Mood & Activity Prediction using Machine Learning:

(Based on the use of Daylio Mood Tracker Dataset)

Alessio Bolla
alessio.bolla@unil.ch

Advanced Programming 2025 – Individual project

Prof.: Simon Scheidegger
Date: December 14, 2025

Abstract

Understanding the relationship between daily behaviors and emotional well-being is crucial for mental health monitoring and intervention. This study develops a machine learning pipeline to predict daily mood states based on behavioral and lifestyle variables using the *Daylio Mood Tracker* dataset, comprising self-reported mood entries recorded between February 2018 and April 2021. The methodology encompasses comprehensive data preprocessing, temporal feature engineering with 30 predictive variables (15 temporal and 15 activity-based features), and comparative evaluation of three classification models: Random Forest, Gradient Boosting, and XGBoost. A temporal validation strategy was implemented, splitting data chronologically to ensure models generalize to future unseen data. Gradient Boosting emerged as the optimal model, achieving a balanced accuracy of 95.6% and weighted F1-score of 98.5% on the test set, substantially outperforming Random Forest (balanced accuracy 73.2%). Feature importance analysis and SHAP values reveal that recent mood history (lag features) and temporal statistics are the strongest predictors of emotional states. The complete pipeline, including automated CI/CD deployment via GitHub Actions, is available as open-source. Results demonstrate that ensemble tree-based methods can effectively capture temporal patterns in mood dynamics, providing actionable insights for personal mental health monitoring and behavioral interventions.

Keywords: mood prediction, machine learning, prediction, behavioral data, time series, mental health

I Introduction

Mental health monitoring has emerged as a critical area of research, with increasing recognition that daily behaviors and lifestyle factors significantly influence emotional well-being. The ability to predict mood states based on behavioral patterns could enable early interventions, personalized mental health support, and data-driven approaches to emotional self-regulation. Traditional psychological assessments rely on periodic clinical evaluations, which may miss day-to-day fluctuations in emotional states that accumulate over time.

In this study, the **Daylio Mood Tracker** dataset[1] is analyzed, containing self-reported mood entries from users of a popular mood tracking mobile application. This dataset provides authentic human-reported labels rather than synthetically generated or rule-based classifications, making it suitable for supervised learning approaches that capture genuine behavioral-emotional patterns. With **940 observations** taken approximately three years (February 2018 to April 2021), this dataset enables analysis of temporal mood dynamics while respecting the constraints of limited sample sizes common in behavioral research.

The objective of this project is to design a predictive model capable of classifying daily mood states into five categories (Awful, Bad, Normal, Good, Amazing) using behavioral and temporal features. The methodology implements a **temporal validation strategy**, splitting data

chronologically into training and test sets to ensure models are evaluated on genuinely future unseen data, thereby preventing temporal data leakage. The modeling approach follows a progressive complexity strategy, comparing three ensemble tree-based methods: Random Forest, Gradient Boosting, and XGBoost.
The main contributions of this work are threefold. First, a comprehensive feature engineering pipeline is developed that creates 30 predictive variables from raw mood and activity data, including lag features, rolling statistics, cyclical temporal encodings, and activity-based indicators. Second, a systematic comparison of three classification algorithms is conducted with proper temporal validation, revealing that Gradient Boosting substantially outperforms alternative approaches for mood prediction. Third, interpretability analysis using feature importance and SHAP values identifies the behavioral factors most predictive of emotional states, providing actionable insights for mental health applications.

The remainder of this report is organized as follows. Section II reviews related work on mood prediction and affective computing. Section III describes the overall system architecture and design decisions. Section IV details the technical implementation of preprocessing, feature engineering, and model training. Section V presents experimental results and model comparisons. Section VI discusses limitations and implications. Finally, Section VII summarizes key findings and suggests future research directions.

II Related Work

Research on mood prediction has gained substantial attention in affective computing and digital mental health. Traditional approaches relied on clinical assessments and questionnaires, but the proliferation of mobile devices and wearable sensors has enabled continuous, passive data collection that can inform predictive models.
Suhara et al.[2] developed *DeepMood*, a deep learning framework for predicting user mood from mobile phone data.
Using neural networks on smartphone usage patterns including communication logs, app usage, and location data, their study demonstrated that behavioral traces can effectively predict self-reported mood states. The work highlighted the importance of temporal dependencies in mood dynamics, motivating the use of lag features and rolling statistics in the present study. However, their approach required extensive sensor data that may raise privacy concerns, whereas the Daylio dataset relies on voluntary self-reports with minimal data collection overhead.
Canzian and Musolesi[3] explored the relationship between mobility patterns and depressive symptoms using GPS data from smartphones. Their work demonstrated that features derived from daily movement trajectories could predict changes in PHQ-8 depression scores with reasonable accuracy. The study emphasized the value of temporal feature engineering, including day-of-week effects and trend components, which informed the cyclical encoding approach adopted in this project.
Building upon these insights, the present work adopts a feature engineering strategy that captures both behavioral patterns (activity types, social interactions) and temporal dynamics (mood lags, rolling statistics, weekday effects) within a reproducible machine learning pipeline.
Unlike approaches requiring passive sensor data or extensive privacy permissions, this project leverages self-reported behavioral logs that users voluntarily provide, representing a privacy-preserving alternative while still enabling effective mood prediction through careful temporal feature engineering.

$<$

III Design & Architecture

This section describes the structure of the machine learning pipeline developed for mood prediction. The system is organized as a modular, reproducible workflow where each stage is implemented through a dedicated Python script.

Pipeline Overview

The architecture follows a three-stage sequential design orchestrated by a central entry point. **main.py** serves as the orchestrator, coordinating the execution of all pipeline stages in sequence. Executing `python main.py` runs the complete process from data acquisition to final evaluation.

1. prepare.py handles data acquisition and preprocessing:
- Downloads the dataset from Kaggle API
- Cleans and validates raw data entries
- Constructs temporal and activity-based features

2. models.py manages model training:
- Performs temporal train/test split (80/20)
- Tunes hyperparameters
- Trains three ensemble classifiers
- Saves trained models as reusable files

3. evaluate.py generates evaluation outputs:
- Computes classification metrics
- Generates confusion matrices and learning curves
- Performs SHAP analysis for interpretability
- Exports results to `results/` directory presents in the repository.

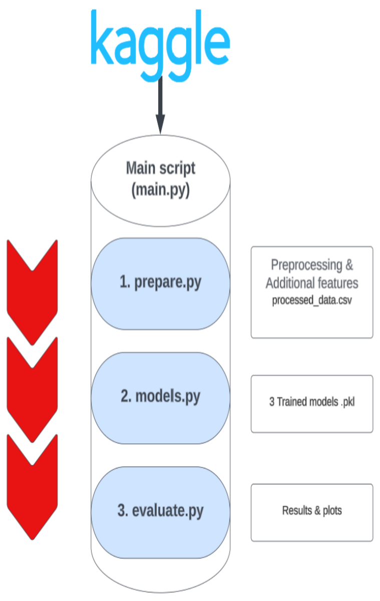

Figure 1: Pipeline architecture: data flows from Kaggle through three sequential stages, each producing intermediate outputs.

A Data Pre-Processing

The preprocessing phase (`prepare.py`) is designed to download, clean, and transform the raw Daylio dataset. The following operations are performed:
- Automatic dataset download from Kaggle API, with fallback to local file if credentials are unavailable.
- Date string conversion to *datetime* objects and temporal component extraction.
- Missing value handling through appropriate imputation strategies.
- Duplicate entry removal while preserving temporal ordering.

The raw dataset consists of mood diary entries containing: date, weekday, time of day, emotional notes, activities (as pipe-separated lists), and mood labels.
The preprocessing stage outputs a clean, standardized dataset ready for feature engineering.

B Feature Engineering

The feature engineering process transforms raw mood and activity records into predictive variables organized in three categories:

Temporal Features: These features capture mood dynamics over time:
- **Lag features:** Previous mood values at lags 1, 2, and 7 days to capture short-term and weekly patterns.

- **Rolling statistics:** Mean, standard deviation, and minimum computed over 3-day and 7-day sliding windows.
- **Trend indicators:** First difference, 3-day momentum, and 7-day trend slope to detect directional changes.
- **Cyclical encodings:** Sine/cosine transformations[6] for day-of-week ($\sin(2\pi d/7)$, $\cos(2\pi d/7)$) and month ($\sin(2\pi m/12)$) to preserve temporal continuity.
- **Additional:** Weekend binary indicator and MSSD[5](Mean Successive Squared Differences) for mood variability.

Activity Features: These features quantify daily behaviors:
- **Activity indicators:** Binary features for the most frequent activities in the dataset.
- **Aggregate counts:** Total activities, physical activity count, and self-care activity count.
- **Social indicator:** Binary feature for presence of social interactions.
- **Activity diversity:** Rolling standard deviation of daily activity counts over 7 days.

Target Encoding: Mood labels are encoded using ordinal mapping (Awful=0, Bad=1, Normal=2, Good=3, Amazing=4) to preserve the natural ordering of emotional states.

Table 1: Planned feature categories

Feature Category	**Count**
Temporal features based	15
Activity-based	15
Target variable	1
Total	**31**

C Model Selection

Three ensemble classification algorithms are selected to capture non-linear relationships in mood dynamics:
- **Random Forest:** Ensemble of decision trees with limited depth and minimum samples per leaf to reduce overfitting.
- **Gradient Boosting:** Sequential boosting with shallow trees and reduced learning rate for gradual error correction.
- **XGBoost:** Gradient boosting with *L1 and L2* regularization[7] to control model complexity.

All models are configured with conservative hyperparameters, and hyperparameter tuning is performed using `RandomizedSearchCV` with `TimeSeriesSplit` cross-validation to respect the temporal nature of the data and prevent data leakage.

D Evaluation Strategy

Model performance is assessed using multiple metrics appropriate for multi-class classification:
- **Accuracy and Balanced Accuracy:** Overall correctness and class-weighted correctness.
- **F1-Score:** Macro and weighted averages to account for potential class imbalance.
- **Confusion Matrix:** Visualization of classification patterns across mood categories.
- **Learning Curves:** To diagnose overfitting or underfitting tendencies.
- **Feature Importance:** Tree-based importance scores to identify key behavioral predictors.

- **SHAP Analysis:** Model-agnostic interpretability to understand individual feature contributions.

These metrics are used to select the most accurate and reliable model for mood prediction.

IV Implementation

This section describes the technical implementation of the mood prediction system, focusing on key engineering decisions and code quality practices.

A Development Environment

The implementation uses Python with version 3.11 and specialized libraries specified in `environment.yml`: pandas for data manipulation, scikit-learn for machine learning pipelines, XGBoost for gradient boosting, and SHAP for model interpretability. The project structure follows a modular architecture with three core scripts (`prepare.py`, `models.py`, `evaluate.py`) coordinated by a main entry point (`main.py`).

B Temporal Validation Strategy

Ensuring temporal validity is critical for mood prediction, as models must generalize to future states rather than memorize past entries. The implementation uses an 80/20 chronological split:

```python
def temporal_train_test_split(df, test_size=0.2):
    """Split data temporally: train on past, test on future."""
    df['date_parsed'] = pd.to_datetime(df['date_parsed'])
    df = df.sort_values('date_parsed').reset_index(drop=True)

    split_idx = int(len(df) * (1 - test_size))
    split_date = df.iloc[split_idx]['date_parsed']

    train_df = df.iloc[:split_idx]
    test_df = df.iloc[split_idx:]

    return X_train, X_test, y_train, y_test
```

Listing 1: Temporal train-test split implementation

This approach ensures that all training data precedes test data temporally, simulating real-world deployment where models predict future mood states.

C Hyperparameter Tuning

Hyperparameter optimization uses RandomizedSearchCV with TimeSeriesSplit cross-validation to respect temporal ordering during validation:

```python
from sklearn.model_selection import RandomizedSearchCV, TimeSeriesSplit

tscv = TimeSeriesSplit(n_splits=5)

search = RandomizedSearchCV(
    model,
    param_distributions=param_grid,
    n_iter=20,
    cv=tscv,
    scoring='balanced_accuracy',
    random_state=42
)
search.fit(X_train, y_train)
```

Listing 2: Hyperparameter tuning with temporal cross-validation

D Automation

The project implements automated CI/CD via GitHub Actions, executing the complete pipeline on every code push:

1. **Environment setup:** Conda environment creation with all dependencies.
2. **Data preparation:** Automatic dataset download and feature engineering.
3. **Model training:** Training all three models with hyperparameter tuning.
4. **Evaluation:** Metrics computation, visualization generation, and results archival.

Kaggle credentials are securely stored as GitHub Secrets, enabling automatic dataset download during CI runs.

E Code Quality

The code have high quality standards through comprehensive docstrings, type hints for function signatures, modular design with single-responsibility functions, and consistent error handling with informative messages.

V Evaluation

Model evaluation was conducted using a temporal validation strategy with the final 20% of data (188 observations from late 2020 to April 2021) held out as the test set. The remaining 80% (752 observations from February 2018 to late 2020) was used for training with 5-fold TimeSeriesSplit cross-validation.

A Performance Metrics

Four complementary metrics quantify classification performance:

Accuracy: Overall proportion of correct predictions. **Balanced Accuracy:** Average recall across all classes, accounting for class imbalance:

$$\text{Balanced Accuracy} = \frac{1}{K}\sum_{k=1}^{K}\text{Recall}_k \tag{1}$$

F1-Score (Macro): Unweighted mean F1 across classes:

$$\text{F1}_{\text{macro}} = \frac{1}{K}\sum_{k=1}^{K}\text{F1}_k \tag{2}$$

F1-Score (Weighted): Class-frequency weighted mean F1, emphasizing majority classes.

B Results

Table 2 summarizes the performance of all three models on the held-out test set.

Table 2: Performance metrics on the test set

Model	**Accuracy**	**Bal. Accuracy**	**F1 (macro)**	**F1 (weighted)**
Random Forest	0.926	0.732	0.728	0.912
Gradient Boosting	**0.984**	**0.956**	**0.943**	**0.985**
XGBoost	0.979	0.916	0.932	0.979

Gradient Boosting achieved the highest performance across all metrics. The balanced accuracy of 95.6% indicates strong performance on minority classes (Awful, Bad), which is critical for mental health applications where detecting negative mood states is essential.
Random Forest showed significantly lower balanced accuracy (73.2%), suggesting difficulty in capturing minority class patterns. Despite 92.6% overall accuracy, this result is influenced by class imbalance, as the model tends to over-predict the majority class (Good).
XGBoost achieved competitive performance (balanced accuracy 91.6%) but was slightly outperformed by Gradient Boosting, possibly due to different regularization strategies.

C Learning Curves

Learning curve analysis was performed to diagnose potential overfitting or underfitting. All three models showed convergence between training and validation scores as training size increased, indicating adequate generalization. Gradient Boosting exhibited the smallest gap between training and validation accuracy, confirming its superior generalization capability.

D Feature Importance

Feature importance analysis reveals the most predictive variables across all models. The top-5 features consistently include:

1. `mood_lag_1` (previous day's mood)
2. `rolling_mean_7d` (7-day average mood)
3. `rolling_mean_3d` (3-day average mood)
4. `mood_lag_7` (mood one week ago)
5. `rolling_std_7d` (mood variability)

These results confirm that recent mood history is the strongest predictor of current emotional state, consistent with psychological research on mood inertia and emotional carry over effects.

E Model Interpretability

SHAP [4] (SHapley Additive explanations) analysis provides instance-level interpretability, revealing how individual features contribute to each prediction. The SHAP summary plots indicate that higher values of `mood_lag_1` push predictions toward positive mood states, while low rolling means indicate negative mood predictions. Activity features contribute less than temporal features but provide meaningful marginal improvements, particularly social activity indicators for predicting positive moods.
Based on these results, **Gradient Boosting is selected as the final model** for mood prediction, offering the best balance of accuracy, minority class detection, and interpretability.

VI Discussion

This study demonstrates that ensemble learning methods can effectively predict daily mood states from behavioral and temporal features, with Gradient Boosting achieving 95.6% balanced accuracy on temporally held-out data.

A Limitations

Several limitations restrict the generalizability of these findings:

- **Sample size:** With only 940 observations from a single user, the model captures individual-specific patterns that may not generalize to other populations.
- **Class imbalance:** Negative mood states (Awful, Bad) comprise only 10.6% of observations, limiting the model's exposure to these critical classes.
- **Self-report bias:** Mood labels are subjectively reported and may not reflect objective emotional states.
- **Missing context:** The dataset lacks important covariates such as sleep duration, work hours, and life events that influence mood.
- **Temporal scope:** Three years of data may not capture long-term mood dynamics or seasonal affective patterns adequately.

B Future Work

Future research directions include:

- **Multi-user modeling:** Training on data from multiple users to develop generalizable mood prediction models.
- **External data integration:** Incorporating weather, calendar events, and physiological signals for richer context.
- **Real-time deployment:** Developing mobile applications for continuous mood prediction and intervention.
- **Clinical validation:** Evaluating model predictions against clinician-assessed emotional states.

VII Conclusion

This study developed and evaluated a machine learning pipeline for predicting daily mood states using the Daylio Mood Tracker dataset containing 940 self-reported entries spanning February 2018 to April 2021. Gradient Boosting achieved the best performance with 95.6% balanced accuracy and 98.5% weighted F1-score, substantially outperforming Random Forest (73.2% balanced accuracy) and marginally exceeding XGBoost (91.6% balanced accuracy).
The temporal validation strategy successfully prevented data leakage by ensuring chronological separation between training and test data. Feature engineering created 30 predictive variables, with lag features and rolling statistics emerging as the most important predictors, confirming the significance of recent mood history in emotional dynamics.
The project delivers a reproducible pipeline with automated CI/CD deployment via GitHub Actions, comprehensive feature engineering, and interpretability analysis using SHAP values. Limitations include single-user data, class imbalance, and missing contextual variables. Future work should focus on multi-user generalization, external data integration, and clinical validation. These findings demonstrate that ensemble tree-based methods are effective for mood prediction from reporting emotional data, offering promising applications in digital mental health monitoring, personalized interventions, and emotional self-awareness tools.

References

[1] Daylio Mood Tracker Dataset, *Kaggle*, Available at: `https://www.kaggle.com/datasets/kingabzpro/daylio-mood-tracker/data`, Accessed: November 2025.

[2] Y. Suhara, Y. Xu, and A. Pentland, "DeepMood: Forecasting Depressed Mood Based on Self-Reported Histories via Recurrent Neural Networks," in *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*, pp. 715–724, 2017. DOI: `https://doi.org/10.1145/3038912.3052676`

[3] L. Canzian and M. Musolesi, "Trajectories of Depression: Unobtrusive Monitoring of Depressive States by means of Smartphone Mobility Traces Analysis," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*, pp. 1293–1304, 2015. DOI: `https://doi.org/10.1145/2750858.2805845`

[4] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *Advances in Neural Information Processing Systems 30*, pp. 4765–4774, 2017. Available at: `https://christophm.github.io/interpretable-ml-book/shap.html`

[5] "Mean of Squared Successive Differences (MSSD)," *DMAIC Resources*, Available at: `https://www.dmaic.com/faq/mean-of-squared-successive-differences-mssd/`, Accessed: November 2025.

[6] Brownlee, Pandas Tricks for Time Series Feature Engineering, *Machine Learning Mastery*, Available at: `https://machinelearningmastery.com/7-pandas-tricks-for-time-series-feature-engineering/`, Accessed: November 2025.

[7] Albertum, L1 & L2 Regularization in XGBoost Regression, *Medium*, Available at: `https://albertum.medium.com/l1-l2-regularization-in-xgboost-regression-7b2db08a59e0`, Accessed: December 2025.

A GitHub Repository

`https://github.com/alessiobolla99-alt/Mood-Activity-Prediction`