Data Mining Techniques Assignment 1

Laurens Hakkesteegt (2577269, lht620), Cathy Tol (2567806, ctl240) & Janine van Wonderen (2573337, jwn203)

1 Introduction

Smartphone applications are becoming more and more available to support people suffering from depression. These applications record all kinds of sensory data about the behaviour of the user and ask the user for a rating of the mood. With the use of the data from this application, a model can be build that is able to predict the average mood of the user on the next day. Two approaches are considered to create a predictive model. The first one is with the use of a learning algorithm approach that can deal with temporal data, such as an ARIMA model. The second way is using temporal abstractions of features and applying a machine learning algorithm which works on instance level data.

2 Data Analysis and pre-processing of the data

2.1 The data

A dataset is given and this dataset originates from the domain of mental health. Smartphone applications are becoming available to support people suffering a depression. These applications record all kinds of sensory data about the behaviour of the user and in addition frequently ask the user for a rating of the mood. The dataset consist of four columns. The first column contains the ID of the user of the application. The second column contains the timestamp of when the event was saved in the data. Multiple things can be saved in the dataset and the third column "Variable" shows the type of variable that is saved in the row. There are 19 different possible variables. The fourth column contains the value.

For all 19 variables some insights were gained to get a clear overview what the variables represent and whether there are some strange values. Table 1 shows for all variables the average value, the minimum value, the maximum value, the standard deviation, the 25th percentile, the 75th percentile and the median. With this information it can be seen whether the data has unreliable values, and this is the case. Something that is remarkable from table 1, is the fact that variables 'appCat.builtin' and 'appCat.entertainment' have a negative value for the minimum value of the variable. Both these variables represent a duration of an application and a duration can never be negative. So, these values do not seem right and the negative values for the variables that represent a duration should be deleted from the dataset. The rest seems reasonable. For some values the average differs a lot from the median and is not even within the range of the 25th and 75th percentile, this is the case for example for appCat.entertainment. This is

Table 1: The variables and their interpretation

Variable	**Average**	**Minimum**	**Maximum**	**Standard Deviation**	**25th Perc**	**75th Perc**	**Median**
mood	6.993	1	10	1.033	7	8	7
circumplex.arousal	-0.099	-2	2	1.052	-1	1	0
circumplex.valence	0.688	-2	2	0.671	0	1	1
activity	0.116	0	1	0.187	0	0.158	0.022
screen	75.335	0.035	9867.007	253.822	5.322	62.540	20.044
call	1	1	1	0	1	1	1
sms	1	1	1	0	1	1	1
appCat.builtin	18.538	-82798.871	33960.246	415.989	2.020	9.922	4.038
appCat.communication	43.344	0.006	9830.777	128.913	5.218	45.476	16.226
appCat.entertainment	37.576	-0.011	32148.677	262.960	1.334	14.922	3.391
appCat.finance	21.755	0.131	355.513	39.218	4.072	20.155	8.026
appCat.game	128.392	1.003	5491.793	327.145	14.148	123.625	43.168
appCat.office	22.579	0.003	32708.818	449.601	2.004	8.044	3.106
appCat.other	25.811	0.014	3892.038	112.781	7.019	16.829	10.028
appCat.social	72.402	0.094	30000.906	261.552	9.030	75.372	28.466
appCat.travel	45.731	0.080	10452.615	246.109	5.087	47.227	18.144
appCat.unknown	45.553	0.111	2239.937	119.400	5.018	44.431	17.190
appCat.utilities	18.538	0.246	1802.649	60.959	3.159	19.331	8.030
appCat.weather	20.149	1.003	344.863	24.943	8.684	25.349	15.117

due to the fact that the variables contain extreme values and this influences the average. It is assumed that this is correct.

A lot of figures were made, which are not included in this paper. Instead some noticeable findings are mentioned here. A piechart of the values of the mood showed that almost half of the time the mood 7 is filled in. Mood 6 and 8 also contribute for a big part to the dataset. A plot of the average mood during the hours of the day showed two peeks. One around 4 and one around 7. However, a plot of the number of records of the mood during the hours of the day showed that the number of records on the times of those peeks is very low. This means that only a few people have filled in the moods during this time, so the moods that were filled in have a lot of influence of the average mood. So, overall can be concluded that time of the day does not have a big influence on the mood. A plot of the number of calls and text messages over the hours of the day showed that, during the night, they were very low. Additionally, the number of calls is much higher than the number of text messages. They both have a peek around 17. A plot of the average time of screen of over the hours of the day showed a peek around 7. A plot of the average activity over the hours of the day showed a peek around 16. The aforementioned plots did not show the same pattern. So when activity is high, the average time of screen is relatively seen not high.

2.2 Transformations

Creating a condensed dataset First, for all the attributes the sum and/or mean were taken per day. The original dataset often contains multiple data points for features on one day. Since those data points are not all at the same time on a day it is not possible to compare those easily. It is decided to take the mean of the features mood, circumplex.arousal, circumplex.valence and activity and the sum of all the other features per day. From this data a new dataset is created which has date and id as index and the new means and/or sums of features as columns. A weekday variable was also added since it seems that this might also affect the mood, this variable ranged from 1 to 7.

Adding temporal abstractions After this temporal abstractions could be made. It was decided to create 3 kind of abstractions. The first one is taking the mean for the last x days before an instance for each feature. With x ranging from 1 till 5. So for each feature 5 new features are created which were called "feature name_time_x" were x is 1 till 5. For all these variables the correlation with the target variable, the mood mean of the next day, is calculated. The second abstraction is a counter which counts for each mood how long it has been that the mood of the user has been so high. So for example if it has been 20 days since the last time the mood has been higher or equal than the current value, then this variable will show 20. This variable is made for each mood number for each user. The last abstraction is a counter for how often a certain mood has been seen in the last 5 days. The 5 days was chosen because this seemed like a realistic time frame to hold on to a mood.

Cleaning the data It happens a lot that there is missing data. However, most of the missing data is activity information. Because no activity equals zero for all activity related features, the NaN values were replaced with zero for those features. For each row, the target value needed to be added. This is the mood mean of the next day. All variables are normalized, this is done to get all the variables on the same scale. This is important for distance based algorithms and if regularization with a shrinkage quantity is used. It also allows machine learning algorithms to converge gradient descents quicker.

Inspecting new dataset visually After the dataset has been transformed, some visualizations of correlations between features are made. Figure 1 left shows the mood mean target against the normalized average mood mean of the previous 2 days. A trend line is added in this figure. As can be seen in figure 1, the trend line is increasing which indicates a positive relation. There are also variables that do not have a relation with the mood mean target. An example of this is the variable the normalized average appCat builtin of the previous 5 days. Figure 1 right shows the mood mean target against the normalized average appCat builtin of the previous 5 days. As can be seen in the figure, the trend line is flat and this thus represents that there is no relation between the target variable and the variable.

3 Models

3.1 Benchmark

A benchmark is implemented to validate the results. As a benchmark the mood on the next day is predicted by just saying it is the same as the previous day.

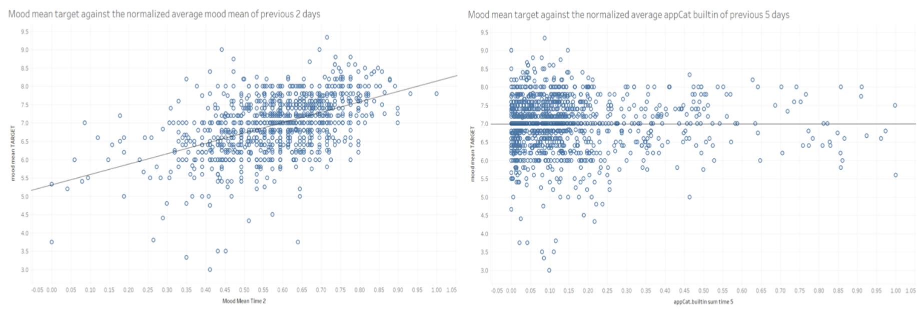

Fig. 1. Left: Mood mean target against normalized mood mean of the previous 2 days **Right**: Mood mean target against the normalized appCat builtin of previous 5 days.

3.2 Recurrent Neural Network

The next model that is considered is a recurrent neural network as explained by Lipton, Zachary in 2015 in the paper 'A Critical Review of Recurrent Neural Networks for Sequence Learning' [2].

First, a recurrent neural network is created which uses only 1 variable. In this case only the mood mean is used. Later, the model is adapted to be able to work with multiple variables. The idea is to predict the next mood using the previous X moods. The python deep learning library Keras is used.

The values for the mood mean are normalized and the non-normalized values of the mood mean must be predicted. Therefore, for the input values, the normalized mood means are used and for the target values the later added value mood mean TARGET is used.

To find the best number of previous time steps to use, which is called the look back value, the train set as explained in evaluation of the models, is again split in train en test set in the same way. For a number of look back values, the model is trained and 'tested'. The results for the model using one variable and two variables can be seen in table 2.

It can be seen that the results for the model using one variable while using as look back variable 5 performs best. Thus 5 is chosen as the look back variable.

Table 2: The mean squared error and R^2 for different look back values of the RNN model using one variable and two variables.

look_back	**MSE_1**	R^2**_1**	**MSE_2**	R^2**_2**
1	0.580	0.059	1.247	-1.024
2	0.561	0.092	0.581	0.060
3	0.405	0.183	0.397	0.200
4	0.294	0.049	0.334	-0.078
5	0.239	0.233	0.272	0.126

3.3 ARIMA

The ARIMA model is a combination of an autoregressive and a moving average model and can be used on difference data [1]. The model has three parameters, namely p, d and q. The parameter p represents "the number of autoregressive terms", the parameter d represents "the number of non-seasonal differences" and the parameter q represents "number of lagged forecast errors in the prediction equation"[4]. The package statsmodel is used to create an ARIMA model. The model predicts for every row of the test set separately the mood mean of the next day. The mood mean of the previous days of the person are given as train data for the ARIMA model to make the prediction.

Leave-one-out cross validation is used on the train set to determine the most optimal value for the parameters. Table 3 shows the mean squared error and the R^2 for different combinations of the parameters. It is preferred to have the mean squared error to be small and the R^2 close to 1. However, this is not the case for this situation. Therefore, the combination of parameters where there is a balance between the mean squared error and the R^2 is chosen. Thus, the for this case is chosen for the final ARIMA model to have the parameters (p, d, q) being equal to (1,1,0).

Table 3: The mean squared error and R^2 for different states of the ARIMA(p, d, q) model.

(p, d, q)	**MSE**	R^2	**(p, d, q)**	**MSE**	R^2
(0, 0, 0)	0.420	-1.425	(0, 2, 1)	0.717	-0.056
(0, 0, 1)	0.433	-1.288	(1, 0, 0)	0.409	-1.198
(0, 1, 0)	0.622	-0.124	(1, 1, 0)	0.540	-0.178
(0, 1, 1)	0.475	-0.263	(1, 2, 0)	1.112	-0.020
(0, 2, 0)	1.828	-0.022	(2, 1, 0)	0.572	-0.257

3.4 Linear Regression

The machine learning technique that is used to predict the mood is a linear regression model. Although this is a "simple" technique it performed best from the tried machine learning techniques. The machine learning techniques we tried are a neural network, boosted gradient boosted regressor and support vector regressor. The linear regression method corresponds to minimizing the sum of square differences between the observed and predicted values. Scientific literature shows that linear regression can be used for temporal abstractions.[3] The hard part in making this model is finding the best features to use. The performance metric that we are trying to maximize is the R squared (R^2). If R^2 is 1 it means that a model is able to perfectly predict the target variable. Two ways of finding the best features were used. First the function RFECV from the sklearn package. This function tries to find the best combination of features to maximize a certain score while using cross validation. The second way was by looking at the correlations. We added a manual threshold for the correlation a feature should have with the target, by shifting this threshold we could find the optimal combination. The final features are shown in table 4, those features were chosen with a correlation threshold of 0.25.

Table 4: Final linear regresion features with weights

Feature	**Weight**
numberOfTimesSeenMood5Days_6	-0.301
numberOfTimesSeenMood5Days_4	-0.237
mood_mean	0.846
mood_mean_time_2	1.610
mood_mean_time_3	-1.840
mood_mean_time_4	0.374
mood_mean_time_5	1.779

3.5 Evaluation of the models

The models are evaluated with a test set. This test set is constructed of the 12 last dates of each ID. The total size of this test set is 326 rows, this corresponds to 26% of the total dataset. Taking the last 12 of each ID was done such that the RNN and ARIMA models can also be properly tested on completely new data. All the models were tuned on the training data before the they were evaluated on the test data. The metrics that we looked at were the R^2 score and the Mean Square Error (MSE).

3.6 Illustrate performance with graphs

For each of the tried methods two figures are generated shown in figure 2. The first one shows the predicted mood mean of the model and the real mood mean of the evaluation set. The second one shows the differences between the predicted and the real mood mean. It can be seen that for all models, when looking at the first figures, the lines have similarities. The blue line follows the orange line. However, in extreme cases the blue line does not go as far as the orange line. When looking at the second figures, it can be seen that the line is centered around 0 with some fluctuations. This confirms what is seen in the first figures. The models are good in predicting the average flow but are not that good in predicting extreme values. When comparing figure 2.2 and figure of 2.4, it can be seen that figure 2.4 shows less fluctuations with smaller ranges. Thus the RNN has better performance. When comparing figure 2.4 and figure 2.6, it is seen that figure 2.4 shows less fluctuations but the average height of the fluctuations in figure 2.8 is smaller. Finally, when comparing figure 2.8 with figures 2.4 and 2.6, the height of the fluctuations in figure 2.8 are the smallest. Thus the overall performance of the linear regression model seems best.

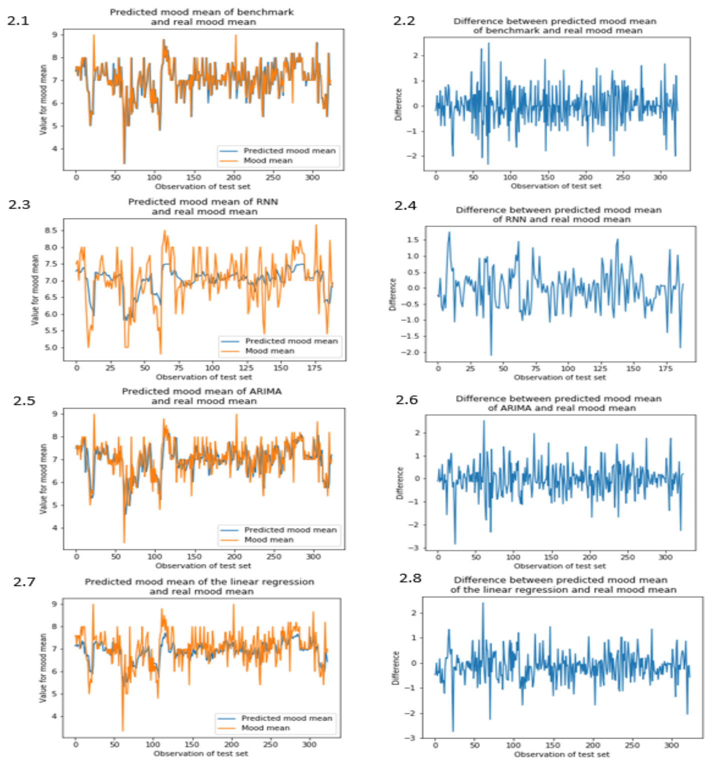

Fig. 2. The predicted and real mood mean and the difference between them for all methods.

4 Results

4.1 Analyze results using statistics

The models are all evaluated on an evaluation set. The chosen important metrics are R^2 and MSE. This are also the metrics the models are optimized for. Table 5 shows the results for the various models. The linear regression model has the highest R^2 while having the lowest MSE. This might be because the linear regression model is able to use the extra temporal abstractions which indicate how many times a certain mood is seen the last 5 days. The ARIMA and RNN model are less suited to find features like that. Table 5 also shows the mean and standard deviation of the difference between the predicted and the target value. The values between the models are very similar, however, the RNN and linear regression score best.

4.2 Analyze results by interpretation

Since the RNN, ARIMA and linear regression models are more sophisticated models than the benchmark, it seems logical that these results are better. Be-

Table 5: Results on the evaluation set.

Model	**MSE**	R^2	**Mean difference**	**Standard deviation difference**
Benchmark	0.490	0.105	0.513	0.475
RNN	0.365	0.340	0.464	0.387
ARIMA	0.441	-0.063	0.486	0.453
Linear Regression	0.361	0.351	0.444	0.405

cause we are dealing with temporal data, it was expected that RNN and ARIMA would work better. This is not the case. Probably because the data was not that periodic that the RNN and ARIMA models could learn a specific behaviour. Due to the fact that the linear regression model takes multiple features, including temporal abstractions, into account, linear regression outperforms the other models.

4.3 Pros and cons of different approaches

The pros of the RNN and ARIMA model are that they can work with a relatively low amount of feature engineering because the models do not need added temporal abstractions. The con of the RNN is that it is hard to understand what exactly happens (blackbox). The con of the ARIMA model is that it cannot model interactions between features. The pro of a linear regression model is the simplicity of understanding of the model because of the weights it outputs. It is possible to exactly see which features get more weight. The con of a linear regression model is that the temporal data needs to be created in this case.

References

1. Heiberger, R., & Holland, B.: Statistical analysis and data display : An intermediate course with examples in r 2nd edn. Springer, New York (2015).
2. Lipton, Zachary. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning.
3. Palm, F. C. and T. E. Nijman. 1982. "Linear Regression Using Both Temporally Aggregated and Temporally Disaggregated Data." Journal of Econometrics 19(2):333–43.
4. Introduction to ARIMA: nonseasonal models, https://people.duke.edu/ rnau/411diff.htm. Last accessed 18 Apr 2019