Mood prediction of a telephone phone user

Hidde van Oijen (2590893), Tiddo Loos(2574974), and Sander Kohnstamm (2564598)

Vrije Universiteit Amsterdam

Abstract. This report has the aim to explore a best method for predicting the average mood of a person for the next upcoming day. The temporal data used for this research contains 33 persons and their telephone behaviour. The data has been be explored and analysed first. Then, the data was used to test different common Machine Learning (ML) algorithms and a Recurrent Neural Network (RNN) approach. During these test and interpreting the results, further adjustments have been implemented to maximise the prediction score. We have found that the common ML approach Random Forest, after feature engineering resulted in the best score for predicting the average mood for the next day of a telephone phone user.

Keywords: Machine Learning · Recurrent Neural Network · Time Series · Data exploration · Random Forest

1 Introduction

Learning from sequential data is a well studied topic in Machine Learning [1]. Where most classical problems focus on the prediction of a class or a value from certain features, sequential data has the advantage of the order of the data points being a feature itself. These models, while mostly used for text data, are also often used for regression problems, finding trends in sequential values such as weather, stock prices and mood values.

In this report we will explore a dataset from the domain of mental health. As smartphones become a larger part of our lives, the way and how much we use them will become a better indicator of our well being. There are now multi applications available which track a users habits, usage and sensory data during the handling of their smartphone. The dataset used for this report is from one of those applications. It contains smartphone data from 33 users between February and June. These users are also frequently asked for a rating of their mood and giving other variable input data during their day.

We will aim to build a model that can predict a users mood for the next day, given the input data up till that next day. To this end we will explore two main implementations. The first one is a more common Machine learning approach, where the features of the previous days are the input and the average mood for the next day is the target. The other is a sequential network approach, where the prediction is fully based on the trends in the past mood levels of the user.

Both of these implementations will be assisted with the necessary pre-processing. We will also review various feature engineering techniques to improve the performance of the models, such as adding and transforming features, dimensional reduction and an averaging rolling window. These performances will be compared to a baseline result, where the last day's average mood is the prediction for the next days average mood.

2 Pre-processing the Data

2.1 Dataframe

To use the data more easily, first some pre-processing has been done. The Pandas library was used to create a data frame. A pivot was done in such a way that every variable had its own column. For every person on every day a row was created. Cells of time containing column values were filled with the accumulated time values for that day per person. Taking the average time could give a wrong view because a person with many different data points could be seen as less active than a person with one data point that is a little higher overall. The 'mood', 'circumplex.arousal' and 'circumplex.valence' values were substituted with their mean value on each day. The 'mood' was rated on a scale from 0.0 to 10.0. both the 'circumplex' values were rated on a scale from -2.0 to 2.0.

2.2 Data analysis

Trend A trend was be recognised when the average mood over time was plotted of all single users. Often a dip in the average mood is followed by an increase in the average mood and vice versa. Fig. 1 shows an example plot of a user's mood over time.

There are two thing we can take away from this plot. Firstly, the previous day does not indicate the next day on a positive linear basis. Secondly there is a clear temporal relation between the days. As stated a dip often follows a spike. We will explore a Recurrent Neural Network (RNN) which very well suited for this kind of sequential data[9].

Correlation The correlations of the different categories are calculated and a correlation matrix is presented in Fig. 2. The figure indicates that the categories 'mood' and 'circumplex.valence' are highly correlated. Also, 'screen' time is highly correlated with 'activity', 'appCat.communication' and 'appCat.social'. Logically, other 'appCat's' are also correlated with 'screen', as they contribute to 'screen' time when using them. These correlations are taken into account during the feature engineering.

Outliers Boxplots of every category data were plotted separately. When analysing the boxplots, it suggested that were outlying data points present in different categories of the data. An example boxplot is presented in Fig. 3. Outlier handling

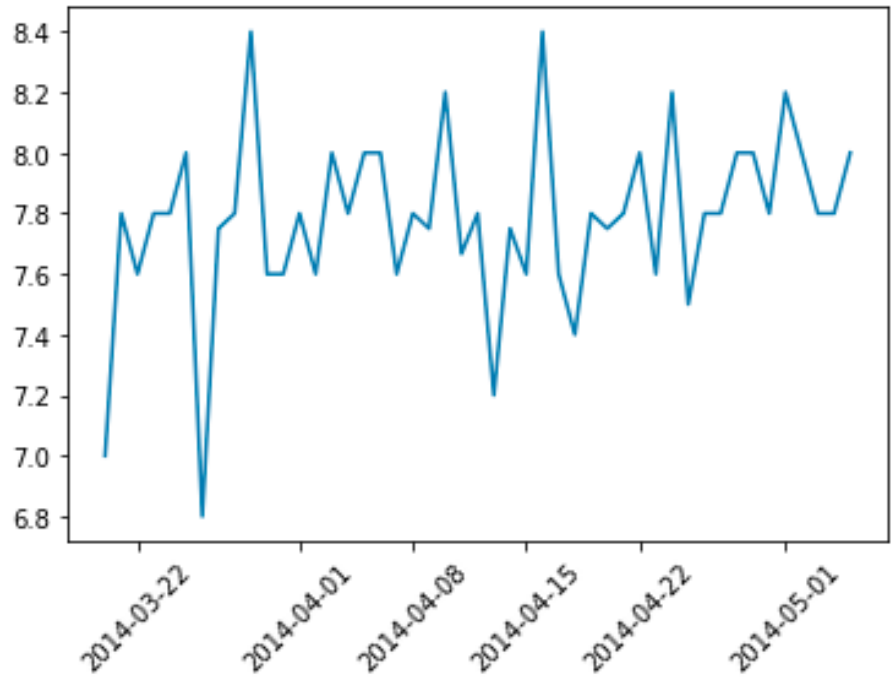

Fig. 1. Average mood over time of one user

will be explained in the Transformation section of the feature engineering paragraph.

2.3 Feature Engineering

Missing values One way of dealing with missing mood values is deleting the rows with no mood value. Rows with missing mood values can't be trained on in when using common ML models, as they predict the mood based on supervised learning algorithms. These algorithms need data with features and a corresponding output value to train on.

Another way to resolve this problem is to perform linear interpolation on the missing mood values. This method uses the outer values to estimate inner values. By using this method some valuable data can be saved which can have a positive effect on the model.

Added features To start, one of the features with most impact that can be used during training is the current average mood, in the dataset under column 'mood_train'. As we have seen in the previous section, there is a clear trend in the mood values of a user. On this pattern, training is easy for any machine learning model.

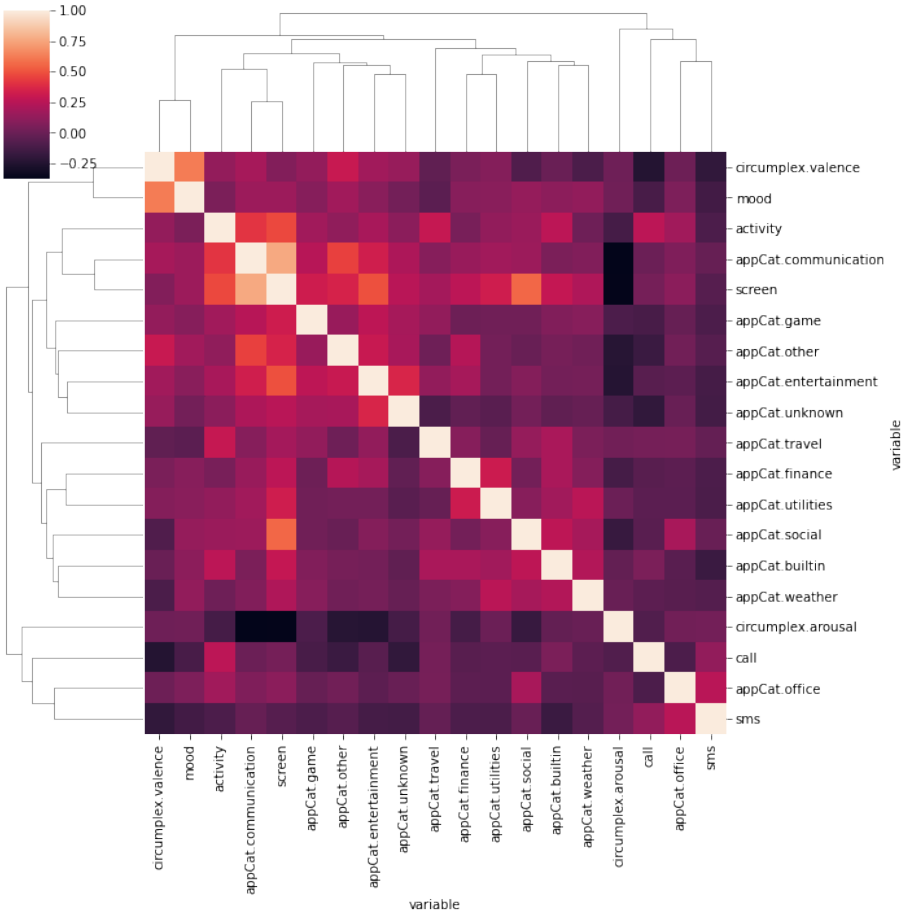

Fig. 2. Correlation matrix of variables

Secondly, a column 'night' was added based on the time information of the users. This column indicates True or False: True for a user who was active between 01:00AM and 06:00AM. This column was added rather intuitively. Being active at night could indicate a night of having less sleep. Less sleep has an effect on the mood on the next day [8]. Also, from our experience, using your phone late at night one way or another leads to a bad mood the next day.

Dimensionality reduction The total number of features that was left as input for our models was 20, many of which were correlated, as we have seen in the previous section, or obtained little information. Intuitively, this meant for us that we had to reduce the dimensionality of the feature space. This is mainly done to decrease the probability of overfitting and generalise the models. We approached this problem in the following two ways.

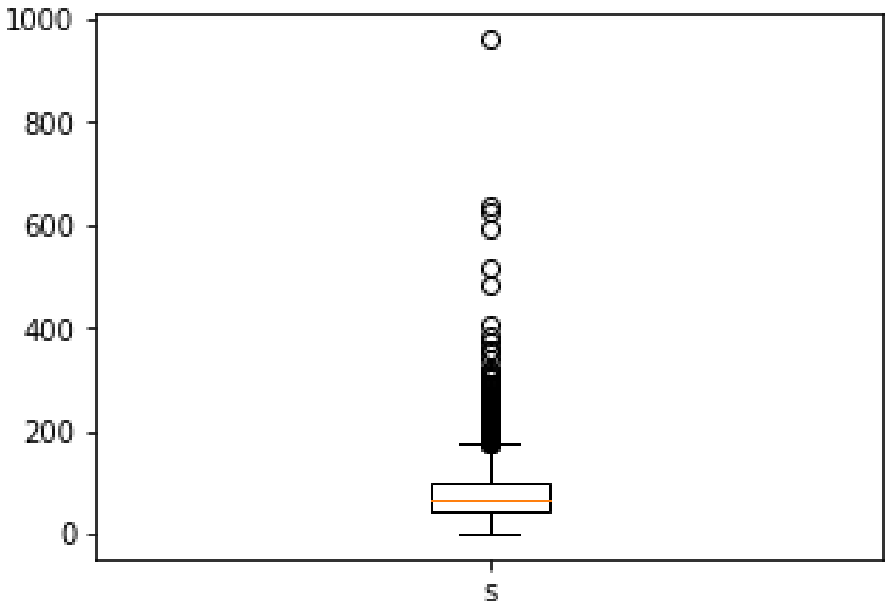

Fig. 3. Boxplot of 'screen' data

The first method that was implemented was Principle Component Analysis (PCA). PCA is a widely used unsupervised learning technique, often used for dimensionality reduction to decrease memory usage or generalise trained models [3]. It relies on the fact that most of the variance in a dataset is obtainable with compression onto a fraction of the dimensions. By only taking the eigenfaces of the dataset onto the first few directions of maximum variance, the dimensionality of the dataset can be greatly reduced.

Another method of reducing the amount of features was to simply take the subset of features which has the most significance on the outcome of the model. This is possible with a decision tree or a random forest, as it can be easily calculated which branches or decisions lead to the most significant changes in the outcome. This process yielded the scores shown in figure 4. Here, the feature 'Mood_train' is the mood of the current day.

From these values we inferred that most of the features below that were less important than screen were probably not beneficial to the model. We have also seen that screen is highly correlated with many of the features. That is why we have selected a subset of features from 'appCat.entertainment' and up in figure 4.

The latter of these two methods was chosen, because of better interim results.

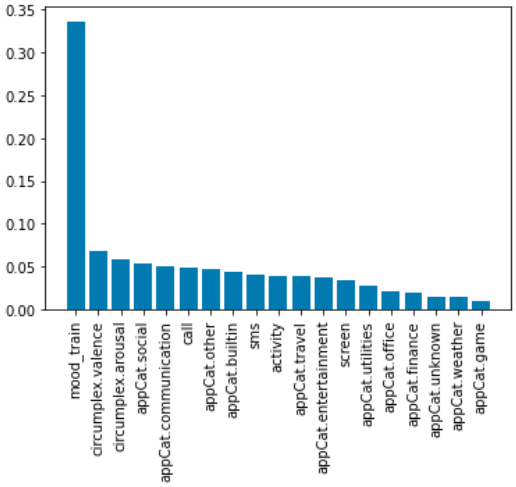

Fig. 4. Feature importance's

Rolling window As mentioned earlier in this report, a rolling window was used. A rolling window is a function that aggregates the feature values over a certain time period from a certain user. We have tried either a window that sums the values or one that takes the mean and we have found that a combination of the two works best. We take the summation for all the values that are cumulative (screen time, call time etc.) and the mean for the values that are bounded (arousal, mood)

Transformations As is common practice in processing datasets, the outliers were removed. To detect outliers, we used *Z-scores*. Data points are subjected to be outliers for Z-values > 3.0. Outliers are not taken into account when training our models and are calculated using the following equation:

$$Z = (X - \mu)/\sigma$$

Where X is the data, μ is the mean and σ the standard deviation.

Finally, normalisation was also performed on the input of the models. Normalisation is an important step to level the amount of influence that the different features have on the final outcome. As different features have different ranges, this can lead to widely varying amounts of influence. There are two main ways of normalising the dataset x given in the following equations:

$$z_i = \frac{x_i}{\max(x)} \qquad (1) \qquad\qquad z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \qquad (2)$$

After comparing the performance of the models using of both, we have chosen the former of the two equations. We did, however, added 4 to each value in the range +-2 to avoid negative numbers.

3 Learn using the dataset

3.1 Random Forest

The more common machine learning approach that we have implemented is the Random forest regression algorithm. This is an ensemble of multiple decision tree predictors in random order. The composition of the algorithm is also where it gets its name from [7]. The algorithm was invented by Breiman through making an addition on the bagging method on decision trees, which was also invented by Breiman [4].

Choice for random forest There are a couple of reasons that we choose random forest over other algorithms. The other algorithms that were considered were linear regression, K-nearest neighbours and multilayer perception. The most important reasons for choosing the random forest algorithm can be found below.

- The random forest with basic features already gave a good mean square error.
- The algorithm has a function were feature importance can measured. This can and has been used to make the final feature selection.
- It is resistant to overfitting [4].
- It can also be seen as user-friendly because of the low amount of hyper parameters [4].

3.2 Recurrent Neural Network

A different approach to the entire regression problem would be to implement a model that can make good use of temporal or sequential data. These types of models are Recurrent Neural Networks (RNN's). RNN's were originally derived from feed forward neural networks in the 1990's and have been the main foundation of sequential data analysis [6]. They have the ability to incorporate inputs along the axis of the nodes to process sequential data. A problem with sequential computation, however, is that gradients can easily vanish or explode.

Long-Short term memory network We will implement and evaluate a Long-Short term memory network (LSTM) which is build on the idea of RNN's and which solves this problem. LSTM's were introduced in 1997 [2]. They maintain certain gates with their network cell which control the flow of information. These gates include an input, output and a forget gate. With a forget gate, they can also decide on which features **not** to remember, which helps with the exploding gradients. Many more variations have been developed since its conception, however we will implement the original LSTM.

Implementation and challenges Our model is implemented in PyTorch using a tutorial found at [5]. This is not a multivariate LSTM, therefore it can not take into account different features. It only learns the sequential nature of our label, the average mood of a user per day, and bases its next prediction on the previous values stored in its memory cell.

To train on the sequential nature of the data, the dataloader has to perform some care full selection. We have chosen a window of 28 days (4 weeks), with the target label being the 29th day. This window was selected were ever the dataloader could within one user without interruption. So if a user has 35 sequential datapoints, the dataloader can retrieve 7 datasets from it (35-28 =7), with 28 features and one label. Then we train the same model on each user sequentially for 75 epochs. Both the time window and the epochs were selected after a brief hyperparameter search.

3.3 Metrics and Benchmark

Most of our model selection, hyperparameter search and feature engineering was done by the guidance of the mean squared error loss (MSE). Mean squared error is the most widely used loss function for regression problems. It is calculated with the following formula:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

Where n is the number of datapoints, Y_i is the target value and $\hat{Y}_i$ is the predicted value. These losses are calculated with a 5 fold cross validation training.

As an addition, we also have calculated the R^2 score for a bit more context. It expresses the algorithms ability to encapsulate the variance of the labels. It is calculated as follows:

$$1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

where y_i is the target value and $\hat{y}_i$ is the predicted value, also.

The benchmark we will test our implementations against is the value of the current day average mood as a prediction of the next days average mood. As the target values are unchanged, unscaled and unnormalised for each model, this will give us a good overview of the performance differences between all the models.

4 Evaluation

4.1 Results

The results of the models are given in figure 5. It is clear that there is not much difference between them, however, Random Forest does seem to perform the best of the common machine learning algorithms. We will move on with RF for

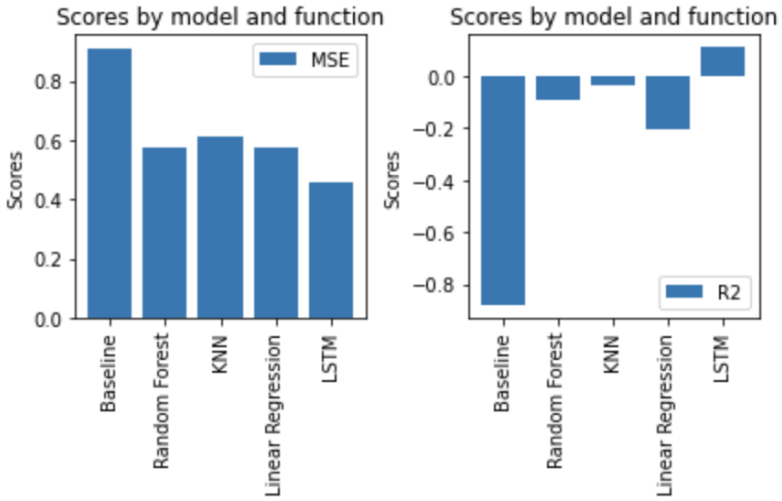

Fig. 5. The MSE (the lower the better) and R^2 (the higher the better) scores for the different algorithms without any feature engineering techniques.

the feature engineering. The LSTM does performs the best in this situation. All models perform very adequately compared to the benchmark.

In figure 6 the results of the different feature engineering techniques are shown. All techniques that we have tried did have a positive impact on the basic model. The combination of all positive techniques performs the best even though their individual impacts are leveled when used in combination with eachother. With this combination, the random forest model outperforms the LSTM model.

4.2 Analysis

As can be seen in the results, the Random forest algorithm has the best MSE and R^2 overall score after feature engineering. The explanation for this is that the other features next to the average mood over a couple of days have a significant impact. These explanatory features can't be used in the LSTM, which could explain the lower performance.

However, what can be seen is that the average mood over a couple of days has the biggest influence on the prediction in the Random Forest model. It could be that this variable hasn't achieved its maximum potential. This could mean that there is to little data points in the time series for the LSTM, or that there is to much missing data points in the interval. It could be the case that when the time series were measured over a longer period, the LSTM could outperform the Random Forest. The use of LSTM is known to be very good approach in time series data.

As future work, we suggest to measure the mood over a longer period of time to test if the LSTM could outperform the Random Forest.

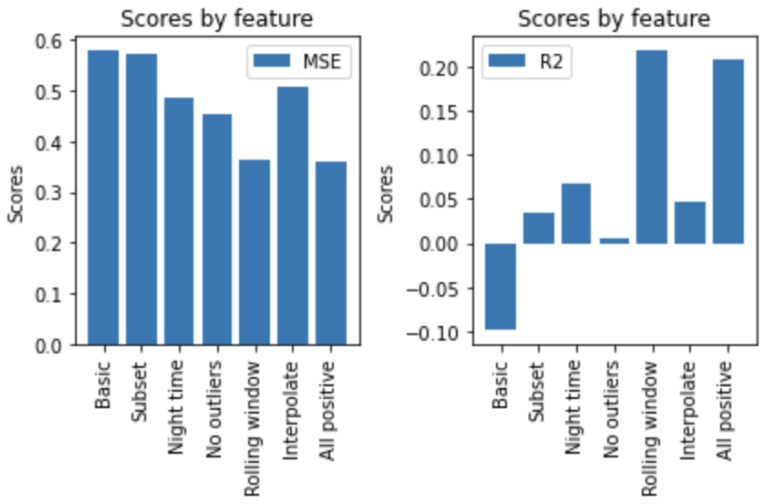

Fig. 6. The MSE (the lower the better) and R^2 (the higher the better) scores for the different variations on the features individually, trained on a Random Forest.

4.3 Conclusion

After analysing our results we conclude that the RF in combination with the different engineered features within the most optimal subset is the better method for predicting the average mood for the next day. This is due to the fact that the LSTM only takes the sequential mood values into account. These values most likely contain too little information, either because they are individual values or because the time series are not extended enough.

References

[1] Thomas G Dietterich. "Machine learning for sequential data: A review". In: *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)*. Springer. 2002, pp. 15–30.

[2] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[3] Sasan Karamizadeh et al. "An overview of principal component analysis". In: *Journal of Signal and Information Processing* 4.3B (2013), p. 173.

[4] Andy Liaw, Matthew Wiener, et al. "Classification and regression by randomForest". In: *R news* 2.3 (2002), pp. 18–22.

[5] Usman Malik. *Time Series Prediction using LSTM with PyTorch in Python*. https://stackabuse.com/time-series-prediction-using-lstm-with-pytorch-in-python/.

[6] Larry R Medsker and LC Jain. "Recurrent neural networks". In: *Design and Applications* 5 (2001).

[7] Mark R Segal. "Machine learning benchmarks and random forest regression". In: (2004).

[8] Michelle A Short and Mia Louca. "Sleep deprivation leads to mood deficits in healthy adolescents". In: *Sleep medicine* 16.8 (2015), pp. 987–993.

[9] 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. Perth, Australia: International World Wide Web Conferences Steering Committee, 2017, pp. 715–724. ISBN: 9781450349130. DOI: 10.1145/3038912.3052676. URL: https://doi.org/10.1145/3038912.3052676.