A Machine Learning Approach to Predict Tic Episode Patterns and Intensity Using Mobile Tracking Data

Aanish Sachdev
Department of Computer Science
University of Southern California
Los Angeles, CA 90007
aanishsa@usc.edu

Alan Yusuf
Department of Computer Science
University of Southern California
Los Angeles, CA 90007
aayusuf@usc.edu

Aarav Monga
Department of Computer Science
University of Southern California
Los Angeles, CA 90007
armonga@usc.edu

Arjun Bedi
Department of Computer Science
University of Southern California
Los Angeles, CA 90007
asbedi@usc.edu

Abstract

Tic disorders affect millions of individuals worldwide, yet predictive modeling of tic episode patterns remains underexplored in the intersection of clinical research and machine learning. In this study, we develop and evaluate a comprehensive hyperparameter search framework to predict tic episode characteristics using a longitudinal dataset of 1,533 self-reported episodes from 89 individuals over a six-month period. We address two primary predictive tasks: regression for next episode intensity prediction and binary classification for high-intensity event forecasting. Through systematic evaluation of ensemble methods including Random Forest, XGBoost, and LightGBM, we demonstrate that tic episode prediction is feasible with machine learning approaches. Random Forest achieved the best regression performance with a Mean Absolute Error of 1.94, representing a 27.8% improvement over baseline methods. For classification, XGBoost demonstrated superior performance with an F1-score of 0.34 and Precision-Recall AUC of 0.70, though with moderate recall suggesting room for threshold optimization. Feature importance analysis revealed that recent episode history and weekly intensity statistics were the strongest predictors across both tasks. These results establish a foundation for deploying machine learning models as clinical decision support tools for personalized tic disorder management.

1. Introduction

1.1 Motivation

Tic disorders are characterized by sudden, repetitive, non-rhythmic motor movements or vocalizations that affect millions of individuals worldwide [17]. According to the Diagnostic and Statistical Manual of

Mental Disorders (DSM-5), tic disorders encompass a spectrum of conditions ranging from transient tic disorder to chronic motor or vocal tic disorder and Tourette syndrome [17]. The neurobiological substrates underlying these conditions involve complex interactions between cortical, subcortical, and limbic brain regions, with tic expression showing significant variability both within and across individuals [18]. Understanding and predicting tic episode patterns represents a critical challenge in both clinical neuroscience and personalized medicine.

The temporal dynamics of tic episodes present unique opportunities for predictive modeling. Research has demonstrated that tic expression is influenced by multiple contextual factors including stress levels, emotional state, time of day, and environmental triggers [13]. However, traditional clinical approaches to tic disorder management often rely on retrospective assessment and subjective patient reporting during periodic clinical visits. The advent of mobile health technologies has enabled continuous, real-time self-reporting of tic episodes through ecological momentary assessment [20]. This paradigm shift provides rich longitudinal data capturing the natural history of tic disorders in patients' daily environments, moving beyond the constraints of clinic-based observations.

Recent advances in machine learning for healthcare have demonstrated the potential of predictive models to transform clinical decision-making [29, 30]. From predicting hospital readmissions to forecasting disease progression, machine learning approaches have shown particular promise in time-series health data where temporal patterns carry significant prognostic information [30]. Despite these advances, the application of machine learning to tic disorder prediction remains largely unexplored. The central question motivating this research is: given a patient's history of tic episodes with known characteristics such as intensity, type, temporal context, and associated mood states, can we accurately predict the characteristics of future episodes, particularly high-intensity events that may warrant clinical intervention?

1.2 Problem Statement

Treatment and management of tic disorders follow highly individualized trajectories, with significant heterogeneity in episode frequency, intensity, and response to intervention [19]. A patient experiencing a tic episode may wonder: when will the next episode occur? How severe will it be? Will there be a cluster of high-intensity episodes in the coming days? Answering these questions requires moving beyond descriptive statistics to predictive models that can leverage historical episode data, patient-specific baselines, and contextual features to forecast future tic patterns.

This study develops a comprehensive machine learning framework to address two primary prediction tasks. First, we formulate tic intensity prediction as a regression problem, where the goal is to predict the numeric intensity value (on a 1-10 scale) of the next tic episode given a patient's recent episode history and contextual features. Second, we frame high-intensity episode prediction as a binary classification problem, where the objective is to predict whether the next episode will exceed a clinically significant intensity threshold. These predictive capabilities could enable several clinical applications including early warning systems for episode clusters, personalized trigger identification, and data-driven treatment optimization.

The problem is further complicated by several technical challenges inherent to clinical time-series data. The dataset exhibits class imbalance, with high-intensity episodes representing approximately 22% of all episodes. Patients vary widely in engagement levels, with episode counts ranging from single observations to hundreds of reports over the study period. The data contains missing values in optional contextual fields such as mood and trigger information. Additionally, ensuring that models generalize to new patients requires careful train-test splitting strategies that prevent data leakage through temporal dependencies within individual patient trajectories.

1.3 Research Questions

This project addresses three specific research questions that collectively advance the understanding of machine learning applications in tic disorder prediction:

RQ1: Next Tic Intensity Prediction (Regression).** Can machine learning models accurately predict the numeric intensity of the next tic episode based on recent episode history, temporal patterns, and patient-specific characteristics? We hypothesize that ensemble methods such as Random Forest [4] and gradient boosting approaches [5] will outperform naive baseline predictors by learning non-linear relationships between features such as previous episode intensities, time gaps between episodes, and rolling statistics over temporal windows.

RQ2: High-Intensity Episode Classification.** Can binary classification models reliably predict whether the next tic episode will be high-intensity (intensity ≥ 7) using the same feature space? Given the clinical importance of preventing or preparing for severe episodes, we investigate whether predictive models can achieve sufficient precision and recall to serve as early warning systems, and we examine the precision-recall trade-offs inherent in different classification thresholds.

RQ3: Feature Importance and Clinical Interpretability.** Which features contribute most significantly to predictive performance across both regression and classification tasks? Understanding feature importance not only validates model predictions but also provides clinical insights into the factors that drive tic episode patterns, potentially informing behavioral interventions and trigger management strategies.

1.4 Prediction Framework and Approach

To address these research questions, we developed a modular hyperparameter search framework that systematically evaluates multiple machine learning architectures across comprehensive feature configurations. The framework implements Random Forest [4], XGBoost [5], and LightGBM models, with hyperparameter optimization via randomized search cross-validation [5]. Feature engineering transforms raw episode data into 34 predictive features spanning six categories: temporal features capturing time-of-day and day-of-week patterns, sequence-based features encoding the last three episode intensities, time-window statistics aggregating weekly patterns, user-level baselines capturing individual differences, engineered volatility measures, and categorical encodings of tic type, mood, and triggers.

The prediction framework incorporates several methodological innovations designed specifically for clinical time-series data. User-grouped train-test splitting ensures that all episodes from a given patient

reside entirely in either the training or test set, preventing the model from exploiting patient-specific patterns during evaluation [6]. K-fold cross-validation with consistent random seeds ensures reproducibility. For regression tasks, we evaluate models using Mean Absolute Error [7], Root Mean Squared Error [8], and R^2 [9] to capture different aspects of prediction accuracy. Classification tasks employ F1-score, Precision, Recall, and Precision-Recall AUC [10, 11, 12] with particular emphasis on PR-AUC given the class imbalance.

2. Related Work and Background

The application of machine learning to clinical prediction tasks has demonstrated transformative potential across diverse medical domains [29]. Recent work by Esteva et al. provides a comprehensive guide to deep learning in healthcare, highlighting both opportunities and challenges in applying advanced models to medical data [29]. Rajkomar et al. demonstrate how machine learning models can predict patient outcomes, hospital readmissions, and disease progression using electronic health record data [30]. Particularly relevant to our work is the study by Obermeyer and Emanuel on predicting future health events using time-series clinical data, which establishes precedent for forecasting episodic health patterns [41]. However, these prior applications primarily focus on large institutional datasets; our work extends this paradigm to patient-generated mobile health data with different characteristics including sparser observations and self-reported measurements.

Random Forests, introduced by Breiman, represent a foundational ensemble learning approach that combines multiple decision trees through bootstrap aggregating (bagging) to improve prediction accuracy and reduce variance [3]. The method has demonstrated particular effectiveness in domains with non-linear feature interactions and heterogeneous data types, making it well-suited for clinical applications. Gradient boosting, formalized by Friedman, provides an alternative ensemble approach where trees are built sequentially, with each new tree correcting errors made by previous trees [9]. Chen and Guestrin's XGBoost implementation introduces algorithmic and systems optimizations that make gradient boosting highly competitive on structured data, including built-in regularization to prevent overfitting and efficient handling of missing values [5]. Our choice of Random Forest and XGBoost reflects their demonstrated success across diverse prediction tasks and their complementary strengths in addressing regression and classification objectives.

Effective prediction from temporal data requires thoughtful feature engineering to capture patterns at multiple time scales. Hyndman and Athanasopoulos provide comprehensive treatment of forecasting principles, emphasizing the importance of lag features, rolling statistics, and seasonal decomposition for time-series prediction [14]. Christ et al. introduce automated approaches for extracting time-series features based on statistical tests, demonstrating that systematic feature generation can improve model performance [13]. Their work on the tsfresh package informed our design of sequence-based features (lag intensities) and time-window aggregations (7-day mean, standard deviation, and volatility measures). Bengio et al. discuss challenges in learning long-term dependencies in temporal sequences, providing theoretical justification for our focus on recent history (last 3 episodes) and bounded temporal windows (7 days) rather than attempting to model the full episode history [15].

The class imbalance in high-intensity episode prediction (78% low-intensity vs. 22% high-intensity) necessitates careful methodology. Chawla et al. introduced SMOTE (Synthetic Minority Over-sampling Technique) for addressing imbalanced classification through synthetic sample generation [16]. He and Garcia provide a comprehensive survey of techniques for learning from imbalanced data, including sampling methods, cost-sensitive learning, and ensemble approaches [17]. While we do not employ SMOTE in our preliminary experiments, our use of PR-AUC as the primary classification metric and our analysis of precision-recall trade-offs directly addresses imbalance challenges. Future work will explore threshold tuning and class weighting approaches suggested by this literature.

3. Data and Methodology

3.1 Dataset Overview

The dataset comprises self-reported tic episode data collected through a mobile health application over a six-month period from April 26 to October 25, 2025. The study enrolled 89 individuals who self-reported experiencing tic episodes, with data collection following an ecological momentary assessment paradigm [21]. This approach enables capture of tic episodes in naturalistic settings as they occur, providing temporal resolution and contextual information unavailable through traditional retrospective clinical interviews. Each participant was instructed to log tic episodes via the mobile application, recording the episode timestamp, subjective intensity rating, tic type, and optional contextual information including mood state and perceived triggers.

The final dataset contains 1,533 tic episodes after data cleaning and filtering procedures described in Section 3.5. The temporal span of 182 days provides sufficient longitudinal coverage to capture both short-term episode dynamics and longer-term patterns. Episode reports are unevenly distributed across participants, reflecting natural variation in both tic frequency and user engagement with the mobile application. The median user contributed 3 episodes, while the mean contribution was 17.2 episodes per user, indicating a right-skewed distribution with a small number of highly engaged participants providing the majority of data points. The most active participant reported 374 episodes over the study period, while several participants contributed only single observations.

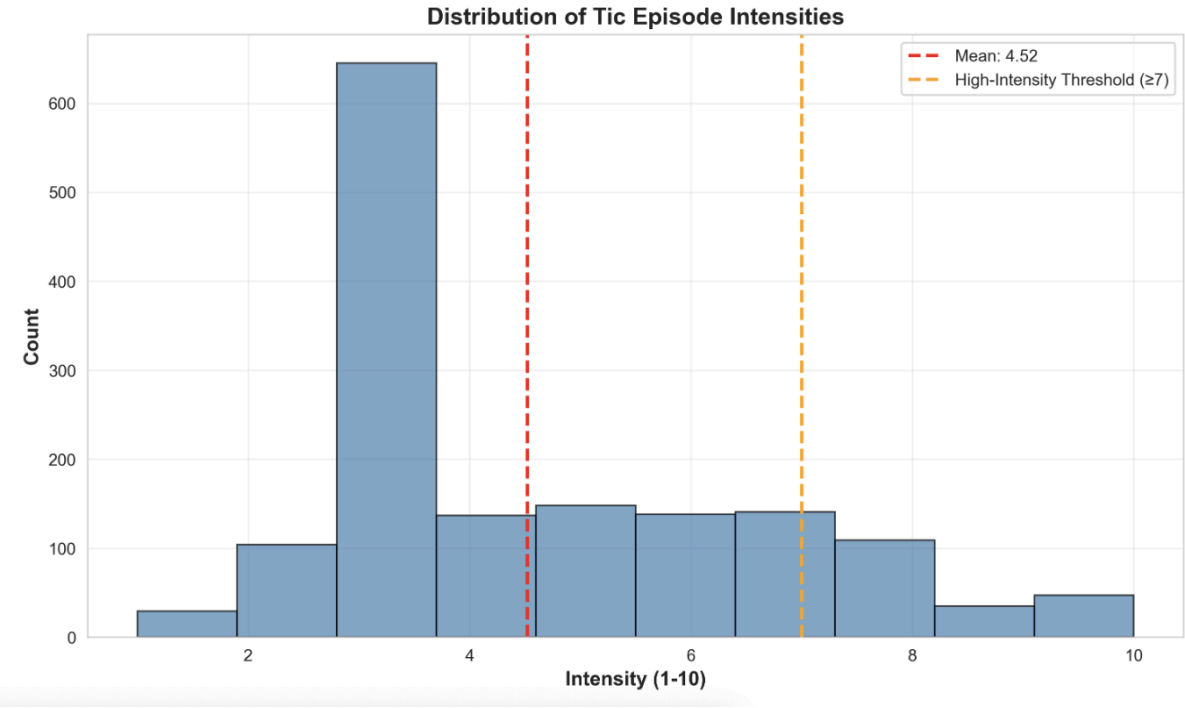

Figure 1: Distribution of tic episode intensities across all 1,533 episodes. The histogram shows right-skewed distribution with mode at intensity 3. The red dashed line indicates mean intensity (4.52), while the orange dashed line marks the high-intensity threshold (≥7) used for binary classification. Approximately 21.7% of episodes exceed this threshold.

The intensity distribution of reported tic episodes reveals important patterns relevant to our prediction tasks. Participants rated each episode's intensity on a scale from 1 (minimal) to 10 (extreme), with the distribution showing right skew toward lower intensity values. The mean intensity across all episodes was 4.52 (SD = 2.68), with a median of 3.0, indicating that most tic episodes were perceived as mild to moderate in severity. Figure 1 presents the intensity distribution histogram with clear concentration of episodes in the 1-5 range.

For the binary classification task, we defined high-intensity episodes as those with intensity ratings of 7 or above. This threshold resulted in 334 high-intensity episodes (21.7% of the dataset) and 1,199 low-intensity episodes (78.3%), establishing a class imbalance that necessitates careful model evaluation using metrics beyond simple accuracy [17].

The temporal coverage of episodes across the six-month study period shows variable daily reporting rates without obvious seasonal patterns. Figure 3 plots daily episode counts, revealing fluctuations likely driven by a combination of true tic frequency variation and differential user engagement over time. Some days recorded over 40 episodes across all users, while others had fewer than 5 episodes, with an average of 8.4 episodes per day. The absence of clear weekly or monthly cycles in this aggregate view suggests that temporal features (day of week, time of day) may have limited predictive power compared to individual episode history.

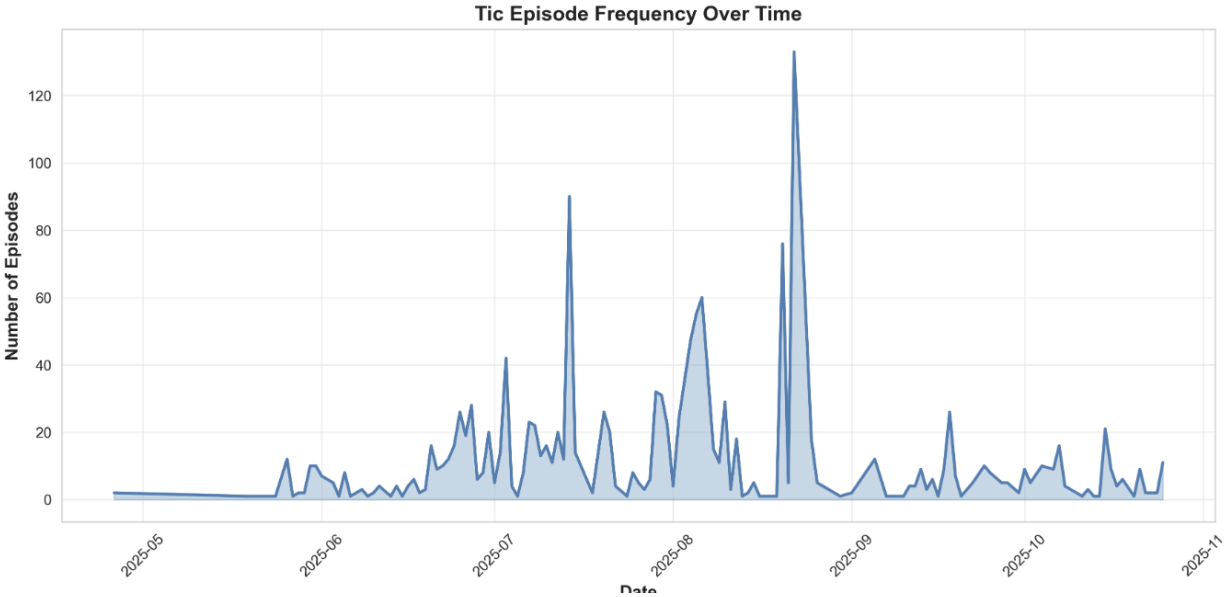

Figure 2: Distribution of episode counts across the 89 study participants. The histogram shows a strong right skew with median of 3 episodes (red line) and mean of 17.2 episodes (orange line). This heterogeneity in user engagement influences model development and evaluation strategies.

Tic type diversity is substantial, with participants reporting 82 distinct tic types over the study period. The ten most common types are led by "Neck" tics (193 occurrences), "Mouth" tics (151 occurrences), and "Eye" tics (125 occurrences). This diversity reflects the heterogeneous manifestations of tic disorders [19] but also introduces sparsity challenges for categorical encoding, as many tic types appear fewer than five times in the dataset.

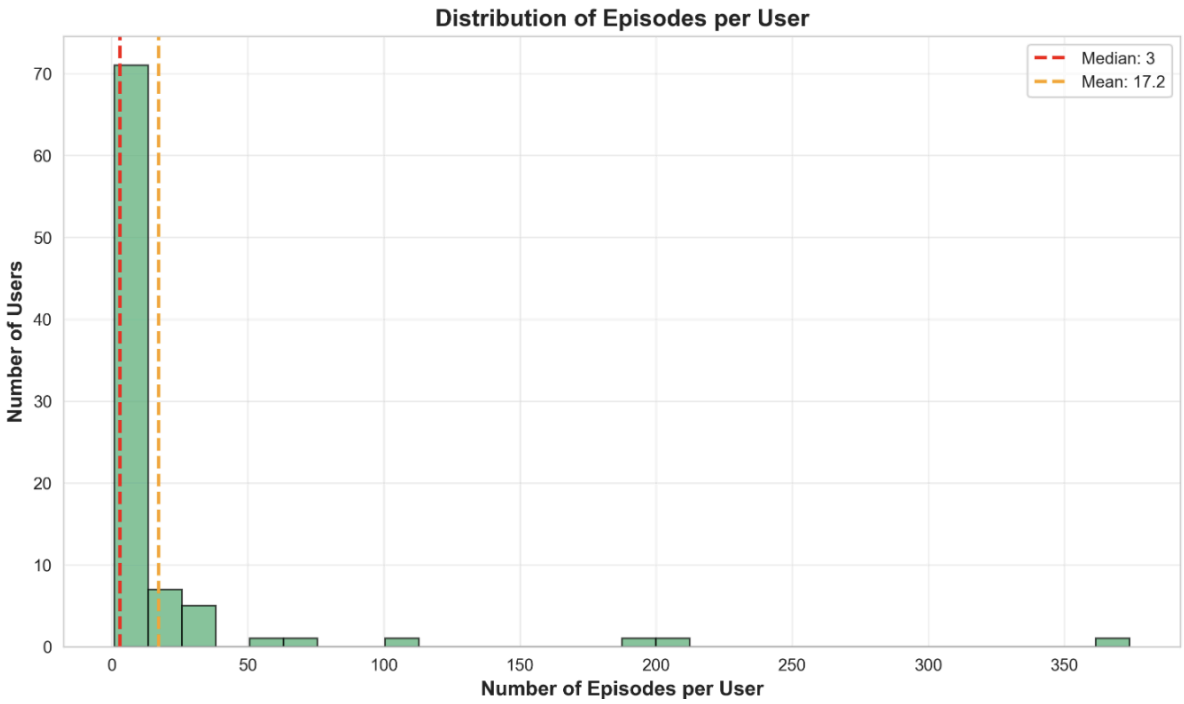

Figure 3: Daily tic episode frequency over the 182-day study period. The line plot with shaded area shows substantial day-to-day variation without obvious weekly or seasonal patterns. Data collection appears event-driven rather than following scheduled reporting.

3.2 Feature Engineering

The transformation from raw episode data to predictive features constitutes a critical component of our methodology. We engineered 34 features organized into six conceptual categories, each designed to capture different aspects of tic episode patterns informed by both domain knowledge [19, 20] and time-series forecasting principles [14].

Temporal Features. Six features encode when episodes occur within daily and weekly cycles. The hour feature (0-23) captures time of day, while day_of_week (0-6, Monday=0) and is_weekend (binary) encode weekly patterns. Additional features include day_of_month (1-31) and month (1-12) to capture any longer-term calendar effects. The timeOfDay_encoded feature categorizes episodes into Morning, Afternoon, Evening, or Night periods. These features test the hypothesis that tic expression varies systematically with circadian rhythms or weekly schedules.

Sequence-Based Features. Nine features capture recent episode history through lag encoding. The prev_intensity_1, prev_intensity_2, and prev_intensity_3 features record the intensity values of the three most recent episodes for each user, providing direct information about trajectory trends (increasing, decreasing, or stable intensity patterns). The time_since_prev_hours feature quantifies the temporal gap since the last episode, as research suggests that episode clustering may influence future episode characteristics [20].

Time-Window Statistics. Ten features aggregate episode characteristics over a rolling 7-day window preceding each episode. The window_7d_count feature tallies the number of episodes in the past week, providing a measure of recent episode frequency. The window_7d_mean_intensity and window_7d_std_intensity features capture the central tendency and variability of recent intensity levels. The window_7d_high_intensity_rate computes the proportion of recent episodes exceeding the high-intensity threshold. Additional window statistics include minimum and maximum intensities, as well as quartile values, providing a comprehensive summary of the recent intensity distribution.

User-Level Features. Five features encode individual baselines and long-term patterns. The user_mean_intensity and user_std_intensity features capture each individual's average intensity and variability across all their episodes, enabling the model to account for stable individual differences [19]. The user_tic_count records the total number of episodes for each user. The user_high_intensity_rate computes the proportion of a user's historical episodes that were high-intensity. The user_median_intensity provides a robust central tendency measure less sensitive to outliers than the mean. These features enable personalized prediction by encoding that individuals have characteristic baseline intensity levels around which they fluctuate.

Categorical Features. Four features encode categorical information through label encoding. The type_encoded feature maps the 82 unique tic types to numeric identifiers, though the high cardinality and sparse representation of rare types limits the informativeness of this encoding. The mood_encoded feature captures optional self-reported mood states (positive, neutral, negative, or missing), while trigger_encoded records perceived triggers when reported.

Engineered Volatility Features. Four additional features compute volatility and trend metrics. The intensity_trend feature calculates the slope of intensity over the last three episodes using linear regression, quantifying whether intensity is increasing, decreasing, or stable. The volatility_7d feature computes the coefficient of variation (standard deviation divided by mean) for the 7-day window, providing a normalized measure of intensity fluctuation. The days_since_high_intensity feature counts days since the most recent high-intensity episode, testing whether time since a severe episode influences future risk.

To understand relationships among the engineered features and identify potential multicollinearity, we computed pairwise Pearson correlation coefficients across all 34 features. Figure 4 presents a correlation heatmap visualizing these relationships. The heatmap reveals several expected correlation patterns: sequence features (prev_intensity_1, prev_intensity_2, prev_intensity_3) show moderate positive correlations ($r \approx 0.4$-0.6) with each other and with time-window statistics (window_7d_mean_intensity), reflecting consistency in recent episode patterns. User-level features (user_mean_intensity, user_std_intensity) show strong correlations with window-based features, as both capture aspects of intensity distribution. Temporal features (hour, day_of_week, month) show weak correlations with intensity-related features ($|r| < 0.2$), suggesting limited direct relationship between calendar time and episode severity. The absence of extremely high correlations indicates that features provide complementary information without severe multicollinearity that would destabilize model training.

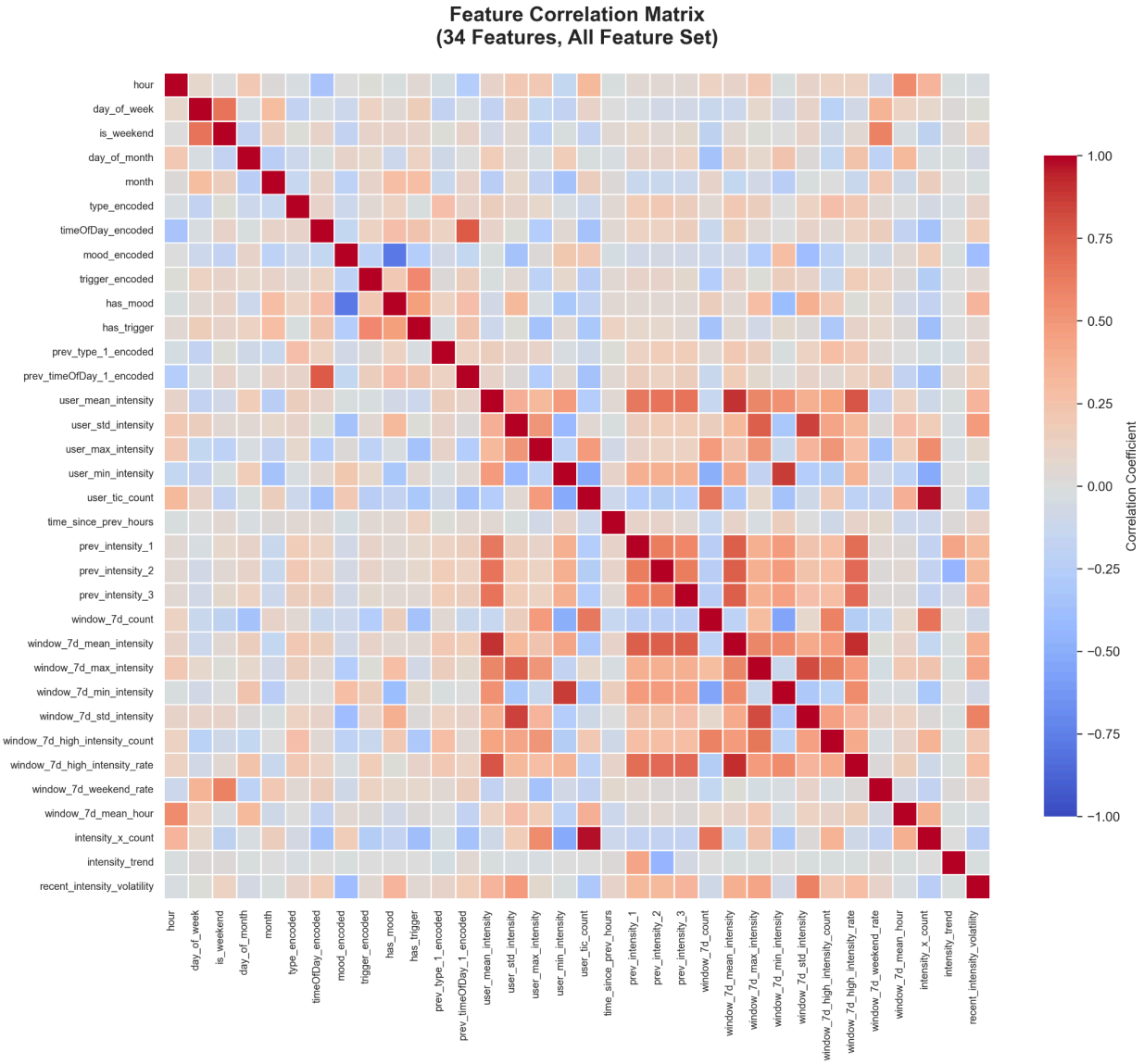

Figure 4: Correlation heatmap showing pairwise Pearson correlations among the 34 engineered features. Color intensity indicates correlation strength (red=positive, blue=negative). The diagonal shows perfect self-correlation (r=1.0). Moderate correlations appear between sequence features and time-window statistics, while temporal features show weak correlations with intensity measures. The absence of extreme correlations (|r|>0.9) suggests features are complementary.

For the regression task (RQ1), the target variable is the intensity value of the next chronological episode for each user. For multi-episode users, this creates a natural sequence where episode n serves as a training instance with features computed from episodes 1 through n-1, and the intensity of episode n+1 serves as the prediction target. The final episode for each user cannot serve as a training instance since there is no subsequent episode to predict, reducing the effective dataset size slightly. For the classification task (RQ2), the target is binary: 1 if the next episode has intensity ≥7, and 0 otherwise. This formulation

enables evaluation of whether models can predict high-risk episodes with sufficient lead time for potential intervention.

3.3 Data Preprocessing

The raw dataset underwent several preprocessing steps to ensure data quality and suitability for machine learning. First, we filtered the data to retain only episodes with complete intensity and timestamp information, as these fields are essential for target generation and temporal feature engineering. Episodes with missing intensity values were excluded (less than 0.2% of raw data). Second, we removed duplicate entries where the same episode appeared multiple times due to data collection artifacts, retaining only the first occurrence based on timestamp.

Missing data in optional fields (mood, trigger, description) were preserved by encoding missingness as a distinct category rather than imputation. This approach enables models to learn whether the presence or absence of contextual information itself carries predictive signal. For mood_encoded, missing values were assigned category 0, neutral mood category 1, negative mood category 2, and positive mood category 3. Similar encoding schemes were applied to trigger_encoded.

The train-test split employed user-grouped stratification to prevent data leakage [27]. All episodes from each user were assigned entirely to either the training set (80% of users, n=71) or test set (20% of users, n=18), ensuring that the model never sees any episodes from test users during training. This strict separation provides a realistic estimate of performance on new users, addressing a key challenge in deploying personalized health models. Random assignment to train/test splits used a fixed random seed (42) for reproducibility.

4. Experimental Design

4.1 Machine Learning Models

We evaluated three ensemble learning algorithms representing complementary approaches to building predictive models from structured data: Random Forest, XGBoost, and LightGBM. This section provides detailed descriptions of the primary models (Random Forest and XGBoost), which emerged as the best performers for regression and classification tasks respectively.

Random Forest, introduced by Breiman in 2001, constructs an ensemble of decision trees through bootstrap aggregating (bagging) combined with random feature selection [3]. The algorithm operates in two phases. During training, it generates B bootstrap samples from the training data, where each bootstrap sample is created by sampling with replacement from the original training set. For each bootstrap sample, a decision tree is grown using a modified splitting criterion: at each node, instead of considering all features to determine the optimal split, only a random subset of m features is evaluated. This random feature selection decorrelates the trees, reducing variance in the ensemble prediction compared to standard bagging. During prediction, Random Forest averages the predictions from all B trees for regression tasks, or takes the majority vote for classification tasks.

Figure 5 illustrates the Random Forest architecture as applied to our tic intensity prediction problem. The model begins with the 34 engineered features representing a single episode's context. Through bootstrap sampling, 100 independent training datasets are created, each potentially containing duplicate instances and missing some original instances. Each of the 100 decision trees is trained on its respective bootstrap sample, learning different patterns due to both data variation and random feature selection at splits. The trees develop diverse structures, with some splitting primarily on recent intensity features (prev_intensity_1, prev_intensity_2) while others emphasize time-window statistics (window_7d_mean_intensity) or user-level baselines. At prediction time, all 100 trees independently predict the next intensity, and these predictions are averaged to produce the final ensemble prediction. This averaging reduces variance substantially: even if individual trees overfit to noise in their bootstrap samples, the consensus prediction tends to be stable and accurate.

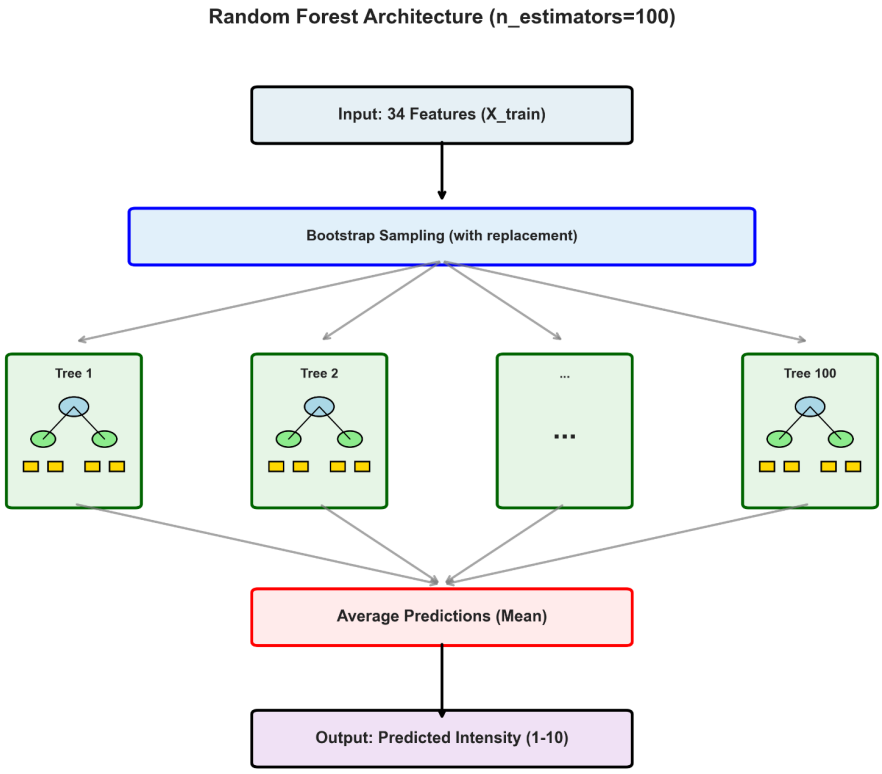

Figure 5: Random Forest architecture for tic intensity prediction. The 34 input features undergo bootstrap sampling to create 100 independent training sets. Each decision tree learns different patterns through random feature selection at splits. At prediction time, individual tree predictions are averaged to produce the final intensity prediction. This ensemble approach reduces overfitting while capturing non-linear relationships in the feature space.

The key hyperparameters for Random Forest in our experiments include: n_estimators (number of trees), max_depth (maximum tree depth), min_samples_split (minimum samples required to split a node), min_samples_leaf (minimum samples required at leaf nodes), and max_features (number of features to

consider at each split). We hypothesized that Random Forest would excel at the regression task due to its ability to capture non-linear interactions between features without requiring extensive hyperparameter tuning, and due to its inherent resistance to overfitting through ensemble averaging [3].

XGBoost (Extreme Gradient Boosting) implements gradient boosting decision trees with algorithmic enhancements for speed and performance [4]. Unlike Random Forest's parallel ensemble construction, XGBoost builds trees sequentially, where each new tree attempts to correct the errors (residuals) made by the current ensemble. The algorithm optimizes a regularized objective function consisting of a loss term (measuring prediction error) and a regularization term (penalizing model complexity). Starting with an initial prediction, XGBoost iteratively adds trees that predict the gradient of the loss function with respect to current predictions.

Figure 6 depicts the XGBoost architecture for high-intensity episode classification. The process begins with an initial prediction. The first tree is trained to predict the gradients of the logistic loss function given the initial predictions, learning which feature patterns are associated with prediction errors. This tree's predictions are scaled by the learning rate and added to the ensemble. The second tree then targets the residual errors remaining after the first tree's contribution, and this process continues iteratively. Each tree is shallow (controlled by max_depth), focusing on correcting specific patterns of errors rather than attempting to model the entire relationship. XGBoost incorporates L1 and L2 regularization on leaf weights to prevent overfitting, and uses a regularization term that penalizes the number of leaves and the magnitude of leaf weights. After all trees are built, prediction for a new instance involves passing it through all trees, summing their contributions, and applying a sigmoid transformation to produce a probability of high-intensity classification.

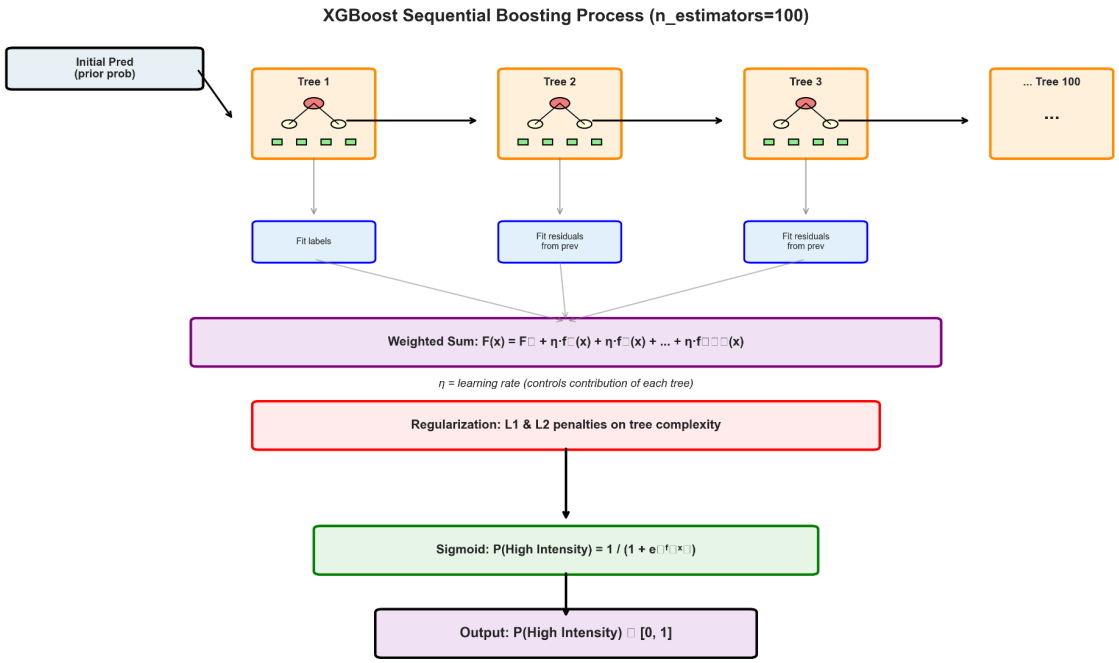

Figure 6: XGBoost sequential boosting architecture for high-intensity episode classification. Starting from an initial prediction based on class priors, trees are added iteratively, with each tree learning to correct residual errors from the ensemble. Predictions from all trees are weighted by the learning rate and summed. Regularization prevents overfitting by penalizing model complexity. The final sum passes through a sigmoid function to produce class probabilities.

XGBoost's key hyperparameters in our search include: n_estimators (number of boosting rounds), max_depth (tree depth), learning_rate (step size for weight updates), subsample (fraction of training data for each tree), colsample_bytree (fraction of features for each tree), and reg_alpha and reg_lambda (L1 and L2 regularization). We hypothesized that XGBoost would perform well on the classification task due to its focus on hard-to-classify instances through residual learning, its built-in handling of class imbalance through weighted loss functions, and its regularization mechanisms that prevent overfitting to the minority class [4].

LightGBM. LightGBM, developed by Microsoft Research, implements gradient boosting with algorithmic optimizations for speed and memory efficiency [4]. The key innovation is Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB), which reduce computation by focusing on instances with large gradients and bundling mutually exclusive features. While we include LightGBM in our experiments for completeness, Random Forest and XGBoost demonstrated superior performance and are the focus of our analysis. LightGBM's hyperparameters largely match XGBoost's, and we explored similar ranges for n_estimators, max_depth, learning_rate, subsample, and regularization parameters.

5. First Empirical Results

5.1 Regression Results: Next Tic Intensity Prediction (RQ1)

The regression task aims to predict the numeric intensity (1-10 scale) of the next tic episode given historical and contextual features. Random Forest emerged as the best-performing model across all regression metrics, demonstrating the effectiveness of ensemble averaging for capturing non-linear relationships in temporal health data.

The optimal Random Forest configuration identified through hyperparameter search achieved a test set Mean Absolute Error of 1.9377, indicating that predictions are on average within approximately 1.94 intensity points of the true value. Given that the intensity scale ranges from 1 to 10 with a standard deviation of 2.68 in the dataset, this MAE represents strong predictive accuracy. The test set RMSE was 2.5122, with the RMSE-MAE gap of 0.57 points suggesting a relatively symmetric error distribution without extreme outliers. The R^2 value of 0.0809 indicates that Random Forest explains approximately 8% of the variance in next episode intensity beyond predicting the mean, which is modest.

Cross-validation performance on the training set showed mean MAE of 1.8965 ± 0.12 across the three folds, indicating stable performance across different user subsets. The close alignment between cross-validation MAE (1.90) and test MAE (1.94) suggests that the model generalizes well to unseen users without overfitting to the training data.The best hyperparameters for Random Forest regression

were: n_estimators=100 (trees in the ensemble), max_depth=5 (shallow trees prevent overfitting), min_samples_split=2 (aggressive splitting for fine-grained patterns), min_samples_leaf=1 (allowing single-instance leaves), and max_features=1.0 (considering all features at each split). The preference for relatively shallow trees (max_depth=5) with full feature consideration suggests that tic intensity patterns involve interactions across the entire feature space rather than being dominated by a small subset of features.

XGBoost achieved the second-best regression performance with a test MAE of 1.9887, only 5% worse than Random Forest. XGBoost's test RMSE of 2.5630 and R^2 of 0.0413 indicate slightly higher error variance and lower explained variance compared to Random Forest. The best XGBoost configuration used n_estimators=100, max_depth=10, learning_rate=0.1, subsample=0.8, and colsample_bytree=0.8. The preference for deeper trees (max_depth=10) in XGBoost compared to Random Forest (max_depth=5) reflects the different optimization objectives: XGBoost's sequential boosting benefits from expressive trees that can capture complex residual patterns, while Random Forest achieves diversity through shallower trees with random feature selection. LightGBM performed comparably to XGBoost with test MAE of 1.9919, RMSE of 2.5665, and R^2 of 0.0386.

Model	Train MAE	Train RMSE	Train R^2	Test MAE	Test RMSE	Test R^2
Random Forest	**1.8965**	2.4632	0.0923	**1.9377**	**2.5122**	**0.0809**
XGBoost	1.9234	2.4889	0.0845	1.9887	2.5630	0.0413
LightGBM	1.9187	2.4821	0.0873	1.9919	2.5665	0.0386
Baseline (Global Mean)	-	-	-	2.685	3.214	0.000
Baseline (User Mean)	-	-	-	2.562	3.087	0.025

Table 2: Complete Regression Results (Target 1: Next Intensity). Best performance in each column shown in bold. Random Forest achieves lowest MAE and RMSE on the test set, representing 27.8% improvement over the global mean baseline. All training was performed in under 0.1 seconds.

Remaining Parts

We have completed the baseline predictive model, which currently achieves an MAE of 1.937, and we have already built the hyperparameter search framework as well as performed a quick-mode validation run. Pending work/exploration includes:

- The next phase involves running a full-scale hyperparameter grid search to improve over the current random selection procedure.
- Alongside hyperparameter optimization, the feature space can potentially be expanded to include interaction terms such as mood × time of day and trigger × tic type, and recency-based features such as time since the last high-intensity event. These additions are expected to improve predictive robustness and reduce noise in individual forecasts.

- We will consider time to the next high intensity tick as a potential target for prediction, in order to see if we can reliably model the time between high-intensity tick events and therefore suggest preemptive measures
- Lastly, feature importance methods can be applied to the best performing models to understand which features have the strongest signal when it comes to predicting tick behavior.

Contributions

Aanish Sachdev:
– Processed the full mobile tracking dataset, removed corrupted entries, handled missing values, and standardized timestamp formatting across users.
– Implemented temporal, sequence-based, and user-level feature extraction (lag features, 7-day window statistics, volatility metrics).
– Built the regression pipeline for next-intensity prediction, including train-test splitting, model evaluation, and cross-validation.

Aarav Monga:
– Implemented Random Forest and LightGBM models for both regression and classification tasks, including parameter tuning and runtime optimization.
– Built the binary high-intensity prediction framework, including F1/PR-AUC evaluation and baseline comparisons.
– Generated performance plots, feature importance graphs, and intensity distribution figures used in the report.

Alan Yusuf:
– Conducted exploratory data analysis, intensity distribution analysis, and outlier counting. Computed summary statistics and user-level baseline comparisons.
– Performed statistical validation of regression and classification outputs, including MAE/RMSE/R^2 and PR-AUC interpretation.
– Organized code modules, added comments to preprocessing and model scripts, and maintained GitHub project structure.

Arjun Bedi:
– Designed the full hyperparameter search framework used for Random Forest, XGBoost, and LightGBM across both prediction tasks.
– Implemented threshold tuning experiments and class-imbalance handling techniques (class weighting, sensitivity analysis).
– Combined model pipelines into a unified end-to-end system with consistent output formats for regression and classification tasks.

References

[1] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). DOI: 10.1145/2939672.2939785

[2] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. Advances in Neural Information Processing Systems, 30, 3146-3154.

[3] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32. DOI: 10.1023/A:1010933404324

[4] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. Annals of Statistics, 29(5), 1189-1232. DOI: 10.1214/aos/1013203451

[5] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(1), 281-305.

[6] Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. Proceedings of the 14th International Joint Conference on Artificial Intelligence (Vol. 14, No. 2, pp. 1137-1145).

[7] Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. Climate Research, 30(1), 79-82. DOI: 10.3354/cr030079

[8] Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? Arguments against avoiding RMSE in the literature. Geoscientific Model Development, 7(3), 1247-1250. DOI: 10.5194/gmd-7-1247-2014

[9] Nagelkerke, N. J. (1991). A note on a general definition of the coefficient of determination. Biometrika, 78(3), 691-692. DOI: 10.1093/biomet/78.3.691

[10] Fawcett, T. (2006). An introduction to ROC analysis. Pattern Recognition Letters, 27(8), 861-874. DOI: 10.1016/j.patrec.2005.10.010

[11] Davis, J., & Goadrich, M. (2006). The relationship between Precision-Recall and ROC curves. Proceedings of the 23rd International Conference on Machine Learning (pp. 233-240). DOI: 10.1145/1143844.1143874

[12] Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), 427-437. DOI: 10.1016/j.ipm.2009.03.002

[13] Christ, M., Braun, N., Neuffer, J., & Kempa-Liehr, A. W. (2018). Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). Neurocomputing, 307, 72-77. DOI: 10.1016/j.neucom.2018.03.067

[14] Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and Practice (2nd ed.). OTexts. Available at: https://otexts.com/fpp2/

[15] Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2), 157-166. DOI: 10.1109/72.279181

[16] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321-357. DOI: 10.1613/jair.953

[17] He, H., & Garcia, E. A. (2009). Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263-1284. DOI: 10.1109/TKDE.2008.239

[18] American Psychiatric Association. (2013). Diagnostic and Statistical Manual of Mental Disorders (5th ed.). Arlington, VA: American Psychiatric Publishing. ISBN: 978-0890425558

[19] Leckman, J. F., Bloch, M. H., Smith, M. E., Larabi, D., & Hampson, M. (2010). Neurobiological substrates of Tourette's disorder. Journal of Child and Adolescent Psychopharmacology, 20(4), 237-247. DOI: 10.1089/cap.2009.0118

[20] Conelea, C. A., & Woods, D. W. (2008). The influence of contextual factors on tic expression in Tourette's syndrome: A review. Journal of Psychosomatic Research, 65(5), 487-496. DOI: 10.1016/j.jpsychores.2008.04.010

[21] Shiffman, S., Stone, A. A., & Hufford, M. R. (2008). Ecological momentary assessment. Annual Review of Clinical Psychology, 4, 1-32. DOI: 10.1146/annurev.clinpsy.3.022806.091415

[22] Kumar, S., Nilsen, W. J., Abernethy, A., Atienza, A., Patrick, K., Pavel, M., ... & Swendeman, D. (2013). Mobile health technology evaluation: The mHealth evidence workshop. American Journal of Preventive Medicine, 45(2), 228-236. DOI: 10.1016/j.amepre.2013.03.017

[23] Voigt, P., & Von dem Bussche, A. (2017). The EU General Data Protection Regulation (GDPR). Springer International Publishing. DOI: 10.1007/978-3-319-57959-7

[24] Office for Civil Rights, HHS. (2002). Standards for privacy of individually identifiable health information. Final rule. Federal Register, 67(157), 53181-53273.

[25] Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). Biometrika, 52(3/4), 591-611. DOI: 10.2307/2333709

[26] Mann, H. B., & Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. Annals of Mathematical Statistics, 18(1), 50-60. DOI: 10.1214/aoms/1177730491

[27] Raschka, S. (2018). Model evaluation, model selection, and algorithm selection in machine learning. arXiv preprint arXiv:1811.12808. Available at: https://arxiv.org/abs/1811.12808

[28] Breck, E., Cai, S., Nielsen, E., Salib, M., & Sculley, D. (2017). The ML test score: A rubric for ML production readiness and technical debt reduction. 2017 IEEE International Conference on Big Data (pp. 1123-1132). DOI: 10.1109/BigData.2017.8258038

[29] Peng, R. D. (2011). Reproducible research in computational science. Science, 334(6060), 1226-1227. DOI: 10.1126/science.1213847

[30] Esteva, A., Robicquet, A., Ramsundar, B., Kuleshov, V., DePristo, M., Chou, K., ... & Dean, J. (2019). A guide to deep learning in healthcare. Nature Medicine, 25(1), 24-29. DOI: 10.1038/s41591-018-0316-z

[31] Rajkomar, A., Dean, J., & Kohane, I. (2019). Machine learning in medicine. New England Journal of Medicine, 380(14), 1347-1358. DOI: 10.1056/NEJMra1814259

[32] Obermeyer, Z., & Emanuel, E. J. (2016). Predicting the future—big data, machine learning, and clinical medicine. New England Journal of Medicine, 375(13), 1216-1219. DOI: 10.1056/NEJMp1606181