Predicting Problematic Internet Usage Among Children and Adolescents Using Physical Activity Data

Adam Liu (ayl62), Avery Chan (ac2978)
Nick Guo (ng495), Rishik Sarkar (rs2634)

December 15, 2024

1 Abstract

Although the Internet can greatly enhance well-being, unregulated use can morph into Problematic Internet Use (PIU), which is often linked to anxiety, depression, and other mental health risks [2]. In this work, we develop a machine-learning pipeline to predict PIU severity from physical activity data in children and adolescents (ages 5–22). By transforming continuous accelerometer (actigraphy) and fitness assessments from the Healthy Brain Network (HBN) dataset [1] into features correlating with a standardized Severity Impairment Index (SII), we demonstrate that movement-based, low-cost monitoring tools may serve as scalable early-warning systems for PIU. Our findings could supplement traditional diagnostics, potentially improving large-scale detection and intervention strategies.

2 Introduction

Addressing the critical need for improved PIU detection, this study leverages data from the Kaggle Child Mind Institute—Problematic Internet Use competition. Traditional mental health diagnostics, such as clinical interviews and self-report surveys, are effective but resource-intensive, limiting their scalability. By contrast, accelerometer readings and routine fitness assessments are widely accessible and cost-effective, offering objective metrics for mental health screening—particularly valuable when comorbidities (e.g., ADHD or mood disorders) complicate diagnosis.

We combine actigraphy data from the Healthy Brain Network (HBN) with established machine learning algorithms (e.g., XGBoost, LightGBM, TabNet) to predict PIU severity. This work constitutes *competing on a machine learning benchmark* and *applying ML methods to a real-world mental health dataset*. By systematically tuning and benchmarking these approaches, we aim to advance scalable, sensor-based early interventions that transcend traditional, resource-intensive diagnostics.

3 Method

To develop our predictive model, we applied several machine-learning techniques and preprocessing steps tailored to our dataset. Focusing on supervised learning, we selected models well-suited for structured data. Below, we outline each step of our data preprocessing, feature engineering, and the supervised learning models we employed.

3.1 Actigraphy Data Processing

- We first processed time-series accelerometer readings from 996 training and 2 test Parquet files in parallel using Python's `ThreadPoolExecutor`.

- Each file provided acceleration metrics (`X, Y, Z`), derived measures (e.g., `ENMO`, angular position), and contextual data (time of day, weekday, ambient light).
- We engineered additional features (nightly movement, sustained activity, circadian rhythm) and computed summary statistics (mean, std, min, max, median, skew) for each participant.
- This yielded processed datasets of shape `(996 × 128)` for training and `(2 × 128)` for testing.

3.2 Tabular Data Processing

- We loaded tabular data from two CSV files. Approximately 30.91% of the 3,960 training samples were missing SII values.
- After dropping rows with missing SII, the training dataset shape was `(2736, 80)`. The final test shape was `(20, 59)`.
- Categorical columns were one-hot encoded, ensuring compatibility with our ML workflow.

3.3 Feature Engineering

3.3.1 Merged Data and Additional Feature Creation

- We merged actigraphy and tabular data on participant `id`, combining activity metrics (e.g., `ENMO`) with anthropometric measures (BMI, height, weight).
- **Physical-Activity Interactions:** Created features linking BMI, height, and weight to movement patterns.
- **BIA-Related Ratios:** Engineered metabolic indicators (e.g., fat-to-weight ratio) showing a 0.42 correlation with SII.
- **Sleep-Activity Patterns:** Linked nighttime movement metrics with daytime activity.

3.3.2 Feature Importance and Selection

- We performed a three-tier feature importance analysis:
  - **Spearman Correlation**: Identified monotonic relationships with SII.
  - **Mutual Information**: Captured nonlinear dependencies.
  - **Random Forest**: Assessed predictive power with a 100-tree ensemble.
- A composite score retained features above thresholds of 0.1 (general) and 0.2 (PCIAT-specific).
- Remaining missing values were imputed via `KNNImputer` (`k=5`), leveraging both tabular and actigraphy features.

3.3.3 Normalization, PCA and Class Weights

- We applied `StandardScaler` to normalize features to zero mean and unit variance.
- Principal Component Analysis (PCA) reduced dimensions to five components, preserving 96.96% of the variance.
- Severe class imbalance (e.g., weight for `No PIU` ≈ 0.429 vs. `Severe PIU` ≈ 20.118) was addressed by computing inverse-frequency class weights.

3.3.4 Final Dataset:

- Training set: `(2736 × 5)`
- Test set: `(20 × 5)`

3.4 Supervised Learning Models

We experimented with several approaches well-suited for structured tabular data and deep architectures. Each model was configured with specific parameters to optimize performance:

- **XGBoost:** Chosen for its efficiency and strong performance on structured data. Configured with `n_estimators=100`, `learning_rate=0.1`, and `max_depth=5`. Its gradient-boosting framework effectively handles missing data and regularizes it to reduce overfitting.
- **LightGBM:** Similar to XGBoost but optimized for speed using a leaf-wise tree growth strategy. For consistency, we set `n_estimators=100`, `learning_rate=0.1`, and `max_depth=5`.
- **Random Forest:** A robust ensemble method that aggregates multiple decision trees to reduce variance. Configured with `n_estimators=100` and unrestricted depth, it captures complex patterns and remains straightforward for smaller datasets.
- **Logistic Regression:** Our baseline for interpretability and efficiency. We applied L2 regularization and used the `'lbfgs'` solver to mitigate overfitting, retaining insight into feature contributions.
- **MLP Classifier:** A Multilayer Perceptron with two hidden layers of 100 and 50 neurons, respectively, each using ReLU activations. Optimized by the `adam` solver, it captures nonlinear relationships essential for this dataset.
- **Support Vector Machine (SVM):** Effective in high-dimensional spaces, we tuned kernel types and regularization parameters via grid search. SVMs model complex, nonlinear decision boundaries well when hyperparameters are carefully chosen.
- **TabNet (`pytorch_tabnet`):** An attention-based architecture designed specifically for tabular data. It uses sequential decision steps and feature selection at each step, enabling interpretability. Key hyperparameters included the decision and attention dimensions (`n_d`=64, `n_a`=64) and `n_steps`=5.
- **TabTransformer (PyTorch):** A Transformer-based model that embeds tabular features and applies multi-head self-attention, capturing feature interactions end-to-end. We experimented with encoder depth, number of attention heads, and embedding dimensions for optimal performance.

3.5 Evaluation Metrics

We evaluated our models using two metrics:

- **Validation Accuracy:** The percentage of correct predictions on the validation dataset.
- **Quadratic Weighted Kappa (QWK):** Measures agreement between predicted and actual classifications, adjusted for chance. The formula for QWK is:

$$\kappa = 1 - \frac{\sum_{i,j} W_{i,j} O_{i,j}}{\sum_{i,j} W_{i,j} E_{i,j}},$$

  where O is the observed matrix, E is the expected matrix, and W is the weight matrix based on the squared difference between labels. We calculated QWK using scikit-learn's `cohen_kappa_score` with `weights='quadratic'`.

4 Experimental Setup/Results

4.1 Baseline Model Performance

We began by loading `train.csv` and `test.csv`. For baseline modeling, we dropped `id` and `sii` and integer-encoded `sii` into four classes (0–3). We then performed an 80–20 **stratified split** (`train_test_split`, `stratify=y`,

`random_state=42`) to maintain class distribution, and scaled numeric features using `StandardScaler`. No advanced feature engineering or preprocessing steps were used at this stage, establishing our baseline performance.

Subsequently, we integrated the actigraphy features discussed in Section 3.1, applied `KNNImputer` for missing data, introduced class weights to address imbalance, and ran PCA (five components) to reduce dimensionality. Table 1 compares these **baseline** results to the **post-preprocessing** performance.

Table 1: Model Performance Comparison Before and After Preprocessing

Model	**Baseline**		**After Preprocessing**	
	Accuracy	**QWK**	**Accuracy**	**QWK**
MLP Neural Network	92.70%	0.938956	97.26%	0.977208
Support Vector Machine	92.15%	0.934727	95.07%	0.959302
Logistic Regression	91.06%	0.926141	94.89%	0.957203
XGBoost	90.15%	0.916356	92.52%	0.936705
LightGBM	89.60%	0.912991	92.52%	0.936541
Random Forest	87.23%	0.888101	93.61%	0.945855

4.2 Hyperparameter Tuning

We used `GridSearchCV` or manual iteration to optimize hyperparameters. Each model was retrained on the stratified 80% training split, then validated on 20%. We tracked **Validation Accuracy** and **Quadratic Weighted Kappa (QWK)**. Key configuration changes are summarized below, referencing typical ranges from our code (`n_estimators` up to 300, `max_depth` up to 6, patience up to 50 epochs, etc.).

4.2.1 MLP Neural Network

- **Data Setup:** Stratified split, scaled features. Three hidden layers `(128,64,32)` with `ReLU` activation. Early stopping used a max of 200 epochs, halting if validation QWK plateaued for 10 epochs.
- **Key Adjustments:**
  - **Solver:** Switched to `lbfgs` for faster, deterministic convergence.
  - **Network Depth:** Increased layer width to capture more nuanced relationships observed in code logs.
- **Outcome:**
  - Accuracy: 0.9726 $\rightarrow$ 0.9818, QWK: 0.9772 $\rightarrow$ 0.9850

4.2.2 Support Vector Machine

- **Data Setup:** Employed `StandardScaler` and class weights to counter low sample counts for class 3.0.
- **Key Adjustments:**
  - `class_weight`: {0:0.429, 1:0.937, 2:1.810, 3:20.118}, proportional to inverse class frequency.
- **Outcome:**
  - QWK: 0.9593 $\rightarrow$ 0.9597

4.2.3 XGBoost

- **GridSearch Ranges:** `n_estimators` {100,200,300}, `max_depth` {4,5,6}, `subsample` {0.8,0.9}.
- **Final Chosen Hyperparams:**
  - `n_estimators`=300, `max_depth`=6, `subsample`=0.9
- **Outcome:**
  - Accuracy: 0.9252 → 0.9361, QWK: 0.9367 → 0.9457

4.2.4 LightGBM

- **GridSearch Ranges:** `n_estimators` {100,200}, `max_depth` {3,5}.
- **Final Chosen Hyperparams:**
  - `n_estimators`=200, `max_depth`=5
- **Outcome:**
  - Accuracy: 0.9252 → 0.9343, QWK: 0.9365 → 0.9441

4.2.5 Random Forest

- **GridSearch Ranges:** `n_estimators` {100,200,300}, unrestricted `max_depth`.
- **Outcome:**
  - Accuracy: 0.9361 → 0.9434, QWK: 0.9459 → 0.9519

Table 2 compares post-tuning performance. MLP obtained the highest accuracy (98.18%) and QWK (0.9850), while gradient boosting (XGBoost/LightGBM) and Random Forest also improved substantially. A quick look at confusion matrices (not shown for brevity) confirmed that Class 3 (Severe PIU) remained the most misclassified, reinforcing the dataset's imbalance challenges despite class-weight adjustments.

Table 2: Model Performance Comparison After Hyperparameter Tuning

Model	Validation Accuracy	QWK
MLP Neural Network	98.18%	0.984984
Support Vector Machine	95.07%	0.959710
Logistic Regression	94.89%	0.957203
Random Forest	94.34%	0.951894
XGBoost	93.61%	0.945660
LightGBM	93.43%	0.944093

4.3 Deep Learning Models

4.3.1 TabNet (`pytorch_tabnet`)

- **Implementation:** `TabNetClassifier` with attention-based feature selection steps.
- **Key Hyperparameters:**
  - `n_d`=64, `n_a`=64, `n_steps`=5, `lambda_sparse`=0.001
- **Training Protocol:**
  - `max_epochs`=1000, `patience`=50, batch size=1024. Early stopping triggered on validation accuracy.
- **Results:**
  - Accuracy: 94.53%, QWK: 0.9536

4.3.2 TabTransformer (PyTorch)

- **Architecture:**
  - Linear embedding (`embed_dim=32`) → 2 Transformer encoder layers (4 attention heads each).
  - Feedforward classifier for 4-class output.
- **Hyperparameter Ranges:**
  - `Encoder depth`: 1–3
  - `Dropout`: {0.0,0.1,0.2}, `Learning rate`: 1e-3 to 5e-4
  - `Batch size`: 64–512
- **Training Protocol:** Up to 1000 epochs, early stopping (`patience=50`) on QWK.
- **Results:**
  - Accuracy: 94.71%, QWK: 0.9561

Both TabNet and TabTransformer produced competitive validation scores but did not outperform our best MLP or ensemble models. Nevertheless, they show promise for large-scale tabular data and might improve further with refined hyperparameter searches or alternative embedding strategies.

5 Discussion and Prior Work

Deep learning architectures specialized for tabular data (TabNet, TabTransformer) did not outperform our previous ML baselines (MLP, XGBoost) on the validation set despite extensive hyperparameter tuning. Given TabNet's attention-based decision steps and TabTransformer's multi-head self-attention, we anticipated strong results. However, both models proved sensitive to the dataset's inherent noise and showed no clear advantage over simpler methods (e.g., MLP).

Moreover, although our MLP and tree-based ensembles achieved high validation scores (QWK > 0.9), performance dropped significantly on the Kaggle hidden test set: XGBoost/LightGBM hovered around 0.456–0.473 QWK, while TabNet reached only $\sim$0.463. These findings align with the competition leaderboard, where even the top models struggled to exceed 0.5 QWK. Since PIU severity is inherently ordinal, direct classification may further distort predictive accuracy. The prevalence of missing data and noise—highlighted by the dataset's curators—also posed substantial challenges and risked overfitting. Notably, boosting methods, while not tops on validation, showed relative robustness on unseen data, suggesting ensemble-based approaches may generalize better in noisy real-world scenarios. A brief ablation test further confirmed the impact of Sleep-Activity Patterns and BIA-Related Ratios, each removal causing a notable drop in QWK.

6 Conclusion

In this paper, we developed a machine-learning pipeline to predict Problematic Internet Use (PIU) severity from physical activity data in children and adolescents. Our experiments encompassed traditional ML techniques (Logistic Regression, Random Forest, XGBoost, LightGBM) and deep learning models (MLP, TabNet, TabTransformer). While the MLP achieved the best validation scores, significant performance drops on the hidden test set highlighted the complexity of real-world PIU detection and the limitations of direct classification for ordinal severity levels.

Promising next steps include exploring ordinal regression strategies, more robust validation schemes (e.g., nested cross-validation), advanced regularization or noise-robust training methods, and ensembles that combine both classical ML and deep learning architectures. Balancing model complexity against interpretability will remain critical for real-world applicability in mental health diagnostics, where data quality and consistency may vary significantly.

References

[1] Santorelli, A., Zuanazzi, A., Leyden, M., Lawler, L., Devkin, M., Kotani, Y., & Kiar, G. (2024). *Child Mind Institute—Problematic Internet Use* [Competition]. Kaggle. https://kaggle.com/competitions/child-mind-institute-problematic-internet-use

[2] Aboujaoude, E. (2010). Problematic Internet use: An overview. *World Psychiatry, 9*(2), 85–90. https://doi.org/10.1002/j.2051-5545.2010.tb00278.x