Data Mining Techniques

Report for Advanced Assignment 1

Group: 144

Johannes Janke (2738866), Dorota Mężyńska (2749593), Neža Šmid (2740786)

1 Introduction

In this study, we analyse whether the mood of a person can be predicted using smartphone data by evaluating the suitability of various models used for time series forecasting. In the domain of mental health, the analysis and prediction of mood potentially provide meaningful insights for depression treatments. As artificial intelligence becomes an increasingly important tool in the area of psychology research and treatment, this study aims to contribute by constructing a model which is able to predict the average daily mood of the user on the next day. The predictions are based on the previously obtained data including the record of smartphone applications which frequently ask the user for a rating of the mood as well as the tracking of the usage of other applications installed.
Traditionally, time series forecasts are based on linear, regressive models that are able to process historical time series data sets. The increasing emergence of machine learning algorithms leads to the fact that these algorithms are also applied in the context of time series forecasting. These instance-based learners require skillful processing of time series data to make predictions about specific points in time in the future. The first part of this thesis, therefore, focuses on the elaboration of a feature engineering model and the application of temporal data mining techniques. In the second part, the prepared data set is used to train a random forest (RF) and a support vector machine (SVM) model. To make a comparison to the machine learning models, a classical ARIMA, as well as a deep learning model (LSTM), are trained, which can process time series without prior feature engineering. The time-series predictions of all trained models regarding the target variable mood will be evaluated afterwards.

2 Pre-process the dataset

The underlying raw dataset captures entries from 27 different users over the period from 2014-02-17 to 2014-06-09, reflecting user data from 113 consecutive days. To prepare the dataset for the upcoming supervised machine learning approach, expressions of the 'variable' column are presented in individual columns with their associated values on a daily basis. Thus, these expressions can be analyzed as possible predictive features. To generate daily data, the average over all users of the features ‘mood’,

‘circumplex.arousal’, ‘circumplex.valence’ is calculated and the remaining features are summed over all users.

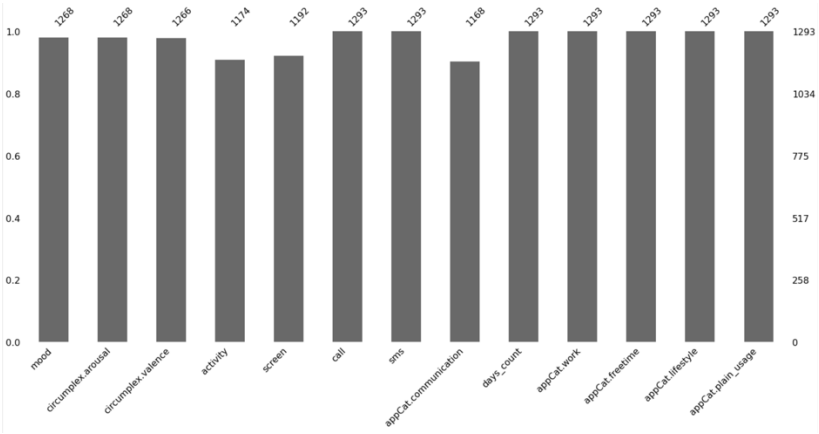
Now the features can be examined for possible errors, inconsistencies, and missing values to obtain information about the data quality. Descriptive statistics are generated for all features, with min, max, and limit values of the quantiles providing numerical information about the distribution of the values. An analysis of these individual distributions shows that the feature 'appCat.builtin' has invalid negative values. Consequently, one negative value is identified and replaced by NaN. Furthermore, the visualization of the value distribution of the features with the help of histograms shows that there is no normal distribution, and some features are positively skewed.
Also, the dataset consists of multiple missing values among the users over time. More precisely, the dataset lacks several measurements for the mood of the users as well as for the smartphone related data. Especially for the smartphone usage related features many missing values exist. The high number of missing values for these features states a problem in terms of their suitability as predictive features for the target variable mood. In order to compensate for the missing values of the smartphone usage related attributes, we decided to follow a feature combination approach which aggregates the single smartphone applications based on their general functionality and purpose. Géron (2019) states in his work that the combination of attributes is helpful to get more information out of the single attributes while the combination also helps to enrich the data as the impact of single missing attributes is diluted.

Table 1. Combining different attributes into new summed features

Combined attributes	**Single attributes**
appCat.work	appCat.office; appCat.finance
appCat.freetime	appCat.game; appCat.entertainment; appCat.social
appCat.lifestyle	appCat.travel; appCat.utilities; appCat.weather
appCat.plain_usage	appCat.other; appCat.unknown; appCat.builtin

The missing values of the forecasting target variable mood demand careful treatment. Anava et al. (2015) emphasize that especially when predicting time-series of data, missing values have a meaningful impact on the predictive algorithm. We decide to drop user data when there are no values for mood for two consecutive days due to the fact that these grouped missing values potentially add noise to this time-sensitive variable when performing aggregation and sequencing over time. For the missing values among the binary variables, we decide to impute them by the most frequent value, applying a traditional imputation method for missing values of categorical variables. After this effort of treating the observed outliers, we still see some missing values among features which are shown in the following bar chart.

Fig. 1. Overview of missing values across variables



For the remaining missing values in our data, we decide not to drop any further observations so that we do not minimize our potential training set and can add more value to the model. In line with Breda et al. (2016), we impute the missing values by the mean of each continuous feature within the observations of each user so that we get an accurate user-specific imputation. After doing this we have a dataset of 1291 observations with 97 different days over the period from 2014-03-04 to 2014-06-08. This dataset reflects the data that will be processed for our temporal forecasting algorithms.

For the Random Forest and Support Vector Machine model, we need to further process the data so that these algorithms are able to perform time-series predictions. In the given case it is of high importance to transform and re-frame the time series data to make use of our desired machine learning algorithms who typically learn based on single instances. Brownlee (2017) highlights the creation of lag features and window features in order to prepare multivariate time series data for the underlying machine learning problem. For the target variable 'mood' we decide to shift the time series for each user so that we can use this lagged feature as the desired prediction target. The goal of adding a rolling window statistic to our dataset is to induce some memory of prior time steps in our desired instance-based dataset. In order to choose the perfect length of the sliding window, we compute different aggregation windows which look back from two to five days. We check the correlation of these aggregation windows with our lagged target variable and obtain the following results, ranked descending by correlation.

Table 2. Top 10 highest correlated variables to the target variable

Correlation to target = 'mood' next day					
Rank	**Feature**	**Correlation**	**Rank**	**Feature**	**Correlation**
1	target	1.00	6	circumplex.valence_2dayavg	0.57
2	mood_2dayavg	0.86	7	circumplex.valence_3dayavg	0.51
3	mood_3dayavg	0.78	8	mood	0.47
4	mood_4dayavg	0.73	9	circumplex.valence_4dayavg	0.45
5	mood_5dayavg	0.70	10	circumplex.valence_5dayavg	0.42

Based on these results we decide to take a lookback window of two days for all features while taking care of the difference in daily timestamps across the users. For the final feature selection, we decide to drop the feature measuring the duration of the plain app usage (appCat.plain_usage) and the feature that indicates if a call is made by the user (call_2dayavg) because their correlation to the target variable is close to zero, smaller than plus/minus 0.01. Due to the creation of sliding windows as well as of the lagged target variable, we end up having additional NaNs in our data which are dropped. Thereby, the feature engineering dataset consists of 1237 observations with 95 different days over the period from 2014-03-06 to 2014-06-08.
Due to the differences in scale of the given features, we decided to apply MinMaxScaler method, in order to translate the values of each feature to the range between zero and one. We decided to not scale our target variable for the purpose of better interpretation of the predictions. This choice is considered when setting up the different alghoritms.

3 Models

The models discussed below are general, which means they were estimated using observations from all users. While we are aware that individual modelling for each user could result in models with better performance, there is still the issue of data availability regarding training such models, since for every user it would have to be split into a training and test set (LiKamWa et al., 2013). Consequently, we decided to focus on predicting the mood of users without distinguishing between individuals, as we believe this approach is more likely to provide relevant insights for the research of the smartphone usage influence on mood.

3.1 Feature Engineered models: SVM and Random Forest regression

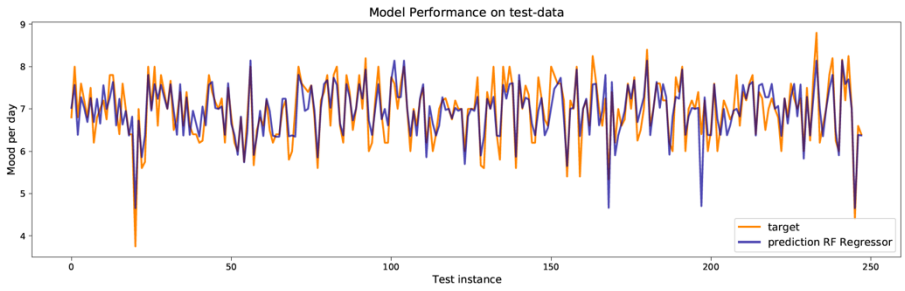
Based on the methods widely used in the literature related to smartphone usage and mood prediction (e.g. van Breda et al., 2016; Garcia-Ceja et al., 2018) the following machine learning approaches were chosen: the Random Forest (RF) and Support Vector Machines (SVM). Both are examples of supervised machine learning, which is an appropriate choice given that we have a target variable available in the dataset, i.e.

mood, and the purpose of the analysis, which is to predict the average mood of the user on the next day based on previous days. Also, both models are non-parametric, meaning there are not a lot of assumptions about the data required, which allows for fitting the training data best while constructing the mapping function. Furthermore, the data was split into a train and test set for validating the models. The proportion of the data set included in the test split was set to 0.2, which is a common approach in such analysis. Moreover, both models are tested on the same train and test sets.

Random Forest

The RF adds to the decision tree model, aiming to mitigate some of its downsides, with one of them being severe over-fitting. It seeks to do that by creating multiple independent decision trees and, in the case of regression, outputs the average prediction of all single trees obtained by majority voting. During the analysis, the GridSearchCV method was used to help determine the optimal hyperparameter values for the model, which resulted in the following: max_depth= 3, min_samples_split= 27. The focus was on those two hyperparameters since we believe they are crucial to help prevent overfitting and thanks to this, it is also less computationally expensive. Moreover, different values of the n_estimators hyperparameter were checked and the value which gave the best results out-of-sample was 800. The bootstrap parameter was set to True, which means that each decision tree is trained on a different random sample taken from the dataset with the purpose of reducing the correlation between them.

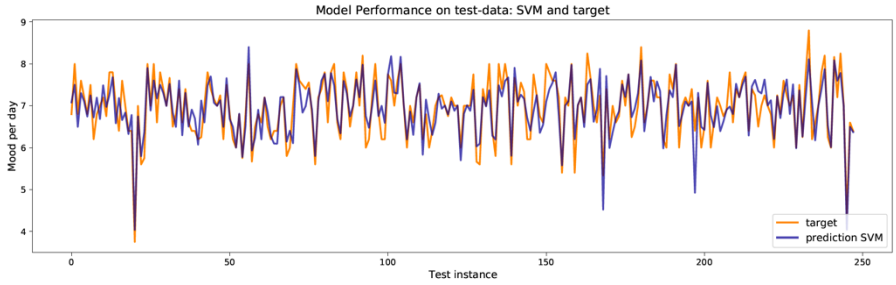
Fig. 2. Random Forest Regressor performance against target on unseen test data.



Support Vector Machine

The SVM is an algorithm that creates hyperplanes in a multidimensional space with the goal of either classification or regression (van Breda et al., 2016). In contrast to RF, the model is not scale-invariant, which means data scaling is recommended, which was done during the data preprocessing stage. Once again, to determine the optimal hyperparameter values the GridSearchCV method was used and the following hyperparameter values were estimated: C=10, gamma=1, kernel='linear '.

Fig. 3. Support Vector Machine regressor performance against target on unseen test data.



3.2 Temporal models

In comparison to the feature engineered models discussed before, temporal models were also used in research to compare, which model performed better on differently preprocessed datasets. Since temporal models take into consideration order of the time series data, they require continuous and sequential daily data, to learn from the historic data accordingly (Bremer et al., 2019). First, we used the "classical" approach with the ARIMA model (Autoregressive integrated moving average model) and the second model we used was the RNN model (Recurrent neural network model). The two models were selected because they are complementary, where the ARIMA model can only deal with the stationary data and the RNN model can also deal with trends and seasonality in the datasets. Also, the performance of the multivariate RNN model will be interesting to compare to the univariate ARIMA model.

ARIMA model

ARIMA models are temporal structured models that can be used for forecasting approaches. ARIMA operates on time series and applies a linear regression-based forecast. Because of the need for stationary data, we ran a Dickey-Fuller test on the mood variable on one user (as a representative for the whole dataset), to check for stationarity. The P-value of the test is 0.000017, which means the data is stationary. To determine the number of lags for AR and MA in ARIMA we plotted autocorrelation (ACF) and partial autocorrelation (PACF). Based on the two plots (figure 3) we test our dataset on the ARIMA (2,1,2) model.

Fig. 4. ACF and PACF for the ARIMA model

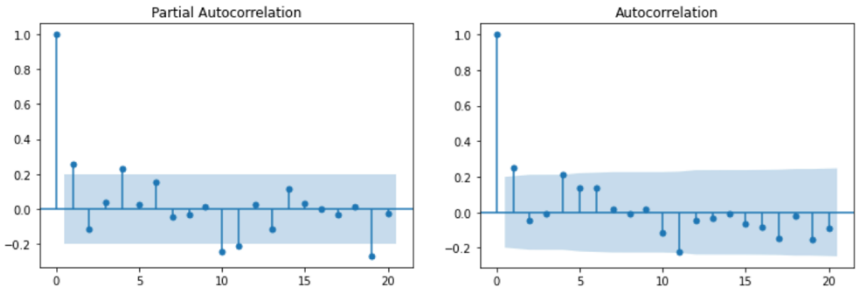
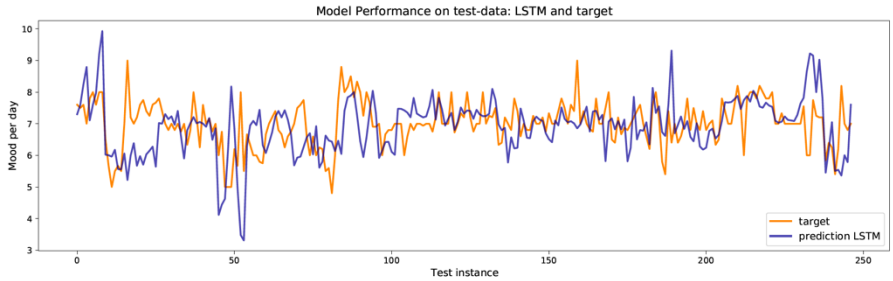


Fig. 5. ARIMA one-step ahead forecast.

LSTM model

We use the long-short-short term memory model (LSTM) as RNN for our time series forecast as this model is useful for identifying potential long-term dependencies in the data. The main characteristic of this deep learning model is that it learns based on memorized time series sequences in the data. We test different lengths of sequences namely three, five and eight, and conclude that a sequence length of three yields the best performance. For each user, we split the time series into 80% training and 20% test time series for all features, while ensuring that the amount of test data is enough to capture a test sequence of three. The prediction target is set to be the mood of the day after the three days sequence allowing for a one-step-ahead forecast of overall dates in the test set. Our LSTM model has two layers and uses the Rectified Linear Unit (ReLu) as an activation function which provides a model that is relatively easy to train and does not require special scaling of data so that we can leave our target variable at the original scales and only normalize the features. By doing so, we can easily compare the performance of the LSTM to the other trained algorithms. The output also uses the ReLu as an activation function. We compile our LSTM model by using Adam as an optimizer and mean squared errors between actual values and predictions as loss function.

Fig. 6. LSTM performance against the target (mood) on unseen test data.



3.3 Benchmark model

As a comparison to the previously established models, which deal with the instance-based dataset (SVM, RF) and time series dataset (ARIMA, LSTM), the benchmark model was constructed to compare the performance of each model. In the paper, by Asselbergs et al. (2016) there are three different benchmark methods used. One is the naive benchmark model, and the second used model is the history benchmark model. The history model in the paper, used the mood of today, as a predictive power of the mood of tomorrow (Asselbergs et al., 2016). In our research, we used the history model as our benchmark model. We constructed the "unintelligent" model, based on a history benchmark explained by Asselbergs et al. In our case, we used the mood of "today" that we already knew and calculated the first lag of the mood variable (if the average mood of today was 6.3, the predicted mood of tomorrow will be 6.3) per user. Also, the benchmark model is a general model, for all users.

Fig. 7. Insert graph of the performance of benchmark vs the target

4 Evaluation of results

To evaluate the performance of the above-mentioned models, the following measures are used: mean squared error (MSE), mean absolute error (MAE), and Mean Absolute Percentage Error (MAPE). Since we deal with regression algorithms, the prediction

values are continuous, and the measure of performance needs to be able to handle that. The MSE, MAE and MAPE are given by the following equations:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \qquad MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{|y_i|} \tag{1}$$

The values cannot be interpreted by themselves, but only by comparing the results obtained for different models based on the same data and forecast period. Models with low values of MSE, MAE or MAPE are considered desirable since this means that the model is good at predicting the values of the target variable. The MSE is a measure of the variance of the residuals, while MAE gives the average error value and MAPE is a measure of accuracy in percentages. One of the disadvantages of MSE is that it tends to exaggerate the impact of outliers compared to MAE. The MAPE is easy to interpret since it can be seen as a percentage error, however, if the predictions can take on both positive and negative values it can become problematic (Brooks, 2019).

The models were evaluated using both the train and test sets, to be aware of the changes in prediction performance for both while tuning the hyperparameters. It is important, since, e.g., good performance in-sample accompanied by low-performance out-of-sample is an indication of overfitting. In the table below are presented the evaluation metrics for the test samples.

Table 3. Comparison of the different metrics of the used models

	Feature Engineered models		*Temporal models*		Benchmark
	SVM	RF	ARIMA	LSTM	
MSE	0.12914	0.12999	0.20132	0.95259	0.55275
MAE	0.27221	0.27312	0.32333	0.69206	0.54404
MAPE	0.04003	0.04052	0.04907	0.10015	0.08148

4.1 Results: statistics

The RF and SVM, as was already mentioned, are supervised machine learning models, with the ability to model complex data. Because of this, the evaluation metrics are expected to be better than for other models. Based on the performance results presented in the table above, that is indeed the case. Both feature engineered models perform better than any temporal model or historic benchmark across all metrics. The best performing model is SVM, however the RF is only slightly worse at predicting the mood. This is in line with the findings of van Breda et al. (2016), where they also found that both SVM and RF outperform their benchmark model while trying to predict mood of individual users. In case of temporal models, ARIMA performs significantly better than LSTM across all metrics. Moreover, while the LSTM model is worse than the benchmark, ARIMA outperforms it. The MAPE of this model is also quite close to the results of feature engineered models.

4.2 Results: interpretation

The first conclusion that could be drawn is that aggregating the data during pre-processing stage seems to be an important step to achieving better predictions, as the models which used the instance-based dataset perform the best. This also underlines the importance of the data preparation step in the overall data mining process and the advantages of temporal data mining. The fact that ARIMA, based on the MAPE measure, seems to perform almost as good as both RF and SVM could be perhaps explained by the analysis of feature importance. This was done for both models through permutation and the results show that the most relevant feature while predicting is the average mood over the last 2 days. This would also consequently suggest, that all the other features, i.e. smartphone application usage by app category, have little predictive power when it comes to forecasting the mood of all users in general. Furthermore, this is also supported by the model performance results of LSTM when compared to ARIMA, as the model performs the worst out of all analysed.

4.3 Pros and Cons of different used Models

Both approaches have their upsides and downsides. The major difference between the feature engineering and temporal models lies in the data preprocessing step, since for the former the data needs to be aggregated. While this is a quite time-consuming process, the results show that it does help obtain better performing models. In general, in favour of the RF model speaks the fact that it is suited for modelling complex relationships and multiple regimes, as well as its flexibility however, there is always a risk of overfitting the model on training data. Furthermore, the results might be difficult to interpret due to the introduction of the random component during sampling. Similarly to the previous model, SVM is also best suited for complex relations, however apart from flexibility, its other advantage is computability, which is the opposite of RF. The issue of over-fitting is also present in case of SVM, as well as the issue of interpretability (Chakraborty and Joseph, 2017). When it comes to temporal models, one significant disadvantage of ARIMA is that it only works for stationary data, while LSTM is able to account for seasonality and trends. Another shortcoming of the ARIMA model is that it only considers lagged values of one variable, whereas LSTM takes all features into consideration.

5 References

1. Anava, O., Hazan, E., & Zeevi, A.J. (2015). Online Time Series Prediction with Missing Data. ICML.
2. Asselbergs, J., Ruwaard, J., Ejdys, M., Schrader, N., Sijbrandij, M., and Riper, H. (2016). Mobile phone-based unobtrusive ecological momentary assessment of day-to-day mood: an explorative study. J. Med. Internet Res. 18, e72.
3. Bremer, V., Becker, D., Genz, T., Funk, B., Lehr, D. (2019). A Two-Step Approach for the Prediction of Mood Levels Based on Diary Data. In. Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2018. Lecture Notes in Computer Science, vol 11053. Springer, Cham.
4. van Breda, W., Pastor, J., Hoogendoorn, M., Ruwaard, J., Asselbergs, J., & Riper, H. (2016). Exploring and Comparing Machine Learning Approaches for Predicting Mood Over Time. Innovation in Medicine and Healthcare 2016, 37–47.
5. Brooks, C. (2019). Introductory Econometrics for Finance (4th ed.). Cambridge University Press.
6. Brownlee, J. (2017). Introduction to time series forecasting with Python: How to prepare data and develop models to predict the future. Machine Learning Mastery.
7. Chakraborty, C., & Joseph, A. (2017). Machine Learning at Central Banks. SSRN Electronic Journal.
8. Garcia-Ceja, E., Riegler, M., Nordgreen, T., Jakobsen, P., Oedegaard, K. J., & Tørresen, J. (2018). Mental health monitoring with multimodal sensing and machine learning: A survey. Pervasive and Mobile Computing, 51, 1–26.
9. Géron, A. (2019). Hands-on machine learning with scikit-learn, Keras, and TensorFlow (2nd ed.).
10. LiKamWa, R., Liu, Y., Lane, N. D., & Zhong, L. (2013). MoodScope. Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services - MobiSys '13.