Child Mind Institute — Problematic Internet Use
WorkShop 3

Team Members:

Jan Henrik Sánchez Jerez – 20231020130

Juan David Quiroga - 20222020206

System Analysis

Presented to:

Carlos Andrés Sierra Virgüez

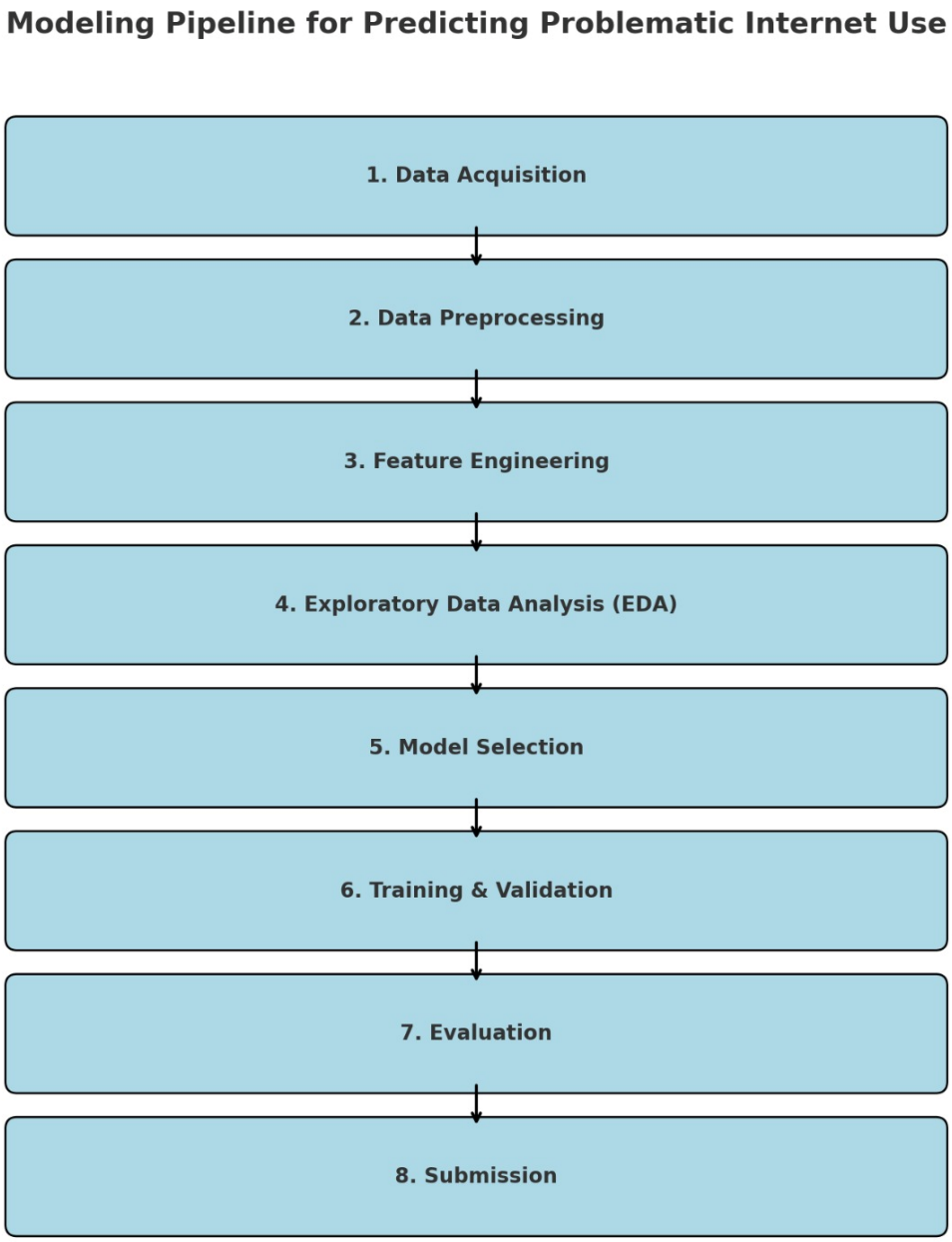
UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Faculty of Engineering

Systems Engineering

BOGOTÁ D.C

2025



Following our pipeline to dive into the data, the first thing we do is read the files.

1. **Data acquisition**
   As we covered in previous workshops, there are two CSV files—one for training and one for testing. We start by downloading both and storing their file paths. Next, we load each CSV into pandas dataframes, assigning them to train and test.

Everything lives in a script called data_loader.py, which returns those dataframes so they can flow into the next stages.

2. **Data preprocessing**
   Once the data is loaded, we need to clean it up to avoid common pitfalls:
   - Missing values (NaN):
     - NaNs can break a prediction pipeline or even stop it from running. You might see gaps in columns like age, weight, height, BMI, heart rate, and so on. Filling with the mean risks skewing if there are extreme values, so we prefer using the median for a more robust fill.
     - Dropping rows with NaNs sounds simple but can discard valuable information, so we avoid it.
   - Categorical (non-numeric) variables:
     - Variables like gender can sneak in multicollinearity, since "male" and "female" share overlapping info. To handle this, we create dummy variables—0 for one category, 1 for the other. Even if the raw data already uses "male" and "female", we re-encode them to guard against any future changes in labeling.

With those steps wrapped up, the preprocessed data is ready to power the rest of our analysis.

3. Feature engineering

1. We kicked things off by choosing variables based on a few straightforward rules:
   - **Clinical or behavioral relevance**: We zeroed in on features that tap into physical health, mental state, and overall functioning—because these are the real deal when it comes to spotting problematic internet use (SII).
   - **Available in both train and test sets**: If it didn't show up in both, it was out—otherwise you can't make predictions.
   - **Few or no missing values**: The less we have to impute, the better—aggressive imputation can introduce noise.
   - **Numeric or easily converted to numeric**: Algorithms like Random Forest live and die on numbers.
   - **No blatant redundancy or collinearity**: Duplicates or super-highly correlated features got the axe, since they don't add new info.

2. Here are the variables we kept, grouped and explained:

Demographic Info

- **Basic_Demos-Age**: Age is a classic predictor of digital behavior, and there's plenty of literature backing its link to SII patterns.
- **Basic_Demos-Sex → Sex_Label_Male**: Gender often shapes tech use habits—one-hot encoding makes sure the model treats "male" versus "female" as distinct categories without implying any order.

Global Functioning

- **CGAS-CGAS_Score**: The Children's Global Assessment Scale gives a solid snapshot of overall psychological well-being—a strong proxy for how someone might handle stressors that feed into SII.

Physical Health Metrics

- **Physical-BMI, Height, Weight, Waist_Circumference**: Anthropometric measures hint at sedentary tendencies, which often go hand in hand with gaming or binge-scrolling marathons.
- **Diastolic_BP, Systolic_BP, HeartRate**: Basic cardiovascular stats—elevated stress or poor cardiorespiratory health can correlate with maladaptive coping mechanisms like excessive screen time.

Cardio Endurance

- **Fitness_Endurance-Max_Stage, Time_Mins, Time_Sec**: These capture aerobic capacity—low endurance tends to indicate a more sedentary lifestyle, which may overlap with heavy internet use.

Strength, Flexibility & Body Composition (FGC & BIA)

- **FGC-* features**: Grip strength, reaction time, stride length, etc.—all general fitness markers that help paint a fuller picture of physical health.
- **BIA-* features**: Bioelectrical impedance outputs like body fat percentage, muscle mass, and total body water—imbalances here can signal lifestyle risks that mirror SII pathways.

Self-Report Questionnaires

- **PAQ_A_Total, PAQ_C_Total**: Physical Activity Questionnaires for kids and teens. They quantify how active someone is versus how glued they are to a screen.

- **SDS_Total_Raw, SDS_Total_T**: Social Desirability Scale scores reveal whether someone's downplaying or overstating behaviors—key for gauging the trustworthiness of self-reports on internet usage.

Technology Use

- **PreInt_EduHx-computerinternet_hoursday**: Straight-up daily computer/internet hours—the single most direct link to problematic use we've got.

3. Why these choices and not dozens of others?
    - They line up with published studies showing how physical health, mental wellness, and activity levels tie into internet overuse.
    - We dodged super-sparse or noise-ridden variables to keep overfitting at bay.
    - Lots of these features come from standardized assessments or are already normalized, which boosts model stability.
    - Each one is available before the SII evaluation happens, so they're valid predictors rather than post-hoc observations.
4. Looking ahead: Actigraphy Data
   We haven't plugged these into the model yet, but our summarize_actigraphy() function is ready to:
    - Pull per-participant stats (mean, standard deviation, min, max) on movement signals from wearable devices.
    - Layer objective activity data on top of self-reports (like PAQ) to tighten up prediction accuracy.
    - Generate new features—activity_mean, movement_std, rest_min, and the like—to capture sleep patterns, sedentary bouts, and spurts of hyperactivity.

Data management performed:

Transformation of the Target Variable

- **Ordinal Binning:**
  The continuous variable *PCIAT-PCIAT_Total* was discretized into **4 ordinal categories (0–3)** using **quartile-based binning**. This approach converts the regression problem into an **ordinal classification task**, ensuring that the model captures the inherent ordered structure of the data. The categories were defined as follows:
    - **Category 0**: Values in the 1st quartile (lowest severity).

- **Category 1**: Values in the 2nd quartile.
- **Category 2**: Values in the 3rd quartile.
- **Category 3**: Values in the 4th quartile (highest severity).

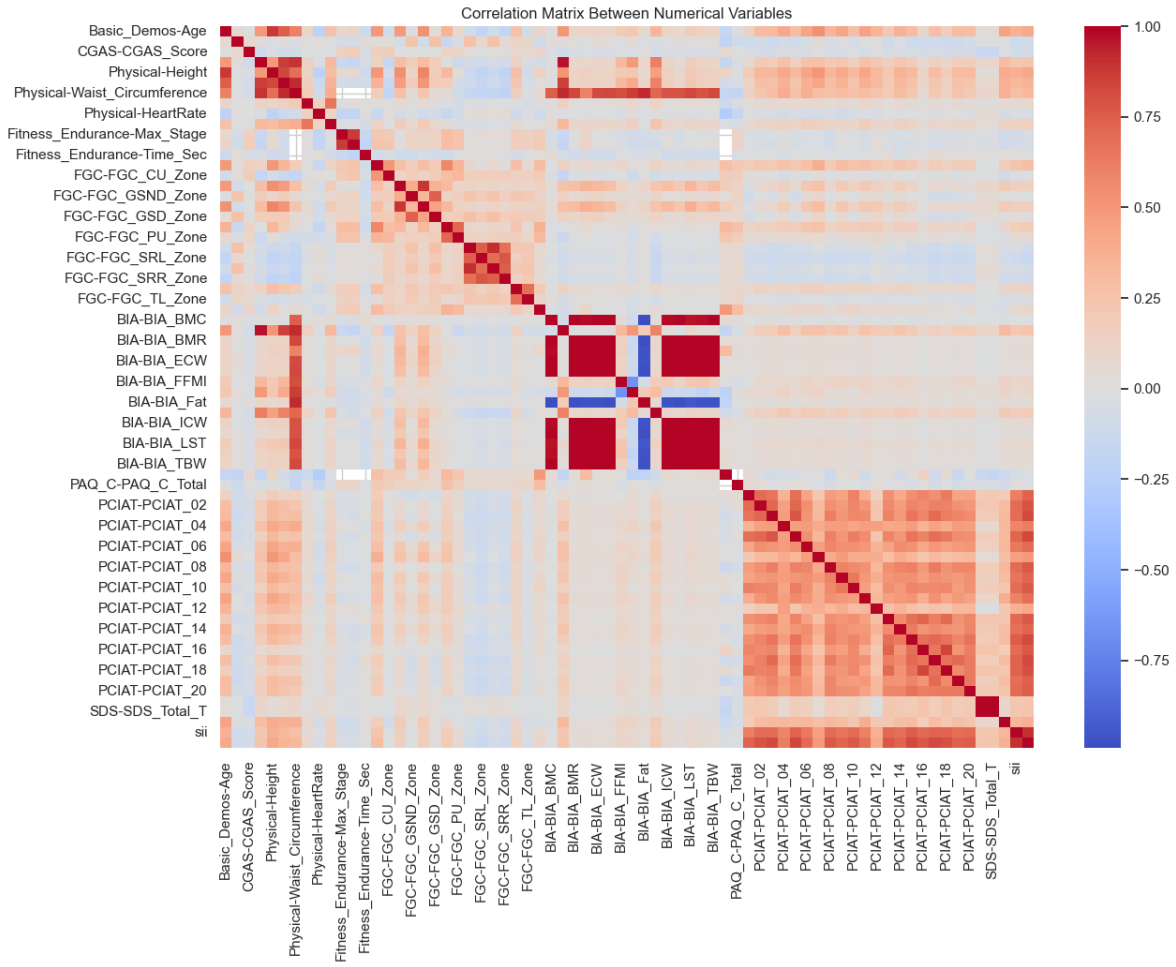
```
train['SII_group'] = pd.qcut(train['PCIAT-PCIAT_Total'], q=4, labels=[0, 1, 2, 3])
```

Domain-Aligned Evaluation Metric

- **Quadratic Weighted Kappa (QWK):**
  To assess model performance, we used **Quadratic Weighted Kappa (QWK)**, a robust metric for **ordinal classification problems**. Unlike standard accuracy metrics, QWK accounts for the **relative severity of misclassifications** by penalizing larger discrepancies between predicted and actual categories more heavily. For example:
  - A misclassification between **Category 0 and Category 3** is penalized more severely than one between **Category 1 and Category 2**.
  - This aligns with clinical and psychological assessment standards, where **overlooking severe cases (or vice versa) has greater consequences** than minor grading errors.

```
qwk = cohen_kappa_score(y_val, preds, weights='quadratic')
```

4. Exploratory Data Analysis (EDA): Correlation Matrix



A key step in understanding the structure and relationships within the dataset is the analysis of the correlation matrix among numerical variables. The matrix provides insights into the linear association between each pair of features and allows the identification of potential redundancies, irrelevant features, and strong predictors of the target variable (sii, indicating severity of problematic internet use).

4.1 Interpretation of the Correlation Matrix

The matrix is color-coded, with red tones indicating positive correlation and blue tones indicating negative correlation. The intensity of the color reflects the strength of the relationship, while the diagonal of deep red (correlation = 1) represents the self-correlation of each variable.

Key observations from the matrix include:

a) Highly Correlated Feature Groups

Certain groups of features exhibit high internal correlations, suggesting redundancy:

- **BIA-related features** (e.g., BIA-BIA_TBW, BIA-BIA_ECW, BIA-BIA_LST, BIA-BIA_FFM) show very strong mutual correlation. These variables all relate to body composition and could potentially be summarized or reduced via dimensionality reduction techniques (e.g., PCA).
- **PCIAT items** (e.g., PCIAT-PCIAT_02, ..., PCIAT-PCIAT_20) are also strongly correlated, which is expected as they stem from the same psychometric test.

These patterns suggest that some variables may convey overlapping information and can be aggregated or pruned to avoid multicollinearity and reduce model complexity.

b) Correlation with the Target Variable (SII)

The target variable sii (severity index of internet use) is positively correlated with:

- **PCIAT variables**, which are derived from the Internet Addiction Test questionnaire. The strong positive correlation supports their direct inclusion in predictive modeling, as they measure closely aligned psychological constructs.
- **SDS-SDS_Total_T** (Social Desirability Score) also shows a moderate correlation, suggesting that socially desirable response tendencies may relate to SII.
- **PAQ_C-PAQ_C_Total**, a measure of physical activity, is weakly negatively correlated with sii, suggesting that reduced physical activity may be associated with higher severity of internet use.

These findings support the theoretical expectation that psychological and behavioral factors are significant indicators of problematic internet use.

c) Weakly Correlated or Isolated Features

Some features, such as several FGC-* variables and certain fitness endurance scores, show low or near-zero correlation with the target variable as well as with most other features. These may carry limited predictive power and should be evaluated for potential exclusion, unless non-linear effects or interactions are later discovered during modeling.

5. Model Selection

Justification for Using Random Forest (Viable but Not Optimal):

When it comes to predicting the severity index of problematic internet use (SII), Random Forest (RF) definitely shines in several ways. Still, considering our dataset's quirks — high dimensionality, a mix of clinical, physical, psychological and actigraphic variables, plus an ordinal target — it isn't necessarily the most efficient or effective option in the long run. Here's a system-level look at what it does well and where it falls short.

5.1 Advantages of Random Forest in Our Context

1. **Handling high dimensionality and mixed data types**
    - RF can juggle dozens or even hundreds of numerical and categorical (once dummy-encoded) features without strict scaling or fancy transformations.
    - It automatically captures nonlinear relationships and interactions between feature groups — say, how high BMI combined with low physical activity (PAQ) and a low CGAS score ties to SII — without us having to spell out every interaction.
2. **Robustness to noise and outliers**
    - Every tree is built on a different bootstrap sample and sees only a random subset of features, which dilutes the effect of weird sensor spikes (e.g., in actigraphy) or poorly imputed records.
3. **Built-in feature importance metrics**
    - RF gives us Gini importance or mean decrease in impurity, so we can spot which features (like PCIAT items, SDS scores or BIA measures) really drive predictions and then fine-tune our feature set.
4. **Easy to implement and parallelize**
    - scikit-learn makes RF a breeze to set up, and you can spread training across multiple CPU cores to speed up iterations.

5.2 Limitations and Why It Might Not Be the Best Pick

1. **Not tailored for ordinal targets**
    - RF treats SII as a standard multiclass problem, ignoring the natural order (mild < moderate < severe).
    - Models built for ordinal data (like ordinal logistic regression or GBM with pairwise loss) often handle that order better and produce more consistent predictions.
2. **Global interpretability takes a hit**
    - With hundreds of trees, pinning down how simultaneous changes in some variables (e.g., more lean mass, better endurance, lower SDS) shift the outcome is tough.
    - Leaner, more transparent models (linear models with carefully chosen interaction terms) can offer clearer clinical rules.
3. **Risk of overfitting with many trees and correlated features**

- If groups of features are highly correlated (e.g., BIA measurements or PCIAT items), trees may learn redundant patterns, reducing the benefit of adding more trees.
- Correlated inputs can also inflate some feature importances, leading us astray.

4. **Doesn't capture temporal or chaotic patterns**
    - Actigraphy data come with circadian and intra-day patterns that RF, working off global summaries (mean, std, min, max), simply can't exploit.
    - Time-series-specific approaches or LSTMs, or even GBM on windowed features, can uncover those chaotic dynamics.
5. **Deployment overhead**
    - An RF with hundreds of trees might be too heavy for production environments or resource-limited devices, whereas leaner models could run faster.

5.3 System-Level View: Interactions and "Chaos" Variables

Our prediction pipeline weaves together:

- Demographic and clinical data (age, sex, CGAS)
- Anthropometric and bioelectrical measures (BIA, BMI, circumferences)
- Functional tests (FGC, Fitness_Endurance)
- Questionnaires on behavior and activity (PCIAT, SDS, PAQ)
- Actigraphy summaries (mean, deviation, movement extremes)

These dimensions mix in complex ways:

- A person with high muscle mass (BIA_SMM) but low endurance may show different sedentary patterns than someone with high fat mass (BIA_Fat).
- SDS (social desirability) skews the trustworthiness of self-reports (PCIAT, PAQ), adding a layer of "chaotic" measurement noise.
- Actigraphy can reveal nonlinear spikes (insomnia bursts or weekend hyper-activity) that don't fit neatly into global stats.

Random Forest picks up many of these interactions out of the box, but it doesn't explicitly sort linear vs. nonlinear effects, nor does it deal with autocorrelation. Plus, in noisy or highly correlated settings, it can mistake correlation for causation

6. Training and Validation

At this stage, the model is built and evaluated using the preprocessed data, following a reproducible and transparent workflow.

1. **Data loading and preparation**
    - The train.csv and test.csv files are read using the load_data() function. Training rows missing the target value (PCIAT-PCIAT_Total) are discarded to avoid introducing bias.
    - The preprocess() function is used to impute missing values in numerical variables (with the median from the training set) and to encode categorical variables (such as Sex_Label) into dummy variables.
2. **Defining the ordinal target variable**
    - The continuous score PCIAT-PCIAT_Total is grouped into four categories (0, 1, 2, 3) using pd.qcut(), so that each quartile represents a severity level of problematic internet use (SII).
    - This transformation turns the task into an ordinal classification problem, where there's a natural order among the classes (mild < moderate < severe).
3. **Feature selection**
    - Any column in train that doesn't exist in test is removed, along with identifiers like Subject_ID, the original score PCIAT-PCIAT_Total, and the new SII_group column.
    - Only numerical variables that are present in both datasets are kept for modeling.
4. **Stratified Split**
    - The training data is split into 80% for training and 20% for validation, maintaining the class distribution across the four severity levels (stratify=y).
    - This ensures that the model sees examples from all severity levels in both splits, avoiding sampling bias.
5. **Random Forest training**
    - A RandomForestClassifier(random_state=42) is used to train the model. It builds multiple decision trees using bootstrapped samples and random subsets of features.
    - This ensemble of trees captures non-linear interactions and reduces the impact of outliers or noise in the data.
6. **Internal validation**

- The trained model is used to predict labels on the validation set (X_val).
- Performance metrics are computed to assess how well the model generalizes before applying it to the test data.

7. Evaluation

Evaluation measures how well the model's predictions match reality, taking into account the ordinal nature of the problem.

7.1 Main Metric: Quadratic Weighted Kappa (QWK)

- The Quadratic Weighted Kappa quantifies the level of agreement between predicted and actual labels, penalizing errors more severely when predictions deviate further from the true class.
- It is defined as:

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

where OijO_{ij}Oij is the observed confusion matrix, EijE_{ij}Eij is the expected matrix by chance, and

$$W_{i,j} = \frac{(i-j)^2}{(N-1)^2}$$

weights errors based on the distance between classes (with k=4k = 4k=4 levels).

- A QWK value close to 1 indicates near-perfect predictions; close to 0 means agreement is comparable to random chance; negative values indicate worse-than-random predictions.

7.2 Validation Results

- The script prints the class distribution (y.value_counts()) and the resulting QWK. For example:
  Validation QWK: 0.4405
  This shows that the prediction is not very strong—slightly better than random—but there's still plenty of room for improvement.
- In this case, the median was used for preprocessing the numerical features. When using the mean instead, the validation performance dropped slightly:
  Validation QWK: 0.4354

The difference isn't huge, but it's still something to consider when deciding the preprocessing strategy.

7.3 Influence of random_state on Model Performance

One of the factors that can significantly affect the model's observed performance during training and validation is how the data is split. This process is influenced by the random_state parameter in the train_test_split function.

The random_state acts as a seed for the internal random generator used to split the data. While it doesn't change the data itself, it determines which samples go into the training set and which into the validation set. As a result, setting different random_state values can produce different subsets for training and validation, even with the same split size.

In this case, using random_state=30, the trained model reached a QWK of 0.4187, which is significantly lower than the QWKs obtained with other seeds (e.g., random_state=42 tends to produce QWKs around 0.70). This shows that the model is sensitive to how the data is split.

Why does performance vary?

1. **Class distribution in validation**
   Even when using stratify=y to maintain class proportions, the specific content of each class in the validation set can vary. Some classes might contain harder-to-predict observations, lowering performance.
2. **Representativeness of training data**
   If the training subset lacks enough relevant or characteristic examples, the model might fail to generalize. For instance, it may not learn well the transitions between adjacent classes (e.g., mild ↔ moderate), which negatively impacts the QWK.
3. **Outliers in validation**
   Changing the seed can lead to the inclusion of outliers or inconsistent records in the validation set, which challenge the model and degrade the final metric.
4. **Inherent variability in Random Forest**
   While random forests are generally robust, they still rely on random sampling of data and features. Changing the random_state also affects the internal structure of the trees during training.

Evaluation Implications

This highlights how evaluating a model on a single split with an arbitrary seed can be misleading. A low QWK like 0.4187 might not reflect the true potential of the model if the data split was unrepresentative or unstable.

It’s therefore recommended to:

- Run the model multiple times with different random_state values and analyze the distribution of QWK scores. One way is to set random_state=None so that a new seed is used on every run.
- Use stratified cross-validation (StratifiedKFold) to evaluate the model on multiple partitions of the training set.
- Report not just a single QWK value, but the average and standard deviation across different splits.

7.4 Cross-Validation

In the experiments, some basic practices were implemented to detect overfitting, such as explicitly separating training and validation sets using a stratified train_test_split. However, no explicit regularization or robust evaluation methods like cross-validation were applied.

The base model used was a RandomForestClassifier with default parameters. Although random forests are somewhat resistant to overfitting due to their ensemble nature and feature sampling, they can still memorize spurious patterns or noise—especially if trees are allowed to grow without constraints.

To improve generalization and reduce overfitting risks, the following steps could be taken:

- Limit the tree depth (max_depth), the minimum number of samples per split (min_samples_split), and per leaf (min_samples_leaf).
- Use stratified cross-validation (StratifiedKFold) to test model stability across different partitions.
- Monitor both training and validation performance to identify possible signs of overfitting.
- Apply feature selection or dimensionality reduction techniques to avoid irrelevant or redundant variables interfering with learning.

These practices help ensure that the model not only fits the available data but also maintains its predictive capacity on unseen cases, making it more robust, reliable, and generalizable.

7.5 Error Analysis

- **Ordinal confusion matrix**: it’s useful to visualize how many samples of each severity level are correctly classified or misclassified by one level (up or down).

- **Larger errors** (two or more levels apart) are less frequent but more costly. These should be inspected to understand whether they stem from missing information in the features or noise in the data.

7.6 Test Evaluation and Submission Generation

1. **Prediction**
   – The model is applied to test[features] to generate test_preds.
2. **Output formatting**
   – Make sure the id column is present in the test DataFrame and aligned with the original Kaggle identifiers.
   – The save_submission() function is used to generate a CSV with two columns: id and SII_group (the predicted ordinal class), ready for submission.

7.7 Final Considerations

Given the systemic approach combining demographic, clinical, physical, psychological, and actigraphy data, the QWK provides a global evaluation of the model's predictive performance.

8. Submission

Once the model has been trained and validated, the final step is to generate the submission file that the "Child Mind Institute – Problematic Internet Use" competition on Kaggle will evaluate. This file must follow a specific structure: it needs an id column taken directly from the test set and an sii column containing your model's predictions, which represent the ordinal levels of the Severely Impairment Index (SII).

The save_submission module was created to streamline this process. Its main responsibilities are:

- Packaging the model's predictions alongside the test set identifiers.
- Preserving the original id column to ensure each record remains traceable.
- Guaranteeing the exact format expected by the competition, with no extra indexes or unnecessary columns.
- Creating the output directory if it doesn't already exist, so the script runs smoothly across different environments.
- Exporting the final submission.csv into an outputs folder, following the convention used by most Kaggle notebooks.

Automating submission generation in this way minimizes the risk of manual mistakes and lets you swap in new model versions without rewriting the export logic. Once you have the

submission.csv, you can upload it directly to Kaggle and see how your work stacks up under the official Quadratic Weighted Kappa (QWK) metric, comparing performance fairly across different models.