Kaggle Competition - Problematic Internet Use

Chan, Lok Hei (20693700, lhchanar) SZE, Fung Ming (20866474, fmsze)
Kwong Chi Yan (20852978, cykwongaf) ZHAO, Mingjie (21161845, mzhaoau)

1 Background

In today's digital age, problematic internet use among children and adolescents is a growing concern. A better understanding of this issue is crucial for addressing mental health problems such as depression and anxiety. Child Mind Institute announces a Kaggle Competition to challenge competitors to develop a predictive model capable of analyzing children's physical activity data to detect early indicators of problematic internet and technology use. Our objective is to develop a predictive model using pyspark.

2 Cloud Computing Platforms

2.1 Kaggle Notebook

Kaggle Notebooks are cloud-based interactive coding environments that allow users to write and run Python. They provide easy access to various datasets, enable collaboration by allowing users to share notebooks, and come with pre-installed libraries. However, it does not have good support for pyspark.

2.2 Databricks

Databricks is a cloud-based data platform that integrates big data analytics and machine learning in a collaborative environment. Built on Apache Spark, enabling fast processing of large datasets.

3 Dataset Description

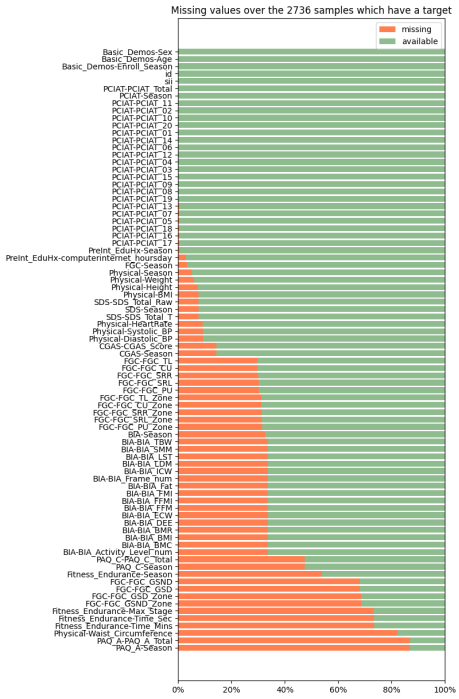
The competition data is compiled into two sources, parquet files containing the accelerometer (actigraphy) series and csv files containing the remaining tabular data. The majority of measures are missing for most participants. In particular, the target Severity Impairment Index (SII) is missing for a portion of the participants in the training set. The SII value is present for all instances in the test set.

4 Data Visualization

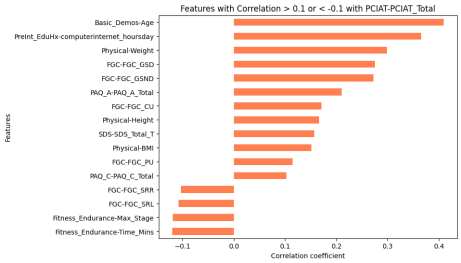
Data visualization allows us to assess data quality, structure, and distribution using intuitive graphs and figures. By using pre-installed libraries, we can easily identify patterns and insights within the data. This process enhances our understanding and enables effective data preprocessing and feature engineering in later stages.

4.1 Data Quality

The dataset contains 3960 training examples, out of which only 2736 are predictable with the target variable sii. Over half of the features have different levels of missing values. 13 features even have 50% missing values. Furthermore, 16 features have more than 0.1 correlation. Overall, the data quality is low. We have to do column removal and data imputation in the later stage.



(a) Percentage of Missing Value of Each Feature

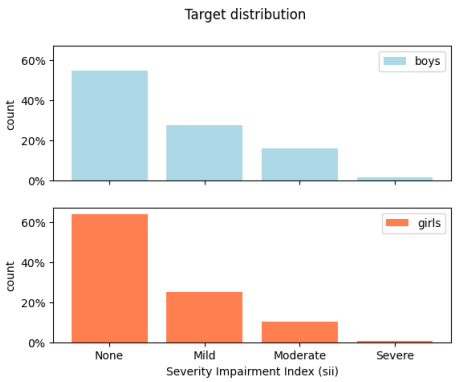


(b) Correlation

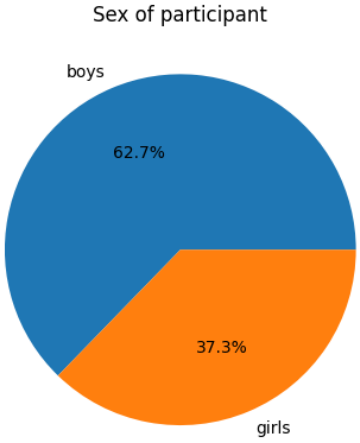
Figure 1: Data Quality

4.2 Data Distribution

The distribution of data shows that most of the data is imbalanced. Approximately 60% of the training data has SII of 0, and only about 1% of the data has SII of 3. This distribution is highly skewed. It might cause the model to tend to predict SII as 0, leading to underperformance in predicting higher SII levels. Besides that, most of the numeric columns are right-skewed, concentrating on low values. Last but not least, the male-female ratio is about 1.75:1. Overall, the data is imbalanced.



(a) Target Distribution



(b) Sex Distribution

Figure 2: Data Distribution

5 Data Preprocessing

We addressed missing values using KNN imputation. High-correlation features were dropped to mitigate multicollinearity. Imbalanced target distribution was handled using oversampling techniques

to ensure better model performance on minority classes.

6 Feature Engineering

The feature engineering process was designed to extract meaningful features from the data and enhance model performance:

6.1 Handling Missing Data and Redundant Features

- Columns related to "Season" were dropped as they were not relevant to the prediction task.
- Features with a high percentage of missing values were removed.
- Missing values in numeric columns were imputed using a **KNNImputer** to preserve data variability.

6.2 Generated Features

Additional features were created to capture relationships between variables:

- **BMI_Age**: Product of BMI and age.
- **Internet_Hours_Age**: Product of internet usage hours per day and age.
- **Hydration_Status**: Ratio of total body water to weight.
- **Muscle_to_Fat**: Ratio of skeletal muscle mass to fat mass.
- **LST_TBW**: Ratio of lean soft tissue to total body water.

6.3 Actigraphy Data Transformation

Time-series actigraphy data was processed into statistical features:

- Features such as **mean**, **standard deviation**, **minimum**, and **maximum** were extracted.
- Derived features included **MVPA time**, **sleep efficiency**, and **activity variability**.

6.4 Dimension Reduction

A neural network-based **Autoencoder** was used to reduce the 60 statistical features extracted from actigraphy data to 36 encoded features:

- The Autoencoder was trained with a ReLU-based architecture to capture latent representations.
- Encoded features were added to the dataset for improved training.

7 Model Training and Evaluation

The **Logistic Regression** model was implemented using PySpark's MLlib to predict the Severity Impairment Index (SII):

7.1 Pipeline

A pipeline was created to streamline the modeling process:

- Features were combined using `VectorAssembler`.
- StandardScaler normalized the feature set.
- Logistic Regression was applied to the scaled features.

7.2 Training and Validation Split

The dataset was split into 80% training and 20% validation sets to ensure robust evaluation.

7.3 Quadratic Weighted Kappa (QWK) Metric

The performance of the model was evaluated using the **QWK score**, a metric that accounts for ordinal relationships between classes:

$$\text{QWK} = \frac{\sum_{i,j} w_{i,j} O_{i,j}}{\sum_{i,j} w_{i,j} E_{i,j}}$$

where $w_{i,j}$ is the weight matrix, $O_{i,j}$ is the observed agreement, and $E_{i,j}$ is the expected agreement.

8 Results

The Logistic Regression model achieved a **QWK score of 0.4479** on the validation set, demonstrating moderate agreement between predicted and true values.

8.1 Confusion Matrix

The confusion matrix provided insights into model performance across SII classes:

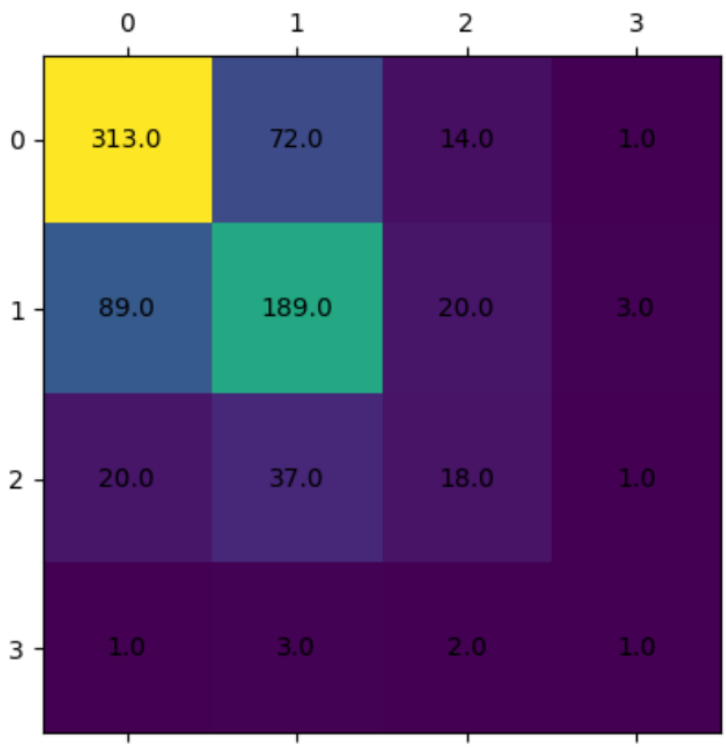


Figure 3: Confusion Matrix for Logistic Regression Model

8.2 Model Limitations

While the model performed well on the majority class, it struggled with higher SII levels due to class imbalance. This indicates potential improvement opportunities through class-weighted loss functions or data augmentation.

9 Discussion and Findings

9.1 Feature Importance

Feature engineering significantly improved model performance. Notably:

- **BMI_Age** and **Internet_Hours_Age** were strong predictors of higher SII levels.
- Encoded features from actigraphy data captured latent relationships not evident in raw time-series data.

9.2 Future Work

Future iterations of the model could explore:

- Ensemble models to handle class imbalance better.
- Advanced neural networks, such as recurrent models, for sequential actigraphy data.
- Hyperparameter tuning for the Autoencoder to enhance encoded feature quality.

9.3 Databricks Community Edition and Kaggle Notebook

When we look at Databricks Community Edition and Kaggle Notebooks, we see that each platform has its own strengths and limitation.

Databricks Community Edition is great for learning PySpark because of its strong connection to Apache Spark. However, we face some challenges. The file system are complex, requiring the use of Databricks Utilities (dbutils) to manage files. Additionally, we have to cold start the cluster each time we use it, which can take about half an hour, slowing down our training progress. Plus, the community edition runs on a single node, which limits its ability to fully use Spark's features.

In contrast, Kaggle Notebook is designed for data science competitions and offers a simple and user-friendly Jupyter Notebook environment. This makes it easy for users to quickly retrieve and work with files. However, it has a major limitation: Kaggle Notebooks aren't optimized for Spark, which makes it harder to train PySpark models and handle large datasets effectively. We have even faced memory issues and failed to train the PySpark models.

In summary, Databricks Community Edition is better for doing PySpark project, while Kaggle is easier to use for smaller projects that not using PySpark.

9.4 Comparison of Spark DataFrame and Pandas DataFrame

When we compare Spark DataFrames and Pandas DataFrames, we see that each has its own strengths and limitations.

Pandas DataFrames are great for small to medium-sized datasets that can fit into memory. They work quickly because they operate entirely in memory, allowing for fast data manipulation. Pandas provides many functions for filtering, grouping, and merging data, all within a simple and user-friendly interface. Also, it is well compatible with other libraries like Sklearn etc. However, a major drawback is that it can only handle data that fits within the available RAM.

In contrast, Spark DataFrames are designed for big data processing in distributed computing environments. They can manage large datasets that exceed memory limits by spreading the data across multiple machines. This is especially useful when working with big data tools like Hadoop and various data sources like HDFS and S3. However, Spark has fewer built-in functions than Pandas, and not compatible with many libraries. Some functions like StratifiedKFold cannot be used.It makes the training complicated.

Overall, both DataFrames are powerful tools. However, Pandas DataFrame is ideal for individual users like us, as Colab, Kaggle Notebook provide the coding environment with 16 GB RAM and we normally do not process very large datasets. A well-compatible DataFrame can make the training easy.