Assignment 1 - Mood Prediction Using Smartphone Data

Lieve Jilesen[ljn278], Ryan Ott[rot280], and Jaime Perez y Perez[jpz240]

Group 17

In this paper we will discuss our approach to assignment 1 of the Data Mining Techniques course. We will first discuss Data Preparation of the advanced dataset, including exploratory analysis, data cleaning and feature engineering. We will then discuss our approach to classification which yielded poor results. We will then briefly discuss Association Rules. Followed by our approaches to numerical prediction. Our code is available on GitHub https://github.com/Ryan-Ott/DMT-Group17

1 Data Preparation

1.1 Exploratory Data Analysis

	Variable	Data Type	Total Records	Unique Values	Missing Values	Mean	Std	Min	25%	50%	75%	Max
0	mood	float64	5641	10	0	6.992555	1.032769	1.000	7.00000	7.000000	8.000000	10.000
1	circumplex.arousal	float64	5597	5	46	-0.098624	1.051868	-2.000	-1.00000	0.000000	1.000000	2.000
2	circumplex.valence	float64	5487	5	156	0.687808	0.671298	-2.000	0.00000	1.000000	1.000000	2.000
3	activity	float64	22965	1868	0	0.115958	0.186946	0.000	0.00000	0.027739	0.158333	1.000
4	screen	float64	96578	64646	0	75.335206	253.822497	0.035	5.32225	20.044500	62.540250	9867.007
5	call	float64	5239	1	0	1.000000	0.000000	1.000	1.00000	1.000000	1.000000	1.000
6	sms	float64	1798	1	0	1.000000	0.000000	1.000	1.00000	1.000000	1.000000	1.000
7	appCat.builtin	float64	91288	23237	0	18.538262	415.989243	-82798.871	2.02000	4.038000	9.922000	33960.246
8	appCat.communication	float64	74276	39890	0	43.343792	128.912750	0.006	5.21800	16.225500	45.475750	9830.777
9	appCat.entertainment	float64	27125	12575	0	37.576480	262.960476	-0.011	1.33400	3.391000	14.922000	32148.677
10	appCat.finance	float64	939	729	0	21.755251	39.218361	0.131	4.07200	8.026000	20.155000	355.513
11	appCat.game	float64	813	792	0	128.391615	327.145246	1.003	14.14800	43.168000	123.625000	5491.793
12	appCat.office	float64	5642	3178	0	22.578892	449.601382	0.003	2.00400	3.106000	8.043750	32708.818
13	appCat.other	float64	7650	5014	0	25.810839	112.781355	0.014	7.01900	10.028000	16.829250	3892.038
14	appCat.social	float64	19145	14660	0	72.401906	261.551846	0.094	9.03000	28.466000	75.372000	30000.906
15	appCat.travel	float64	2846	2606	0	45.730850	246.109307	0.080	5.08650	18.144000	47.227250	10452.615
16	appCat.unknown	float64	939	846	0	45.553006	119.400405	0.111	5.01800	17.190000	44.430500	2239.937
17	appCat.utilities	float64	2487	1732	0	18.537552	60.959134	0.246	3.15850	8.030000	19.331000	1802.649
18	appCat.weather	float64	255	250	0	20.148714	24.943431	1.003	8.68400	15.117000	25.349000	344.863

Fig. 1. Overview of data

There are 376912 records in the dataset, 27 participants, 113 unique days, and 19 variables. The data set consists of the mood, arousal, valence, screen time, activity, calls and texts of the user. Additionally, the dataset also contains several variables within one column, which denote several app categories that the user may spent time on (such as communication, travel, builtin, weather, etc.). The range of values present for each variable can be seen in Figure 1. Furthermore, there is some noise present in the dataset. Notably, within the variables of arousal and valence, there are a significant amount of missing values. When analyzed further as in Figure 2, it can be seen that the majority of these missing values occur in just a few participants. First, a heat map was constructed for the correlations between all variables, but as not many correlations between single variables were significant, it was chosen to only portray the correlation

heat map for the target variable mood. Additionally, there are negative values for variables that are measured in time, which in this case should not be possible. Furthermore, there are some significant outliers in various variables, such as in the variable that holds built-in apps. Additionally, the distributions of each variable was calculated, however, most of the variables were found to not follow a normal distribution, and were mostly heavily skewed with lower values occuring most frequently. The variables of mood, arousal and valence as the exceptions. Mood is centered around the value of 7, as can be seen in Figure 4.

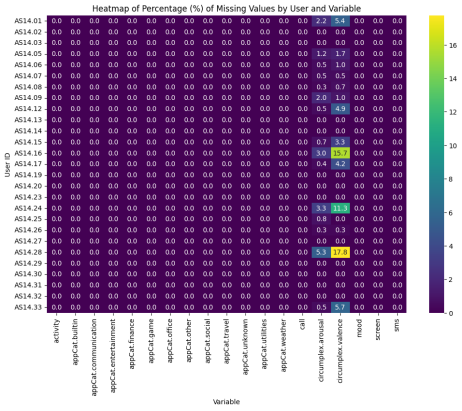

Fig. 2. Heatmap of missing values according to percentage

Additionally, as can be seen in Figure 5, there is some variance in total time the data was collected in between ids, this is especially notable in the plot through the graph getting more scarce in the beginning and end of the plot.

1.2 Data Cleaning

First, all variables with time as a measure were analyzed to see if they have any negative values. These values were then removed from the dataset. This resulted in 3 values being removed from the builtin apps, and one value being removed from entertainment apps. For numeric variables that are not self-reported by the participants (screen and all app variables), we remove values that lie outside of the interquartile range (IQR). To do this, we first calculate the IQR, which is the difference between the 75th percentile (Q3) and the 25th percentile (Q1) of the data. Values that fall below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR are considered outliers and are thus excluded from the analysis to prevent them from skewing the results. This method is suited was chosen since this method does not assume an underlying normal distribution (unlike for example, the z-score method), as most variables in this dataset are not normally distributed. In total, 35465 outliers were removed using IQR, with the majority coming from screen time and builtin apps.

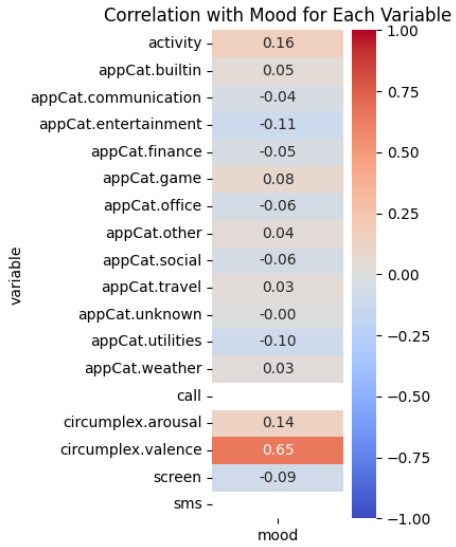

Fig. 3. Correlation Heatmap of Mood

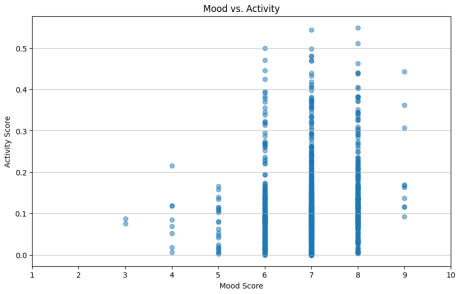

Fig. 4. Mood vs Activity

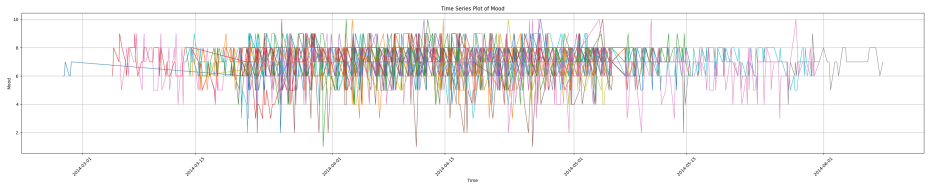

Fig. 5. Time Series plot of Mood per id

Two common approaches for imputing missing values in time series data are:

1. Mode: for a given user and variable, the most common value for that variable is used to impute the missing values.
2. Linear Interpolation: this method assumes that the change between two data points is linear and imputes the values accordingly.

We will apply both methods and then discuss which one might be more suitable. Overall, mode imputation replaces every missing value of a variable by the mode of the known values [1]. However, while it provides a rough estimate, and is a fast and simple method, using mode may introduce significant bias to the data and is therefore more suited in cases where not many values are missing [5]. In contrast, linear interpolation approximates the value of data points by considering the values of the neighboring points in a one-dimensional data sequence [4].

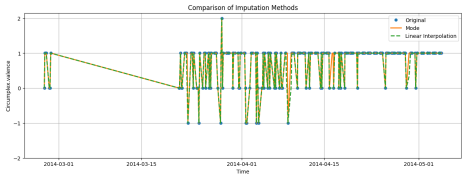

Fig. 6. Imputation comparison of User 1

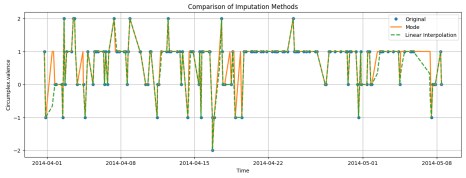

Fig. 7. Imputation comparison of User 2

Based on the above plots (Figure 6 and 7), we can see that both methods impute NaN values to fill gaps in our data. While the mode method is more conservative and keeps the original shape of the data, the linear interpolation method introduces values that are not necessarily seen in our dataset (e.g. fractional values), but creates smoother transitions between readings. In the interest of maintaining the original shape of the data, we will use the mode method to impute missing values.

1.3 Feature Engineering

Feature Engineering for LSTM Before constructing a predictive model using long short term memory (LSTM), we must first consider feature engineering as a means to improve performance. The feature engineering for the LSTM model is relatively simple, as the sequence of the values of the mood as reported by the users is taken, and zero padding is applied in order to create sequences of equal length, as is required by the tensorflow LSTM architecture. These zeros are ignored in the masking layer of the LSTM and thus should not have any effect on the final output of the LSTM.

Feature Engineering for the Random Forest Classifier For the Random Forest Classifier, the history of a day is aggregated in order to predict the mood for the next day. We did not use a larger aggregation history, as we recognized that the mood of a day is best predicted by the values closest to that day and any long term effects of previous days would most likely be represented in the values of the day before. Additionally, values in the data that have similar meaning were combined, e.g. the variables 'appCat.entertainment', 'appCat.game' and 'appCat.social', as the usage of these apps have a similar improvement of the mood. Therefore, it was decided to treat them as the same category. In this way, all 'appCat' variables are divided into three buckets: 'entertainment', 'util' and 'other'. This reduces model dimensionality and will therefore hopefully improve performance. For further transformations, the mean of the mood, circumplex.arousal and circumplex.valence variables, and the sum of the other variables were calculated.

The omission of certain values can also be a predictor for the mood of the user. However, we can not include this in the dataset. In addition, this means that some user-day pairs do not have all variables included in the dataset. Merging multiple variables reduced this problem, but it still is a significant issue in the dataset. We therefore employed a strategy of imputation by assigning dummy values of -10 to signify missing variables. This approach was considered appropriate given the expectation of positive values for all variables. We anticipated that the Random Forest classifier will effectively discern and accommodate these placeholder values, potentially finding them as meaningful indicators of mood states. Finally, we apply a StandardScaler normalizer to all variables used.

2 Classification

2.1 Application of Classification Algorithms

For both approaches that we will discuss we took as target the mood of the next day. We defined the mood of the next day as the rounded average of all the reported mood values for the next day.

LSTM For the LSTM neural network we created a model which for the first layer has a masking layer which allows the model to ignore the zero padding applied in the feature engineering. This layer is then followed up with the LSTM layer using tanh as its activation function. This layer is then followed up by a dens layer of ten nodes using a softmax activation layer. These ten nodes represent the ten possible labels that can be represented, i.e. 1-10. When predicting values using this model one should feed a sequence of mood values to the model which is padded out with zeros until the the max sequence length of the training data which is 6. So the the sequence [3,4,5] should be padded to [0,0,0,3,4,5]. The model returns a probability distribution over the possible label, the label with the highest probability should be chosen. This model yielded an accuracy of 0.54 on the test set

Random Forest We created a Random Forest model using the sklearn implementation of it optimizing its hyperparameters using gridsearch. The parameters under test were the number of estimators, the maximum number of features per estimator, the max depth of a decision tree estimator and the splitting criterion. The resulting model yielded an accuracy of 0.56 on the test set.

Evaluation As can be determined from the accuracy scores of these models, their performance is very poor. Due to the distribution of the labels of one can create a very simple model with an accuracy of 0.50 by predicting every label as a 7, which is by far the majority class of the labels. The next two biggest classes are 6 and 8.

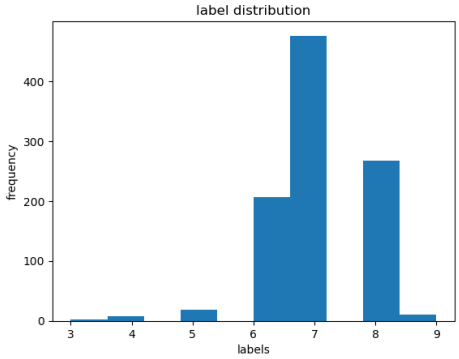

Fig. 8. The only classes with a substantial amount of labels are 6,7 and 8.

The trained LSTM and Random Forest models are only slightly better then this simple baseline. We think this poor performance is due to the algorithms not being able to really distinguish between the majority classes 6,7 and 8 as their variables might be very similar, with the additional issue that different

people will self report different values whilst having the same/similar values for certain predictors. This would make it hard for the classifiers to get a good accuracy. For the Random Forest classifier we also checked whether or not the performance had anything to do with our way of using dummy values to encode missing values. To test this hypothesis, we also ran our Random Forest classifier on a dataset which only used the samples which had all variables. However, this classifier had similar performance. Also, we note that we could make the classifier user dependant, i.e. include the id variable as one of the predictors. This might eliminate some of the user created noise in the dataset. However, one drawback of this approach is that it would make the model useless for new users not yet included. Due to time constraints we have not tested this approach.

2.2 Winning Classification Algorithms

For this competition[1], participants had to develop a model which could distinguish AI-generated text from essays written by middle and high school students. The competition was held between October 31st 2023 and January 22nd 2024. Its dataset consisted of around 10,000 essays, some written by students, and some generated by a variety of large language models (LLMs).

Each essay written by students was written in response to one of seven essay prompts. It is uncertain if the exact same input prompts were used for generating the LLM essays.

The training set consists of essays in response to merely two prompts, with the rest of the essays forming the hidden test set. The vast majority of the test-set provided was written by students instead of generated, however, participants in the competition were encouraged to generate more essays to use as training data. Submissions were evaluated on area under the ROC curve between the predicted probability and the observed target.

The Public leaderboard is calculated with approximately 46% of the test data, whereas the final leaderboard is calculated with 54% of the test data. For this task, we will analyze the top 1 approach from each leaderboard. We have chosen this because while the Public leaderboard winner additionally scored a top 10 position within the final leaderboard, while also providing speculation on how their code achieved this. The final leader does not provide much context in how their ranking was achieved for their models, but claim their performance is mainly based on the extension method of the dataset. However, both winners of the leaderboards seem to agree on one thing: focusing on diversity of the dataset as a core component for a well-performing model.

The Public leaderboard winners speculated that their win can mainly be contributed to the reverse-engineered pipeline, their carefully curated datasets with diversity in both prompts and models, and to their cluster based post-processing. Specifically, while most other competitors used a simple TFIDF pipeline, their pipeline uses a TFIDF with several classifiers on top of the TFIDF features,

[1] `https://www.kaggle.com/competitions/llm-detect-ai-generated-text/discussion/470255`

and combines it with a BERT based pipeline for classifying both human and generated texts. Additionally, a reverse engineered approach was used in order to correct systematic spelling mistakes. For post-processing, the TFIDF pipeline was used in order to compare pairs of texts with high similarity, which often proved to be LLM generated.

3 Association Rules

In order to group products into higher level categories, we can use (hierarchical) clustering methods based on features. Depending on which higher level categories are appropriate for the use-case, relevant features such as ingredients, can be extracted from each data entry and then analyzed for similarity based on these features. Products can be clustered by "*type, shape, occasion, materials, features, price, style, design, colour, size, family, brand, function, and more*" [2]. According to similarity scores, products can be clustered and these clusters can then be analyzed, create a distinct class for each available cluster, and then assign all points in the cluster to the respective class.

This approach has the benefit of flexibility in use-cases, as the chosen features on which the clustering is based can easily be adapted based on what provides better performance. For example, if feature A B, and C form better prediction together as opposed to feature A, B, and D, then one might go with the former in order to maximize profits. In other words, the variety in possible combinations of chosen attributes is a significant advantage of grouping products. Additionally, hierarchical structuring can aid in interpreting the connections between product categories by providing oversight in classes and their respective sub-classes.

However, depending on the size of the dataset and the chosen similarity metric, hierarchical clustering can be computationally intensive. Another downside of hierarchical clustering is that a metric needs to be defined on how to separate the clusters, there might be products that fit into multiple clusters or products that fit into none. Therefore, rules need to be specified in order to handle these cases. Additionally, as hierarchy between clusters is taken into account, it could potentially be challenging to determine the metrics for how hierarchy between clusters is handled, or to specify the optimal total amount of clusters.

4 Numerical Prediction

For predicting the mood of a participant as a numerical target rather than a class, we can make use of regression models, which instead of picking the class with the highest probability of fitting our data, predict the most likely value for the variable of interest. The two approaches chosen were **Decision Tree Regressor** and **Linear Regression**.

Firstly, a Decision Tree Regressor constructs a decision tree model to predict the value of a target variable - in our case a participant's mood - by learning simple decision rules. Decision trees are non-parametric supervised models, meaning

[2] https://retalon.com/blog/product-clustering

that the number of parameters is not specified ahead of time, but determined by the data, and that they require labeled data to train on, allowing a model to be learned that can then be used to infer the value for new, unseen data. The fact that they are non-parametric makes these models in theory well suited to capturing non-linear relationships between variables, without the need for extensive feature engineering [2]. Furthermore, these models are highly interpretable due to their flow-chart like nature and as such allow simple insights to be gained into the decision-making process, which is beneficial in psychology/medical contexts like mood prediction.

In linear regression, data are characterized by linear predictor functions, with model parameters being inferred from the available data. Specifically, linear regression typically entails modeling the conditional mean of the response variable Y given the predictor variable X as an affine function of X [3].

For our implementations specifically we used the 'sklearn' Python package. The hyperparameters to tune for the Decision Tree Regressor would include the maximal depth of the tree, the minimum number of samples required to split a node as well as the minimum number of samples a leaf must have. Through fine-tuning these, a balance between under and over-fitting can be achieved. Using the sklearn model selection functionality grid search over suitable hyperparameters could be conducted to systematically discover optimal settings.

For evaluation, both the MSE and MAE metrics serve as indicators of how well our models perform. The results can be seen in section 5.

The difference between these approaches is the decision tree uses a tree structure of nodes with decision rules, whereas regression predicts a numerical value and uses an assumed linear relationship between features and the target variable. However, both approaches use optimization techniques to minimize a cost function, such as gradient descent for linear regression. Furthermore, their evaluation metrics are similar, as they are both able to use MSE and MAE in order to assess performance.

5 Evaluation

5.1 Characteristics of Evaluation Metrics

Mean Squared Error (MSE)
The MSE measures the average of the squares of the errors; the difference between the estimated values and the actual value.

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

Where Y_i is the actual value, $\hat{Y}_i$ is the predicted value and n is the number of observations.

Mean Absolute Error (MAE)
The MAE measures the average of the absolute differences between predicted values and actual values, providing a linear score that reflects the average magnitude of the errors in a set of predictions.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|$$

Reasons to Use MSE over MAE

As MSE squares the errors before averaging, a relatively high weight is given to large errors. This characteristic means that MSE is useful when large errors are particularly undesirable and should therefore be penalized more heavily. Furthermore, the squaring of each term allows MSE to be differentiable, facilitating analytical treatment and making it easier to calculate gradients for optimization algorithms. These algorithms are especially important in the field of machine learning, particularly in regression models, where its also commonly known as L_2 loss.

Reasons to Use MAE over MSE
An advantage of MAE is that unlike MSE, MAE does not square the error terms, making it less sensitive to outliers. This property is useful in cases where data can be volatile and outliers should not heavily influence the model performance metric. Additionally, MAE is often easier to interpret as it is measured in the same units as the data, therefore making it easier to understand the magnitude of the errors directly.

Example Situation: Identical Results for MSE and MAE
Consider a dataset where all predictions $\hat{Y}_i$ are consistently off by a small and constant error ϵ from the actual values Y_i, and there are no outliers or large deviations. For instance, if $\epsilon = 1$ for every prediction relative to the actual value, i.e., $|Y_i - \hat{Y}_i| = 1$ for all i. For example, in the case of $Y_i = 5$ and $\hat{Y}_i = 4$. We then have the following:

$$\text{MSE error } = \frac{1}{n}\sum_{i=1}^{n}(5-4)^2$$

$$\text{MAE error } = \frac{1}{n}\sum_{i=1}^{n}|5-4|$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(1)^2 = 1, \quad \text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|1| = 1$$

Here, both MSE and MAE would compute to 1, giving identical results because the constant error does not benefit from the squaring in the MSE formula nor

suffer from the absolute value in the MAE formula. The identical results stem from the uniformity of the error across all predictions, which neutralizes the impact of squaring versus taking absolute values.

5.2 Impact of Evaluation Metrics

Applying the MSE metric to our decision tree regression model of task 4 yields a result of 0.97. Applying the MAE metric to our decision tree regression model yields a result of 0.77. When applying the same metrics to our linear regression model we get a MSE of 0.53 and a MAE of 0.58. From these values we can conclude that the linear regression model is preferable over the decision tree model as the errors are lower. Another sign that the linear regression model is quite good is the fact that for this model the MSE is smaller then the MAE, which is unexpected as the MSE score squares the errors which usually results in larger scores. However, when the error becomes smaller than 1 then squaring the error will actually reduce this value. Therefore, we can conclude for the linear regression model a lot more of the errors are less than 1 when compared to the errors of the decision tree regressor.

References

1. Batista, G.E., Monard, M.C.: An analysis of four missing data treatment methods for supervised learning. Applied artificial intelligence **17**(5-6), 519–533 (2003)
2. Breiman, L.: Classification and regression trees. Routledge (2017)
3. Groß, J.: Linear regression, vol. 175. Springer Science & Business Media (2003)
4. Huang, G.: Missing data filling method based on linear interpolation and lightgbm. In: Journal of Physics: Conference Series. vol. 1754, p. 012187. IOP Publishing (2021)
5. Zhang, Z.: Missing data imputation: focusing on single imputation. Annals of translational medicine **4**(1) (2016)