time step are used as an input to forecast the observation at the current time stamp. In addition, 12*0.5 hours' previous lag time stamps were included for each instance. The lag time stamps data and past observations as input were conducted was because during Back-propagation through time (BPTT), the recurrent layer would take the output from the previous time stamp to accumulate error. So the input from prior time step would influence the error and decide the weight update. [1] Therefore, we used the last half hour's observation as input value. Finally, we scaled the data to a value between -1 and 1 to meet the default hyperbolic tangent activation function of LSTM model.

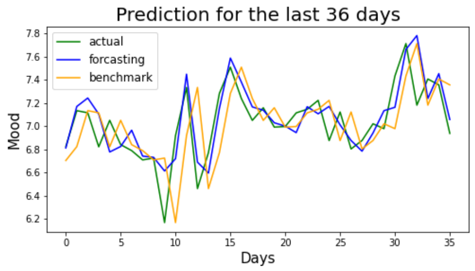
LSTM Model The LSTM model was defined with 50 neurons in one hidden layer, and one output layer for predicting the mood. Mean Absolute Error (MAE) was used as loss function, and Adam stochastic gradient descent was used as optimization scheme.

Training and validation For validation scheme the walk-forward model validation was used: test set is chosen behind the training set and we increase the size of the training set each validation iteration. Since the ultimate goal is to predict the mood value for the next day, we predict the next 48 instances (24 hours) and calculate the mean as the prediction for the next day. (Fig. 5a) In order to carry out statistical analysis, we do the validation 100 times on the entire dataset. To evaluate the trained model, we calculate the Mean Square Error (MSE) between forecasting mood values and actual values. In addition, we set a benchmark model where the prediction was the mood value from the previous day, and we calculated the MSE between it and actual values.

Performance For each forward validation run, we calculated the mean of the predictions of the mood values over 24 hours. After the training we calculated the MSE between the actual vs predictions, and between the actual vs benchmark. We got 0.027 for the former one, and a lot higher value 0.23 for the latter one.



(a) Walk-forward model validation



(b) Mood prediction for the last 36 days

Fig. 5

This means the RNN model we built performed a lot better than the benchmark model. This difference could also be witnessed from the line plot (Fig.5b) for the prediction for the last 36 continues days. The blue line (forecasting) and green line (actual) are almost on top of each other, while the orange line (benchmark) is far away.

4 Evaluation

To further validate the differences in their performance, a statistical analysis was conducted. For the prediction of each time stamp, we calculated the following:

$$Percentage\ distance = \ln(\frac{\|M_{prediction} - M_{actual}\|}{M_{actual}} \times 100\%)$$

The log function was used here to normalize the distribution. We concluded that the two groups have respective variances of 2.88 and 2.83. The p-value of 3.6381e-6 got from Levene test further showed their equal variances. Now that the assumption of normal distribution and equality of variances are met, we do the two-sample t-test. From this we got a p-value of 3.27e-11 (<0.01). This demonstrates that two groups are significantly biased. Therefore we have confirmed that the RNN model preforms better than the benchmark model. The same processing was done for feature engineered model. But unfortunately two distance groups have very biased variances even after normalization. This might be because feature engineered model had differentiated prediction performance for different patients. The outliers in (Fig.6, right panel) is probably because the rolling windows generated same mood values for the current day and the previous day.

The overall distinguishes between the performances could be seen from the Kernel estimation plot (Fig.6): LSTM model has lower log percentage difference and so it is shifted left away (close to -infinite). While the feature engineered model's distribution is on top of the benchmark model's distribution and so we could deduce there's not a significant difference between the two models. This observation could also be supported by the MSE that both models got.

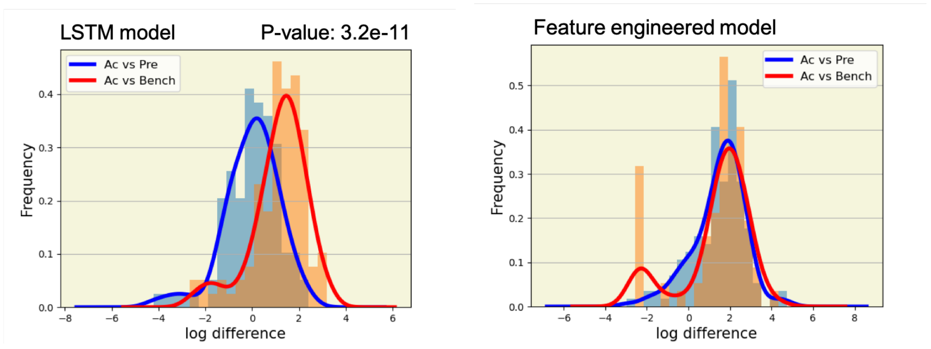


Fig. 6: Kernel density estimation

5 Conclusion

From the above mentioned statistical analysis one can safely conclude that the performance of the temporal model versus that of the features engineering model is much better and using classical machine learning approaches in forecasting problems could be challenging. Although this approach is very fast in terms computational time during both training and testing it is very challenging to choose the most appropriate features given a problem. To overcome this problem, requires experience on data exploration and very deep understanding of the nature of the studied problem. In addition to that it might be difficult for this model to become generalized for groups of people with very different habits while they are using their phone. On the other hand, forecasting approaches although are in general more computationally expensive could give us better results with less effort in terms of features engineering. Specifically, BPTT that is used by the recurrent neural net becomes computationally expensive as the time steps increases. Additionally, while the input time series increases there might be the possibility that the accumulated error using time derivatives vanish or explode the weights and make slow learning and model skill noisy.

Further studies about the machine learning approaches could explore additional features construction, features selection, different methods of aggregation the time such as expanding windows etc, and different machine learning algorithms.

For the temporal model we could try to tune the hypeparameter with the limit size of data set with more reasonable validation scheme. Besides, Truncated BBTP could effectively reduce the computation time and somehow boost the performance of RNN model. In some ways we could even combine the data engineering with the temporal model to improve the accuracy.

References

1. J. Hoogendoorn M. Ruwaard J. Asselbergs J. Riper H W., Pastor. Exploring and comparing machine learning approaches for predicting mood over time. *Smart Innovation, Systems and Technologies*, pages 37—-47, 2016.
2. Michael Gazzaniga, Todd Heatherton, and Diane Halpern. *Psychological Science*. Norton, New york, 2010.
3. What factors influence our mood. https://exploringyourmind.com/factors-influence-mood/.
4. M. C. Mozer. A focused backpropagation algorithm for temporal pattern recognition. *Theory, architectures, and applications.*, pages 37—-169, 1995.