Predicting Mood from Smartphone Data

Group 100

Urban Sirca, Matus Halak, and Kieran Keesmaat
2834709 2724858 2843427

Vrije Universiteit, Amsterdam 1081 HJ, Netherlands

1 INTRODUCTION

Mobile phone and app usage data has been gaining awareness as a means to monitor various health attributes and outcomes of individuals. One particular area of interest is in the mental health sector, where individuals may rate their mood throughout a day while other apps monitor and gather behavioral data, such as screen usage and even physical activity. This development offers the opportunity to mine these data for personalized insights and interventions.

In this assignment, we explore this idea, utilizing a dataset consisting of smartphone-derived behavioral and self-reported information to try and predict a users next-day mood. By following the CRISP-DM process model, we begin with data understanding, cleaning and feature engineering to identify factors important in predicting next-day mood. Subsequently we apply machine learning techniques for both classification and regression tasks. The aim is to develop a successful predictive model for next-day mood that is valuable for the users.

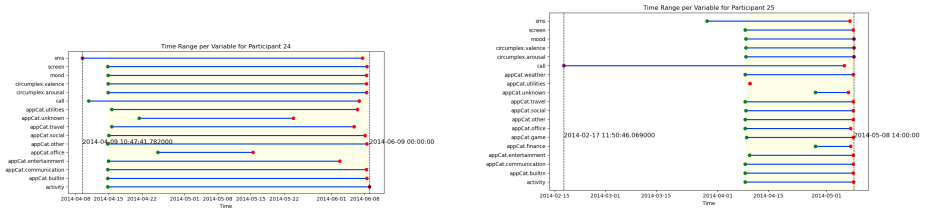
2 DATA PREPARATION

2.1 EXPLORATORY DATA ANALYSIS

This assignment used the "dataset_mood_smartphone.csv" dataset, which consisted of 376,912 records obtained from 27 unique users. Descriptive statistics for the dataset are found in Table 1. The dataset contained 19 attributes, with twelve being time spent using a specific type of app in seconds (builtin, communication, entertainment, finance, game, office, other, social, travel, unknown, utilities, weather). The remaining 7 attributes were: self-reported mood (the target variable, ranging from 1 and 10), arousal and valence (psychological metrics on a scale from –2 to 2), physical activity (normalized between 0 and 1), screen time (in seconds), calls made and texts (SMS) sent (both encoded as 1).

The time range of records ran from 17/02/2014 until 09/06/2014; however, this was inconsistent for data both within and between participants (Figure 1).

Upon inspecting the raw dataset, eight values were identified as potential outliers due to unusually high usage durations exceeding 20,000 seconds (5.6 hours). However, these records were retained, as such durations were deemed plausible. High usage in app categories such as builtin, entertainment, social, and office can reasonably reflect valid user behavior. Namely, long default browser or media player usage (builtin), extended streaming sessions (entertainment), social



(a) Records Range for Participant 24 (b) Records Range for Participant 25

Fig. 1: Time Range of Records for Participants 24 and 25

media engagement (social), or remote work-related tasks (office) may naturally span several hours. The maximum observed value of approximately 9 hours still falls within a realistic range for heavy or continuous usage patterns. Further

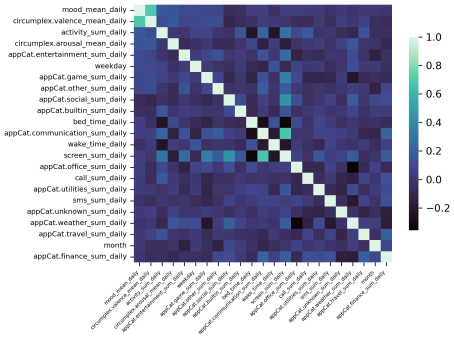
Table 1: Descriptive Statistics of Raw Data

variable	count	mean	std	min	median	max
activity	22965	0.12	0.19	0.00	0.02	1.00
appCat.builtin	91288	18.54	415.99	-82798.87	4.04	33960.25
appCat.communication	74276	43.34	128.91	0.01	16.23	9830.78
appCat.entertainment	27125	37.58	262.96	-0.01	3.39	32148.68
appCat.finance	939	21.75	39.22	0.13	8.03	355.51
appCat.game	813	128.39	327.14	1.00	43.17	5491.79
appCat.office	5642	22.58	449.60	0.00	3.11	32708.82
appCat.other	7650	25.81	112.78	0.01	10.03	3892.04
appCat.social	19145	72.40	261.55	0.09	28.47	30000.91
appCat.travel	2846	45.73	246.11	0.08	18.14	10452.61
appCat.unknown	939	45.55	119.40	0.11	17.19	2239.94
appCat.utilities	2487	18.54	60.96	0.25	8.03	1802.65
appCat.weather	255	20.15	24.94	1.00	15.12	344.86
call	5239	1.00	0.00	1.00	1.00	1.00
circumplex.arousal	5597	-0.10	1.05	-2.00	0.00	2.00
circumplex.valence	5487	0.69	0.67	-2.00	1.00	2.00
mood	5641	6.99	1.03	1.00	7.00	10.00
screen	96578	75.33	253.82	0.04	20.04	9867.01
sms	1798	1.00	0.00	1.00	1.00	1.00

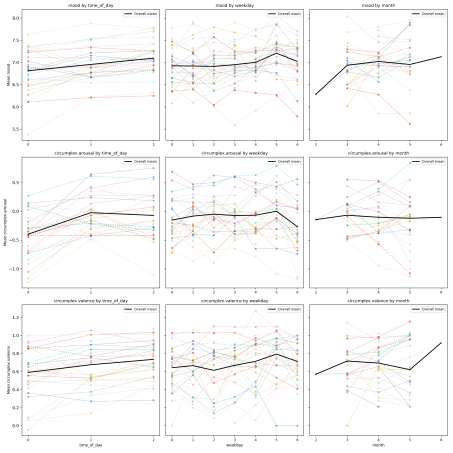
inconsistencies included 202 missing entries in arousal and valence, and four negative values found for app usage times. The negative usage times were removed due to their implausibility. Missing arousal and valence values were also removed. Since most days included multiple valid mood assessments and data were later aggregated at the daily (and subdaily) level(s), isolated missing entries were unlikely to bias the aggregated outcomes. Hence, these missing entries were safely discarded without requiring imputation at the raw data level.

Due to the time-series nature of the data, we explored temporal trends. Initial exploration showed that valence and arousal (often used as dimensions to describe mood according to the Circumplex model of emotion [1]) were among the

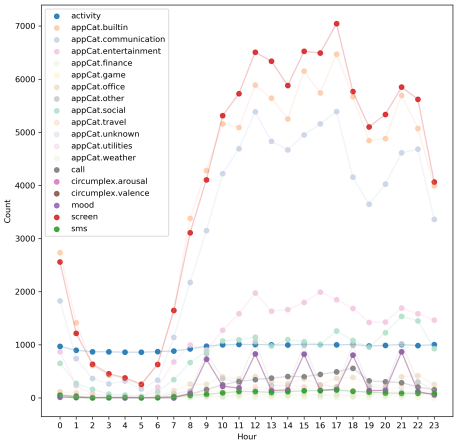
variables most correlated with mood (Fig 2a). Therefore, we explored variations in mood and circumplex variables across times of day, weekdays, and months for each participant (Fig 2b). We noticed an upward trajectory in the variables throughout the day from morning until evening. All three variables peaked on Saturday (weekday 5), although there was individual variation. Monthly trends seemed highly individual, which matched "month" as one of the initial variables least correlated with mood (Fig 2a). We next wondered, whether we could ex-



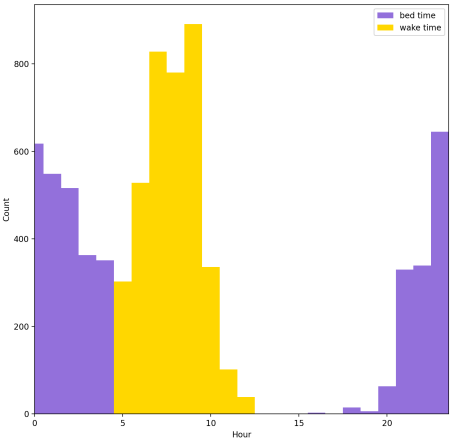
(a) Correlations of initial aggregated features. Sorted by correlation with mean daily mood.



(b) Mood & circumplex temporal trends



(c) Variable data-points by hour



(d) Distribution of wake and bed times

Fig. 2: Exploration of data and temporal patterns

tract sleep/wake information from the data. The distribution of data-points for each variable by hour (Fig 2c), showed that "activity" was the only feature that did not show a circadian rhythm and sent constant hourly data throughout the day and night. Data-points from other variables, due to their circadian patterns,

were assumed to reflect user activity in that hour. We extracted bed times as the last data-point between 6 PM and 5 AM the next day, and wake time as first data-point between 5 AM and 1 PM on the current day (excluding "activity" data-points). The distribution of wake and bed times was realistic (Fig 2d), and thus, these variables were included in further feature engineering. Apart from mood-correlations, sleep/wake variables negatively correlated with activity (third most mood-correlated initial feature) and total screen time (Fig 2a).

2.2 DATA CLEANING

As mentioned in the previous section, NAs and incorrect negative values were removed from the raw dataset, and potential outliers were kept in as they were deemed plausible. Furthermore, as mood was the target variable, records without corresponding mood values were deemed unable to contribute to model training or evaluation. Therefore, we did not consider data before and after the first and last mood measurement respectively for each participant. The area highlighted in yellow in Figure 1 shows the time range in which data was kept for each user.

Following this, the data was aggregated per day for all attributes to enable predictions of average next-day mood. This was done in different ways for each attribute to explore various representations of user behavior and emotion. The different aggregations used included the mean, std, max, min, sum, first and last (record for that attribute) for each day. For some attributes we applied multiple aggregations to explore potentially interesting aspects of variability within that attribute. For example, with valence, we applied almost all aggregations as each was thought to potentially offer something unique about a persons day that is predictive of their next-day mood. For other attributes, we only found one sensible aggregation, like summing daily number of SMS and calls. To capture finer-grained within-day fluctuations, mood, valence, and arousal were also aggregated using mean, minimum, and maximum values across three distinct time-of-day bins: morning (00:00–10:59), afternoon (11:00–17:59), and evening (18:00–23:59) and formed a separate "Time of Day" dataset (discussed in 2.3). An overview of all attributes and the aggregations applied to them can be found in Table 2. Our custom "Std.slide" aggregation, was a sliding standard deviation over all values from the current and previous 2 days. The idea was to provide a more accurate estimate of recent variability, still computed from the raw data.

After aggregation, we analyzed the resulting dataset and discovered many users were missing attribute values for some days. To address these missing values, we took the following imputation approach. For SMS, calls, and all app usage, missing values were imputed with 0. This decision was based on the assumption that missing entries reflected absence of activity. For example, a user not sending texts, making calls, or using a particular app category on a given day. This absence of data was deemed informative rather than random, and imputing with 0 appeared a reasonable and interpretable representation of actual behavior. For missing values in minimum and maximum aggregated attributes, mode imputation was used. This was done to retain data realism, because a mean or linear interpolation for discrete variables would lead to non-integer values that never actually occurred throughout the dataset. For the remaining attributes

Table 2: Aggregation Methods for Different Attributes

Aggregation	Attribute
Mean	Mood, Valence, Arousal
Std	Mood, Valence, Arousal
Std.slide	Mood, Valence, Arousal
Sum	Activity, Screen, Call, SMS, appCat (all)
Max	Activity, Mood, Valence, Arousal
Min	Mood, Valence, Arousal
First	Mood, Valence, Arousal
Last	Mood, Valence, Arousal
Time of Day	Mood, Valence, Arousal

(derived from mood, valence, arousal, screen time and sleep/wake variables) we first applied a mean imputation. However, we noticed that it destroyed clear temporal trends in the data. We then tried linear interpolation and decided to use it instead, as it maintained temporal trends throughout the data. This was particularly true for relatively long stretches of missing values. For missing time-of-day mood, valence, and arousal values, in the "Time-of-Day" dataset, linear interpolation was also used to capture within-day trends.

2.3 FEATURE ENGINEERING

We created two datasets - a "Time of Day" (ToD) dataset, for recurrent neural networks, with data for three time-points per day (as described above), designed to provide more training data for deep learning; and a "Daily" dataset, consisting of daily-aggregated training instances for classical non-temporal machine learning algorithms. A potential weakness of our approach is that the ToD dataset only contained ToD aggregations mood-related features, and otherwise contained the same daily features as the "Daily" dataset. Since the daily information was the same in all three ToD samples for each day, this could have introduced errors or inefficiencies into the deep learning models.

Overall, the feature engineering process involved 1) creation of initial features from the raw data (month, weekday, time of day, wake time, bed time), 2) aggregation of features on daily level and time of day level (described above), and 3) creation of new features from aggregated and imputed data. An overview of features we created from imputed data can be found in Table 3. For composite features, we combined two attributes to create one feature (calculating sleep duration from bed and wake times, or classifying emotion into quadrants based on the combinations of positive / negative valence and arousal). For lag features, we computed the change in a given attribute from it's previous measurement (for Daily attributes, this meant change from the previous day, for ToD attributes, it was change from previous time of day). Finally, to provide summary of recent history, we computed the exponentially weighed moving average (to emphasize recent events and impact on recent mood fluctuations) from previous seven measurements. We chose to keep training instances on daily level

(for classical ML models), supplemented with historical summary in this form for many variables, to maximize the number of training samples, instead of reducing training instances to several days and having less training data.

Table 3: Types of engineered features

Feature type	**Attributes used**	**Frequency**
Composite – Circumplex Quadrant	valence, arousal	Daily and ToD
Composite – Sleep	bed time, wake time	Daily
Lag – Change	mood, valence, arousal	Daily and ToD
Lag – Change	wake time, bed time, screen, activity	Daily
Summary – Exponentially Weighted MA	mood, valence, arousal	Daily and ToD
Summary – Exponentially Weighted MA	wake time, bed time, sleep, screen, activity	Daily

The pair-plot of final features most correlated with next day mood (target) for a selection of attributes is shown in Fig 3. Of note is that while Summary (mva7) and Lag (_change) features were among the most correlated with the target for multiple attributes, Composite features were not better than simple aggregations or Summary features of the underlying attributes (circumplex and wake time). Interestingly, the relationships between variables seem to differ between participants, which could be problematic for building a general model.

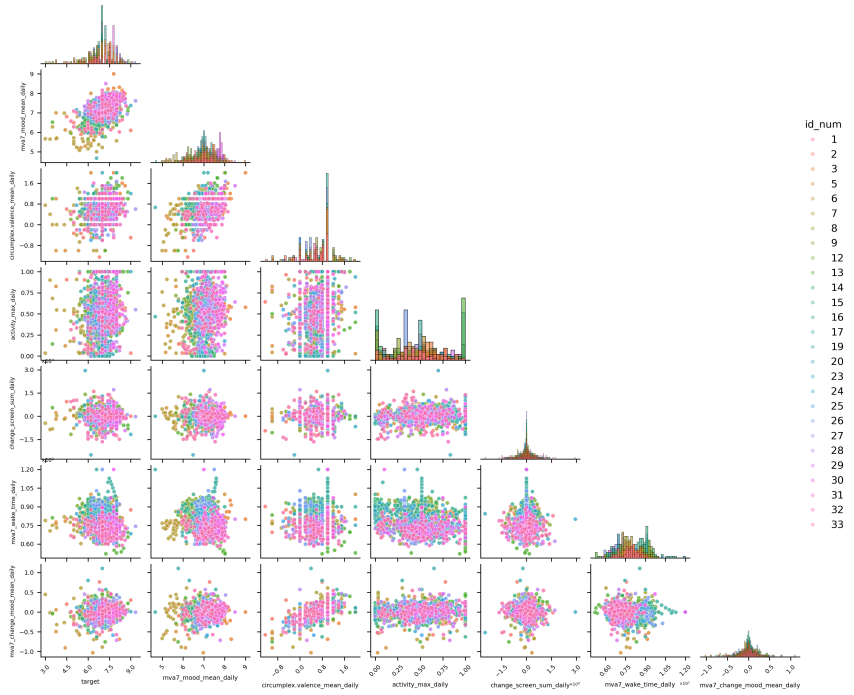


Fig. 3: Selection of engineered features (*participants by color*)

3 CLASSIFICATION

3.1 APPLICATION OF CLASSIFICATION ALGORITHMS

In this classification task, our goal was to predict next-day mood using smartphone derived behavioral and self-reported variables. To structure this as a classification problem, the continuous target variable mood_mean_daily was discretized into three mood categories where the model would predict whether mood increased or decreased significantly or stayed the same. This discretization was based on the change in next-day mood relative to a personalized threshold, defined as 0.7 times the user's 7-day moving average of daily mood standard deviation. Using a 7-day window provided a more stable estimate of recent mood variability compared to a single-day standard deviation. The scaling factor of 0.7 was chosen to produce a reasonably balanced distribution across the three classes, as higher thresholds resulted in most values falling into the "same" category. This allowed us to frame the task as a multiclass classification problem.

Feature-Based Classification We began by creating a baseline model using mood_mean_daily as the sole predictor. We then extended the model by testing two feature sets from the 'Daily' dataset: the top 35 most correlated features with the categorical target, and a full feature set. Initial testing revealed overfitting when using all features, thus we chose to proceed with the 35-feature 'Daily' dataset for the final evaluations. We tested a myriad of classification algorithms, including Logistic Regression (multiclass), Decision Tree, Random Forest, Gradient Boosting, Support Vector Classifier (SVC), and XGBoost Classifier. All models were trained using categorical cross-entropy loss, a standard choice for multiclass classification tasks [4]. This loss function measures the difference between the predicted class probabilities and the actual class label, encouraging the model to produce high probabilities for the correct class.

Hyperparameter tuning was performed for each model using grid search with 3-fold cross-validation. For example, for the XGBoost Classifier, our best-performing model, we searched across the parameter grid found in Table 4.

Table 4: XGBoost Classifier Hyperparameter Grid

Hyperparameter	Values
n_estimators	[500, 1000]
max_depth	[2, 3]
learning_rate	[0.005, 0.01]
subsample	[0.5, 0.7]
colsample_bytree	[0.5, 0.7]
gamma	[0.2, 0.5]
min_child_weight	[5, 10]
reg_alpha	[1.0, 5.0]
reg_lambda	[10.0, 50.0]
scale_pos_weight	[1]

The final dataset was split into a training set (1142 instances) and a test set (127 instances), with instances randomly shuffled before the split. During eval-

uation, we examined multiple performance metrics, such as accuracy, precision, recall, and F1-score, to give a comprehensive model assessment (see Table 5). The test and train accuracy for each of the models can be seen in Fig 4.

Table 5: Model Performance on Test and Train Set

Model	**Acc_test**	**Acc_train**	**Prec_test**	**Rec_test**	**F1_test**
XGBoost	0.543	0.647	0.546	0.543	0.538
Gradient Boosting	0.512	0.666	0.517	0.512	0.486
Decision Tree	0.488	0.495	0.520	0.488	0.476
Random Forest	0.480	0.632	0.491	0.480	0.473
Logistic Regression	0.417	0.528	0.433	0.417	0.401
SVC	0.394	0.514	0.410	0.394	0.381

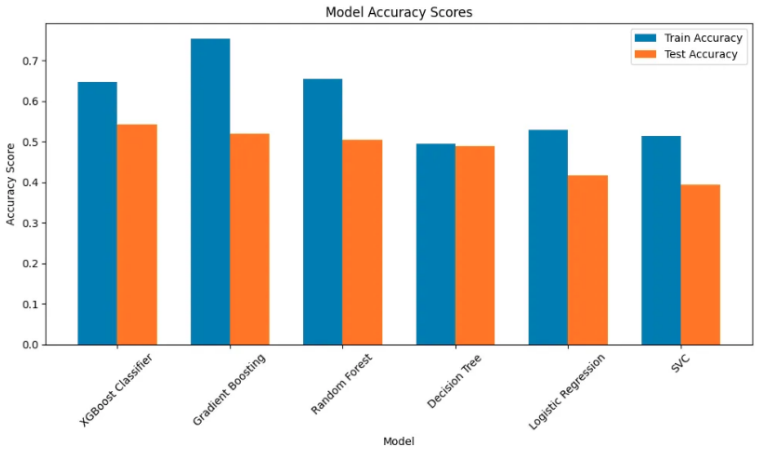


Fig. 4: Train and Test Accuracy for Differing Classification Models

XGBoost achieved the highest test performance across all evaluation metrics, making it the preferred model. However, the gap between training and test performance suggests some degree of overfitting. Gradient Boosting showed similar trends but underperformed slightly.

XGBoost also provided insight into feature importance. The most important predictors were mood-related variables (e.g., mood_mean_daily, mood_min_daily) and valence (e.g., valence_mean_daily). This aligns with psychological expectations and supports the relevance of these features in mood prediction. Given the structured nature of the features and the moderate sample size, tree-based ensemble methods were best suited to handle complex feature interactions.

Temporal Classification In addition to feature-based methods, we implemented a recurrent neural network (RNN) to capture temporal patterns in mood dynamics. The task remained the same: multiclass classification of next-day mood change (increase, decrease, or same), using a discretized version of the

continuous mood variable. Unlike the feature-based approach, this model takes sequences of daily observations (e.g., mood, valence, arousal) as input with the aim of leveraging temporal dependencies. Additionally, the RNN used data from the 'ToD' dataset, with three sets of observations per day. Each input sequence consisted of 5 consecutive days of data from a single individual, and the model was trained to predict the mood class for the day immediately following the sequence. A SimpleRNN architecture was selected after comparing RNN, GRU, and LSTM models. The output layer used a softmax activation, and the model was trained using categorical cross-entropy loss and the Adam optimizer.

A grid search was performed across a broad set of parameters to optimize the model architecture, training, and data preprocessing strategy. The hyperparameters included sequence length, model type (SimpleRNN, GRU, LSTM), hidden layer size, number of layers, dropout, learning rate, and whether to normalize features and/or targets. Gradient clipping and early stopping were also explored to address stability and overfitting. The best model was a 2-layer SimpleRNN with 16 hidden units and 0.3 dropout, trained with a learning rate of 0.001 and early stopping (patience = 5 epochs). Min-max normalization was applied per participant to both features and targets.

We used a 90/10 train-test split, prioritizing a larger training set to reduce overfitting. All sequences in both training and test sets were shuffled, and care was taken to ensure that each sequence contained data from only a single participant. Accuracy, precision, recall, and F1-score were used for evaluation.

The final SimpleRNN model achieved a test accuracy of 48.8%, outperforming the majority-class baseline. Precision, recall and the F1-score were all 49% on the test set, indicating moderate generalization. Early improvements in model training were followed by plateauing and eventual overfitting, which was mitigated by early stopping.

Despite its simplicity, the RNN captured some temporal structure in the data, however, it fell just short of reaching performance comparable to tree-based methods. The model started to overfit the data early on in the training process and it was unable to be determined how to avoid this. Nonetheless, this temporal model offered a complementary perspective to the feature-based approach by modeling mood evolution over time.

3.2 WINNING CLASSIFICATION ALGORITHMS

A competition titled "ICR - Identifying Age-Related Conditions" was held from May 11, 2023 until Aug 11, 2023 [2]. The dataset was anonymised health data with 56 features (mostly numeric, but with one categorical) and a binary target which indicated either the presence (1) or absence (0) of age-related conditions. A practice test set (without class labels) and a metadata file was also provided, while the real test set was kept hidden for accurate evaluation.

The task was binary classification, where the prediction was whether or not the patient did or did not have one of three age-related conditions. The evaluation was done via a balanced logarithmic loss as found below:

$$\text{Log Loss} = \frac{-\frac{1}{N_0}\sum_{i=1}^{N_0} y_{0i} \log p_{0i} - \frac{1}{N_1}\sum_{i=1}^{N_1} y_{1i} \log p_{1i}}{2}$$

For each observation, the model must output a predicted probability for each class. If the predicted probability is high for the correct class (the model is confident), it results in a lower loss if also correct. Conversely, confident but incorrect predictions result in a higher loss.

The winner was "Room722" who used a Deep Neural Network (DNN) based on a Variable Selection Network (VSN) architecture. This allowed the model to learn which input features were the most important, replacing the need for manual feature selection. They also used custom feature normalization. opting for "a linear projection with 8 neurons for each feature" instead of relying on other standard techniques such as MinMaxScaler or StandardScaler. Extremely high dropout rates were also used across the layers ($0.75 \rightarrow 0.5 \rightarrow 0.25$) with the aim of improving generalization (as the network is not able to rely on only one or a small group of neurons). Another important aspect includes the 10-fold cross-validation strategy which was repeated 10-30 times for each fold to account for unstable training (due to the high dropout used and the small dataset size). Additionally, a secondary "hardness to predict" label was created which helped the model focus on difficult samples during training.

By comparing this method to that of user "Anna" (who scored 9th) we can see what makes the winning method stand out. To begin with, Anna used a more traditional machine learning pipeline, completing scaling, imputation and feature engineering either manually or with simple well-established functions. Anna also used well-established tree-based classifiers (XGBClassifier) which run the risk of overfitting when not properly regularised. In contrast, the winning approach used a unique deep learning approach, where the model learns its own internal feature transformations. Rather than engineering features manually, the network learns which inputs matter most. While this approach is more complex and sensitive to overfitting, by combining it with extreme dropout rates throughout the network, the model is forced to generalise, thereby decreasing the chance that it overfits the training data. Overall, the winning entry outperformed the others due to the specificity of the approach. This is in contrast to relying on a "standard" pipeline that may not be sufficiently tailored to the task at hand.

4 ASSOCIATION RULES

Traditional association rules, such as the Apriori algorithm, identify frequent co-occurrences of items in transactional datasets. An example being that customers who purchase bread and butter may also buy milk. However, a problem with such a technique is that the rules are learned in a bottom-up way with no real semantic learning of the items that are grouped together. This can make it difficult to decide which of the found "rules" are of value or interest. A way to capture semantic relationships and develop more meaningful and interpretable rules is by using taxonomies, which group individual items into meaningful hierarchical

categories (such as the example: Pizza Margherita and Pizza Quattro Formaggi both belong to the pizza category).

Van Leeuwen and Kok [3] explored this in their paper "Interesting Association Rules in Multiple Taxonomies", where they describe a method to enhance association rules by using multiple taxonomies. They organize items into hierarchies, where leaf nodes represent individual products and intermediate (internal) nodes represent broader categories. This allows rules to be discovered in higher levels of abstraction, leading to the discovery of more generalized patterns. The authors also describe the notion of "interestingness", which is when the support of a lower-level rule differs significantly from what would be expected based on the support of the parent-level rule. Such deviations are interesting as they may reveal lower-level semantically meaningful relationships that are not captured by general category patterns.

The advantages of such a taxonomy-based approach is that it allows for a more interpretable and condensed explanations of the rules that are found. This also allows for generalizations that could lead to better results compared to the lower level rules that are formed from more classical association rule algorithms. It can also offer unique insights and opportunities for further exploration when the lower level and higher level rules don't align (when there is "interestingness").

The disadvantage, however, is that this method requires accurate taxonomies that may not always be available or that may take a significant amount of time and effort to develop. Furthermore, this method increases the computational complexity due to the larger number of possible item combinations introduced by hierarchical groupings. This also opens the door to ambiguity in rule interpretation when items belong to simultaneous higher-order categories, making it hard to determine which category is truly responsible for certain detected associations.

5 NUMERICAL PREDICTION

Feature-Based Regression For the regression task, we reused the same dataset and feature selection process as in the classification task, using the top 35 most correlated 'Daily' features. However, instead of discretizing next-day mood into categories, we treated it as a continuous variable. This allowed us to directly model mood prediction using regression algorithms. We tried the following models: Linear Regression, Ridge Regression, Lasso Regression, Elastic Net (L1 + L2 regularisation), Decision tree, Random Forest, XGBoost, gradient boosting, and SVR. For all models, we used Mean Squared Error (MSE) as the loss function during training, as it penalizes larger errors more heavily and is well-suited for continuous outcome prediction.

Each model was optimized using grid search and 3-fold cross-validation on the training set in a similar way as for the classification task. The hyperparameter grid for Gradient Boosting (the best performing model) is found in Table 6, with the best configuration in bold.

To prevent data leakage and ensure fair evaluation, the dataset was split into 1142 training instances and 127 test instances, randomly shuffled. Data was

Table 6: Hyperparameter Grid for Gradient Boosting

n_estimators	[50, 100, **200**]
learning_rate	[0.01, **0.05**, 0.1]
max_depth	[2, **3**, 4]
min_samples_split	[2, **5**]
subsample	[**0.8**, 0.9, 1.0]

standardized per participant using the mean and variance from their training data only. We decided to evaluate the model based on both MSE and MAE.

Gradient boosting outperformed all other models achieving the best performance on the test set with a MAE of 0.46 and a MSE of 0.40. Visual inspection of predictions (see Figure 5) show that while the model made reasonable predictions for central mood values, it struggled with extreme moods, potentially due to the infrequency of these values which made them hard to model accurately.

Gradient boosting was chosen as the final model due to its superior generalization ability on unseen data. Feature importance analysis revealed mood_last_daily, valence_max_daily, and weekday_4 (Friday) as the most influential predictors, suggesting that both end-of-day mood and contextual factors (e.g., weekday) play a substantial role in predicting next-day mood.

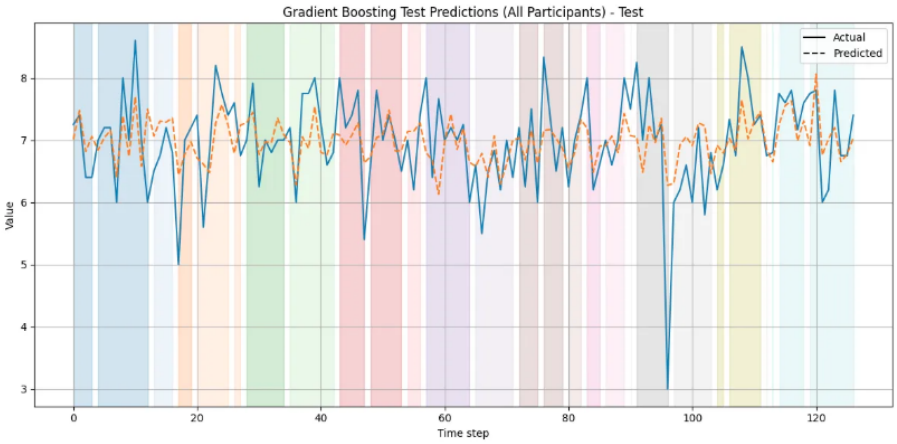


Fig. 5: Gradient Boosting Predicted vs Actual Mood *(participants by color)*

Temporal Regression To capture the temporal dependencies inherent in mood prediction, we tested an LSTM, RNN, and GRU with the final decision to implement a recurrent neural network using a Gated Recurrent Unit (GRU). GRUs are well-suited to sequential data and can effectively model time-dependent patterns in behavior and mood. We fed the model 5-day 'ToD' sequences of standardized features, ensuring each sequence came from the same individual to preserve temporal consistency. A residual connection was used to allow the model to directly

map the final input to the output, improving learning stability and enabling fallback to simpler linear behavior when needed. As with the feature-based models, the temporal models were also trained using used the mean squared error loss.

We conducted hyperparameter tuning of key GRU architecture parameters, experimenting with sequence lengths, hidden dimensions, and number of layers. The best results were obtained using a single-layer GRU with 16 hidden units, no dropout, and a residual shortcut connection. We trained for 20 epochs using a batch size of 256 and the Adam optimizer (learning rate = 0.001), which adapts the learning rate for each parameter separately based on gradients. Hidden states were initialized from a normal distribution N(0, 0.01) to encourage better convergence early in training.

The data was split into an 80/20 train-test split, ensuring the scaler was fit only on the training set and applied consistently to both sets to avoid data leakage. Features and the mood target were normalized between 0 and 1 using MinMax scaling for stable gradient-based optimization. The target was scaled per participant to account for individual differences in mood range and prevent global bias toward population-level averages. As with the feature based approach, performance was assessed using the MSE and MAE.

The final GRU model achieved a test MAE of 0.541 and a MSE of 0.522 with the predictions observed to closely match the mean of the targets (See Fig 6), but with a compressed variance (lower std) and a restricted range.

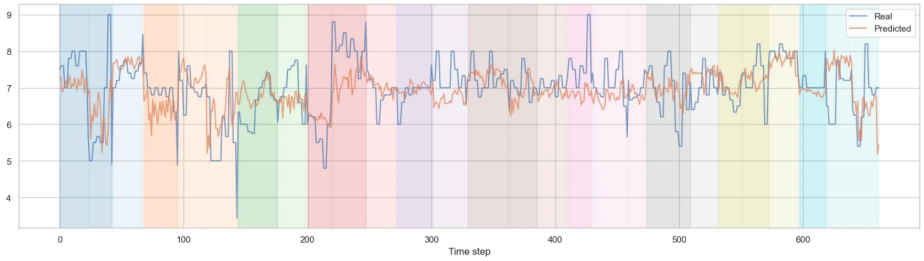


Fig. 6: GRU Predicted vs Actual Mood *(participants by color)*

Visualizations confirmed that the GRU struggled with extreme mood predictions, producing more accurate results in the mid-range.

The difficulty for GRU to model more extreme or volatile mood swings may be due to the under-representation of such examples in the training data, limiting the model's ability to generalize to these cases. The residual shortcut allowed the model to learn useful mappings even in this simple setup, but further gains may require either more expressive models (e.g., deeper architectures) or better feature engineering strategies. The temporal model's performance was comparable to our non-temporal Gradient Boosting model, but its ability to incorporate recent history makes it more theoretically suitable for modeling mood as a sequential process.

6 EVALUATION

6.1 CHARACTERISTICS OF EVALUATION METRICS

Mean Squared Error (MSE) **Mean Absolute Error (MAE)**

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

Both the MSE and MAE are measures of prediction error, however, they differ in how they penalize incorrect predictions. For MSE, each deviation (difference between prediction and true score) is squared, giving a significant weight to large deviations compared to smaller ones (significantly wrong predictions are severely penalized). This is important if the goal is to punish severely inaccurate predictions, important for applications such as forecasting medical dosages or detecting anomalies in financial transactions, where large errors can have severe consequences. Comparatively, MAE takes the absolute difference which is more robust against such outliers as it treats all errors linearly (the contribution to the loss grows in direct proportion to the size of the error). MAE may be preferred in use cases like predicting housing prices, where overly penalizing occasional large deviations (resulting from unusual properties or rare listings) is not necessary as the focus is on providing a reliable average estimate.

6.2 IMPACT OF EVALUATION METRICS

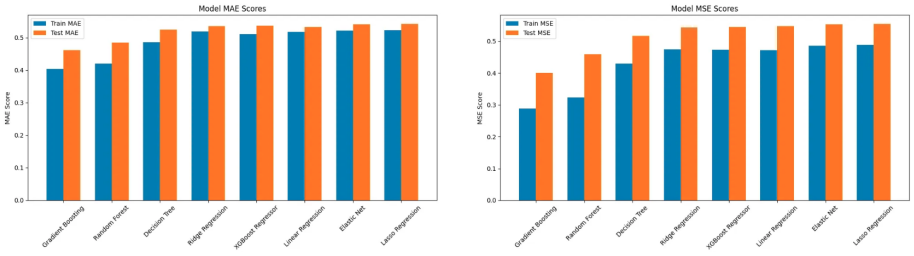


Fig. 7: MAE vs MSE

As shown in Figure 7, MAE values across models were consistently higher than the corresponding MSE values. This first appeared strange as MSE is known to penalize large errors more harshly. However, in this situation, most prediction errors were smaller than one. When errors fall within the interval (0,1), squaring them results in smaller values than the original errors (for example 0.5 squared equals 0.25). This behavior of MSE explains why it produced smaller numerical scores than MAE in our results.

In terms of model performance, Gradient Boosting achieved the lowest MAE and MSE scores on the test set, meaning that both evaluation metrics both suggest that this model performs the best compared to others.

References

1. James A. Russell. "A Circumplex Model of Affect." *Journal of Personality and Social Psychology*, 39(6):1161, 1980.
2. Aaron Carman, Alexander Heifler, Ashley Chow, CGlenICR, and Ryan Holbrook. *ICR - Identifying Age-Related Conditions*. Kaggle, 2023. https://kaggle.com/competitions/icr-identify-age-related-conditions
3. van Leeuwen, M., Kok, J.N. Taxonomies and Interesting Itemsets. *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, pp. 313–324. Springer (2002). https://liacs.leidenuniv.nl/ kosterswa/asso2.pdf
4. GeeksforGeeks. Categorical Cross-Entropy in Multi-Class Classification. Available at: https://www.geeksforgeeks.org/categorical-cross-entropy-in-multi-class-classification/. Accessed: April 18, 2025.