Advanced Assignment 1
Data Mining Techniques

Eui Yeon Jang, Hannah Lim, and Tim van Loenhout

2569512, 2677439, 2525199
Group 106

1 Introduction

There are a variety of mood tracking applications available for smartphones that log users' self-reported mood assessments along with various data retrieved using built-in sensors [2]. The data can be analysed to gain insights to factors contributing to certain mental states, and help build predictive models to forecast users' mood. Such information can be employed in various ways, such as, in music and media content recommendation [7] or screening, detecting, and monitoring different mental disorders [4, 5].

This report explores processing the raw data obtained from a mood tracking application and building models to predict the average mood of users on the following day. We consider two predictive models: a standard model using Support Vector Regression and a temporal model with Long Short-Term Memory network. We focus on building models that are not specific to any one user.

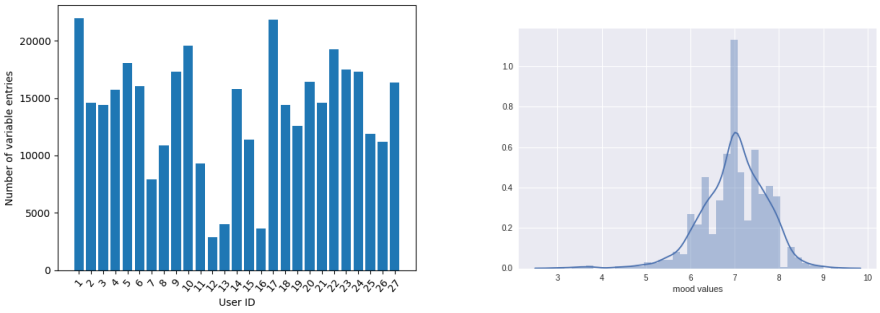
The report is structured as follows: Section 2 details data preparation and feature engineering. Section 3 focuses on the different models and their implementations. Section 4 and 5 presents and discusses the results.

2 Data

The provided raw data set consists of numerous sensory data of users' behaviour and their self-reported mood assessments. The data set contains entries for 27 unique users. It has a total of 376,912 entries, each entry is logged with a unique identifier for the user, a timestamp, a variable with its corresponding value. Multiple entries for a variable in a given day is possible, as well as no entry at all.

2.1 Initial Analysis

There are an average of 139,960 entries per user with a standard deviation of 5,118. The distribution of entries per user can be seen in Figure 1(a). The average entries per day are 191.03 with a standard deviation of 200.79. This means that there is a considerable difference in the number of entries per day. The data set contains total of 19 different variables, including our target variable, `mood`. The values of `mood` entries are centred around 7 as shown in Figure 1(b).



(a) (b)

Fig. 1. (a) Distribution of number of entries per user over all variables. (b) Distribution of the mood values.

The different variables in the raw data and percentage of the entries of their measurements in the data set are presented in Figure 2. Entries for `screen`, `appCat.builtIn`, and `appCat.communication` have the highest number of entries, whereas `appCat.weather`, `appCat.game` and `appCat.finance` have very few.

2.2 Data Preprocessing

2.2.1 Detecting and removing outliers. As our data set is rather small, bewaring outliers is even more important. In order to prevent the models from being biased by outliers, for each feature, we remove those values that lie outside of the z standard deviations from the feature's mean value. This poses a trade-off between the size and reliability of the data set. The optimal value for z is discussed in Section 2.7.

2.2.2 One-day aggregation. We first organise the data into a table such that each entry is logged with a user ID, a timestamp and a value for each variable. Subsequently, we aggregate all entries for one day into a single entry. For `mood`, `activity`, `circumplex.arousal`, and `circumplex.valence`, we take the average of their values as these are scaled scores. For the other variables, their sum is stored as these are time measurements or counts. When a variable contains no measurements over the whole day, a `NaN` value is stored.

Due to this aggregation, the time of day for each entry is lost. We discuss how to incorporate back the information in Section 2.3.

2.2.3 User identification. As the effect of the use of smartphone on the mental state is user dependent, we retain identification of the users, enabling the model to also learn the effects of behaviour on the mood of users individually.

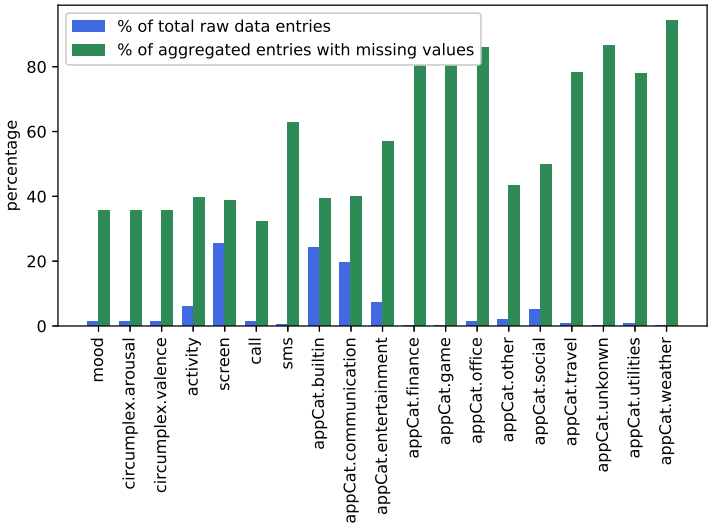


Fig. 2. Raw variables and their occurrence in the dataset

As our models cannot cope with categorical data, we use one-hot encoding: A new feature is created for each user ID and are given value `1` if the entry belongs to the corresponding user, and `0` otherwise.

2.2.4 Missing value interpolation. Unfortunately, the data set contains many entries with missing values - each entry has at least one variable that is missing a value. Figure 2 shows the percentages of missing values per variable. Although some variables contain a high percentage of missing values, we decide to keep all the variables for now as removing them may lead to loss of relevant information on an already small data set. Hence, we fill in the missing values while trying to introduce as little bias as possible.

The missing values of `mood`, `activity`, `circumplex.arousal`, and `circumplex.valence` are interpolated by taking the average of n previous days. The window n is determined for each variable by performing autocorrelation, where the correlation of the values of each variable to their previous days is measured. This is done in such manner that no value is interpolated by using another interpolated value. The window sizes were 3 for `mood` and 2 for the rest. In case the previous days do not contain any values, the overall average of the feature for the user is used, which is a common approach for interpolation.

For all the other variables, we set the missing value to `0`. These variables are measured automatically by the smartphone, and therefore, we can assume that if the value is missing, there was no measurement for that day. For example, if the values for `call` is missing, then the user has not made a phone call.

We believe whether the value for a feature was missing or not could provide valuable information for our models. We discuss how we incorporate this information in Section 2.3.

It should be noted that, we do not use interpolated `mood` values as target values - we only consider entries corresponding to non-interpolated `mood` values as input to our models.

2.3 Creating New Features

2.3.1 Part of day. Due to the one-day aggregation in Section 2.2.2, the temporal information of the sub-entries is lost. Since mood differs during different parts of the day [3], we use the timestamps of the sub-entries to generate new features. This is done by creating four new features capturing the information about the parts of the day during which the underlying sub-entry were logged. These parts of day are: `night` (00.00-06.00) `morning` (06.00-12.00), `afternoon`(12.00-18.00) and `evening` (18.00-00.00). For instance, if two calls were made between 06.00 and 12.00, the value for the new feature `morning_call` is `2`.

2.3.2 Missing-flag. We also generate new binary features representing whether the value was initially missing for each feature. For instance, if the variable `appCat.weather` had a `NaN` value for that entry, the new feature `missing_appCat.weather` will store `0`. This additional information could provide insight into the liability of each feature value, subsequently giving the model the opportunity to put more weight unto the feature values that are based on more underlying data.

2.4 Finding Significant Features

In order to test which variables contain the most predictive power, we compute the F-score, based on a simple linear regression function, between each variable and `mood`. Figure 3 shows that `activity` has a high correlation with the mood, hence we expect this variable to be a valuable feature. On the other hand, the variables `sms`, `call` `appCat.builtin`, `screen`, and `appCat.entertainment` show a low correlation with `mood`, hence we consider these variables to be insignificant. In Section 2.7 we test whether removing these insignificant features positively influences the model's performance.

2.5 Multi-day Window Aggregation

In order to predict the mood for the following day, we use information from the previous n days. The optimal size is tested in Section 2.7. Note that the window looks at the n previous days (related to the dates), rather than n previous entries. This is done to prevent the windows from containing information that is based on entries that have been logged a longer time ago than the window size implies.

As the standard model does not deal with time-series data, the information from the previous days needs to be aggregated to a single entry. The data set for

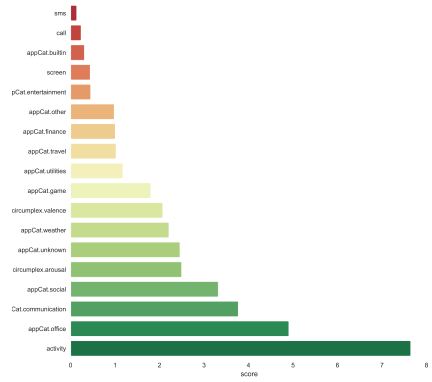


Fig. 3. Correlations between different variables.

the temporal model is be aggregated as it considers all the days in the selected window sequentially.

To aggregate the multi-day window for the standard model, we consider mean and sum. For the features `mood`, `circumplex.arousal`, `circumplex.valence` and `activity` we use the mean, whereas for the rest of the variables, we aggregate the values by taking their sum. We also use sum for the part-of-day and the missing-flag features. Summing part-of-day features then provides a ratio of sub-entries logged at different parts of the day, but also the number of raw data entries used to creating the entry. Furthermore, by using sum for the missing-flags, the model also gains information on the number of values every feature of the window is based on.

Additionally, we believe that including the variance for the features aggregated using mean may result in a better performance as this extra measure could provide information on how reliable the average value for the given feature is. We test whether this is a feature to be included in Section 2.7.

Finally, we suspect that information from more recent days are more correlated to the day's mood. Hence, we also experiment with taking a weighted aggregation over the features. This weighted statistic weighs each entry by its the inverted number of days to the target variable. Whether this approach results in a better performance is also discussed in section 2.7.

2.6 Normalisation

Some learning models put more emphasis on features containing larger values. In order to prevent this, we apply normalisation on both the final data set for the standard and temporal model. This method squeezes all values between 0 and 1 while maintaining the original ratios. An alternative method would be

standardisation. However, this approach assumes that the values within a feature are Gaussian distributed, which is a claim we cannot make, especially given the relatively small size of the data set.

2.7 Feature Tuning

In order to test the aforementioned feature engineering choices, we tune our feature set using a form of the top-down approach by training a `sklearn`'s out-of-box Support Vector Regression model. We compare the performances of the models trained on the different feature sets and select the optimal setting to produce our final data set to input to our models.

For each generated feature set, we make a 8:2 split to create a training and a test set. We train and validate using a 10-fold cross-validation on the shuffled training set, using mean absolute error (MAE) loss. We use MAE because it allows for a more interpretable comparison between models. Due to time constraints, we do not perform a full grid-search. Instead, we start with default settings, and iteratively update each feature to its most optimal setting as we move down the list [1].

Tuning shows that a feature set based on an outlier detection threshold of 2 standard deviations and a weighted window size of 5 is most optimal. The inclusion of user ID features, and exclusion of part-of-day and missing-flag features, as well as the variables relatively uncorrelated to mood (`sms`, `call` `appCat.builtin`, `screen`, and `appCat.entertainment`) yield better performance. Furthermore, adding the standard deviation to the features that use mean as the aggregation metric does not improve the performance. With these settings, the final number of features to input into the standard model is 43.

Given that a neural network model is more tolerant to less significant features, and moreover may squeeze out some potential information of the available features, we create a separate data set with all the same settings and include all the features for the temporal model, a total of 142.

The data sets for both models contain less than 500 data instances.

3 Models

We implement two different models, one using a standard machine learning and another a temporal approach. We build a Support Vector Regression (SVR) as our standard model, and a more complex Long Short-Term Memory (LSTM) network as the temporal model. The SVR is less proned to overfit and works well with a small data set. The LSTM network deals with time-series data and is able to learn temporal trends in the data.

[1] For tuning, we use the following order: threshold size for outlier reduction, window size for multi-day window aggregation, inclusion of user ID features, inclusion of part-of-day features, inclusion of missing-flags features, inclusion of standard deviation when using a mean aggregation metric, weighted or non-weighted aggregation, exclusion of insignificant features.

3.1 Support Vector Regression

We implement `sklearn`'s SVR and tune for hyperparameters `C` and `tolerance`. `C` is the penalty term for misclassifications, and `tolerance` specifies the tolerance for the stopping criteria to stop searching. All other hyperparameters are set to `sklearn`'s default. We make a 8:2 split of the final data set for each model for a training- and a test-set, and perform a 10-fold cross-validation on the shuffled training set using MAE to find the optimal hyperparameters. We use MAE for the same reason as explained Section 2.7. We find that `C` $= 0.1$ and `tolerance` $= 0.1$ provide the best average validation loss. We also experimented with different maximum iteration values, and there were no signs of overfitting. The final model is trained on the entire training set.

3.2 Long Short-Term Memory

We use the `pytorch` framework to implement the LSTM network. The model has one hidden layer of size 50 and one output layer of size one. Given the low complexity of the optimisation problem, we do not expect the problem requires a more complex model. We use the root mean squared error loss[2] in combination with the Adam optimiser [6] and a learning rate of 0.001 to train the model.

The data set is split into training and test set using 8:2 ratio. In order to prevent overfitting, the training set is shuffled and further split into a training and validation set using 9:1 ratio. The model is stopped early when the standard deviation over the previous n validation losses decreases below a threshold t. The optimal values for n and t are determined empirically, resulting in an early stopping operation which interrupts the training process when the validation set is converged, while the training loss remains decreasing (Figure 4).

3.3 Benchmark

To provide a baseline against which to test the final models, we implement a non-learning model which predicts the average mood of the following day to be the same as the average mood of the current day: $\text{mood}_{d+1} = \text{mood}_d$. The missing mood values are interpolated in the same manner as described Section 2.2.4. The data set is split into a training and test set using 8:2 ratio. Since this model does not require training, we ignore the training set.

Furthermore, given the autocorrelation result of a 3 day correlation for `mood`, an additional model is computed, where instead of predicting tomorrow's mood based solely on today's mood, we take the average over the previous three days. We refer to this model as the alternative benchmark. The data set is again split using 8:2 ratio.

[2] We use RMSE during training because it is smoothly differentiable, and RMSE will be higher than or equal to MAE.

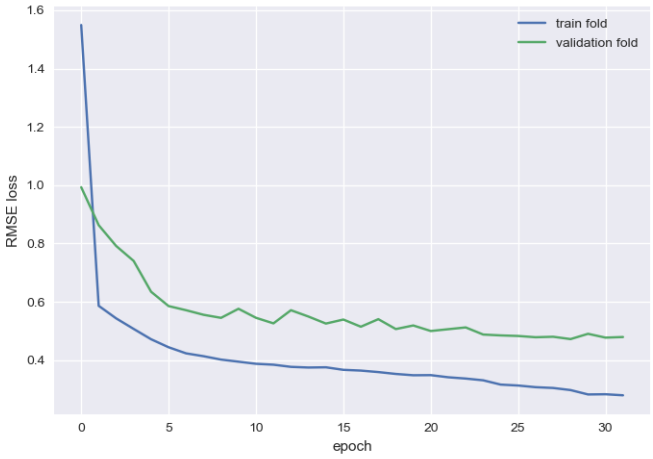


Fig. 4. Train and validation loss curves.

4 Results

All four models described in Section 3 were evaluated on the test set using MAE loss. The benchmark model achieves 0.56 MAE loss, the alternative benchmark 0.49, the SVR 0.46, and the LSTM 0.43. Table 1 shows the p-values of a two-tailed T-test for each pair of models. A difference is considered significant when this p-value is smaller than 0.05.

The alternative benchmark performs better than the original benchmark but not significantly, according to the p-value. Despite SVR outperforming both benchmarks, and the LSTM outperforming all three, the T-test shows that the differences between the performances of the SVR and LSTM, and the SVR and benchmark are not significant. Only the difference between the LSTM and the benchmark model is significant (p-value$<$ 0.05).

models	**p-value**
benchmark vs. alternative benchmark	0.122
benchmark vs. standard	0.074
benchmark vs. temporal	**0.004**
alternative benchmark vs. standard	0.737
alternative benchmark vs. temporal	0.116
standard vs. temporal	0.458

Table 1. p-values of T-test between the benchmark, average, SVM, and LSTM models, with bold letters indicating significance.

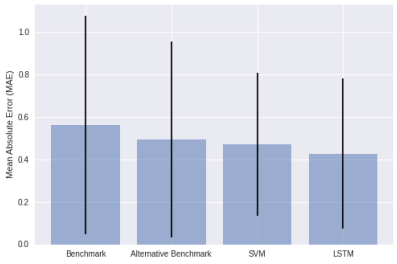


Fig. 5. Test score and uncertainty for the benchmark, alternative benchmark, standard and temporal models

5 Discussion

5.1 Test Results

The LSTM obtaining the lowest MAE loss, followed by the SVM is as expected, however, we anticipated a more significant improvement between the standard model and the benchmark model. We attribute this to the small size of the data set for two reasons; it causes low statistical power in terms of sample size and variance. The small test would require a large difference in performance between to models to be able to make any valid conclusions. Moreover, due to the small size of the test set, the scores display a high standard deviation between the individual test losses for each model. This standard deviation is illustrated as an uncertainty interval in Figure 5.

Furthermore, the small data set also diminished the predictive power of the models. A small data set consequently means a small training set. Therefore, it might be possible that the training set was too small for the models to be able to train sufficiently. Hence, this high epistemic uncertainty could be reduced by using more data.

This is a fairly difficult task, given that there are many contributing factors to one's mood. Hence, the low significance could also be attributed to the potential presence of aleatoric uncertainty in the data itself.

5.1.1 Model Comparison. While both models are able to handle non-linear data, the advantage of the LSTM model is that it is well-suited for time-series data, making it appropriate for the task at hand. Moreover, it does not require feature selection, as the model is able to learn the importance of the individual features and their combinations. On the other hand, while the SVR model is not able to learn sequential information, it is able to perform relatively well with little data. Despite the data being sparse, it is relatively unlikely to overfit, which is a point of caution for more complex models, such as an LSTM. Hence,

the SVR is effective in cases where the number of features is greater than data instances.

5.2 Quality of Data Set

Alhassan et al. [1] have found a positive correlation between smartphone addiction and depression. Therefore, it was expected that the `screen`, `activity`, and `appCat` variables would correlated highly with `mood`. However, according to the the plot in Figure 3, there is very low correlation between `screen` and `mood`. This discrepancy could be possibly attributed to poor description of the variable or the quality of the measurement.

Overall, the provided data set was shown to be sparse and very limited - removing the outliers yielded less than 500 instances to train and test our models.

5.3 Further Investigation

We recognise a few points of improvement with regards to our feature selection process in our experiment. It would be desirable to use more scientific literature to guide the feature selection process. Moreover, it would have been preferable to plot the correlation between part-of-day, missing-flags, user ID, and `mood` to detect the possible significance of these features, as opposed to performing tuning.

Unfortunately, the part-of-day features did not seem significant for the SVR model. As an additional feature, it could be interesting to also test how using day-of-week information would affects the model's performance, as people generally tend to have better mood in the weekends [3].

To further improve the performance of the LSTM model, we could use a more fine-grained data set, where the entries in the window consist of entries for each timestamp rather than whole days. Also, on top of using an early stopping mechanism to prevent overfitting, the model could benefit from using a form of regularisation, such as adding a L1 or L2 penalty to the loss function,

Lastly, to gain more insights in the decisions made by the LSTM model, it may be beneficial to use interpretability methods such as LIME or SHAP, which are able to do an analysis on the internal components of the model and their interactions. This would, for instance, enable detection of the most important features for predicting the mood. Alternatively, a self-explaining neural network could provide similar information. Moreover, these models are optimised for transparency, which might be a desirable trait as this could increase the certainty of the model being reliable.

References

1. Alhassan, A.A., Alqadhib, E.M., Taha, N.W., Alahmari, R.A., Salam, M., Almutairi, A.F.: The relationship between addiction to smartphone usage and depression among adults: a cross sectional study. BMC psychiatry **18(1), 148** (2018)

2. Caldeira, C., Chen, Y., Chan, L., Pham, V., Chen, Y., Zheng, K.: Mobile apps for mood tracking: an analysis of features and user reviews. In: AMIA Annual Symposium Proceedings. vol. 2017, p. 495. American Medical Informatics Association (2017)
3. Egloff, B., Tausch, A., Kohlmann, C.W., Krohne, H.: Relationships between time of day, day of the week, and positive mood: Exploring the role of the mood measure. Motivation and Emotion **19**, 99–110 (01 1995). https://doi.org/10.1007/BF02250565
4. Faurholt-Jepsen, M., Bauer, M., Kessing, L.V.: Smartphone-based objective monitoring in bipolar disorder: status and considerations. International journal of bipolar disorders **6**(1), 1–7 (2018)
5. Kim, J., Lim, S., Min, Y.H., Shin, Y.W., Lee, B., Sohn, G., Jung, K.H., Lee, J.H., Son, B.H., Ahn, S.H., Shin, S.Y., Lee, J.W.: Depression screening using daily mental-health ratings from a smartphone application for breast cancer patients. Journal of medical Internet research **18(8)** (2016)
6. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014)
7. LiKamWa, R., Liu, Y., Lane, N.D., Zhong, L.: Moodscope: Building a mood sensor from smartphone usage patterns. In: Proceeding of the 11th annual international conference on Mobile systems, applications, and services. pp. 389–402 (2013)