Child Mind Institute — Problematic Internet Use

CS114.P21.KHTN - Machine Learning

Nguyễn Đình Thiên Quang - 23521285
Đồng Quốc Thắng - 23521421
Nguyễn Thiện Nhân - 23521083

Faculty of Computer Science
University of Information Technology
VNU-HCM

May 21, 2025

Tables of contents

Table of Contents

Problem Description

Problem

Predict the **Severity Impairment Index (SII)** for problematic internet use by identifying biological markers to improve diagnosis and treatment of mental health and learning disorders.

- **Dataset**: ∼5,000 participants, aged 5–22, from the HBN study.
- **Competition**: Code-based with a hidden test set of ∼3,800 instances.

Web

Data Overview

Physical Activity Data

- Wrist-worn accelerometer (actigraphy) recordings.
- Fitness assessments, including FitnessGram and treadmill tests.
- Questionnaires on physical activity levels.

Internet Usage Data

- Surveys and Parent-Child Internet Addiction Test (PCIAT).
- SII derived from PCIAT: 0 (None), 1 (Mild), 2 (Moderate), 3 (Severe).

Tabular Data : CSV file

Key Instruments

- **Participant's ID**
- **Demographics**: Age, sex.
- **Internet Use**: Hours spent online daily.
- **Physical Measures**: BMI, heart rate, height, weight.
- **FitnessGram**: Cardiovascular fitness, strength, endurance.
- **Sleep Disturbance Scale**: Sleep disorder assessment.
- **PCIAT**: Internet addiction behaviors.

- **Note**: Many missing values; SII missing for some training data.

Actigraphy Data : parquet file

- **Source:** Collected from wrist-worn accelerometers worn up to 30 days continuously.

Key Fields:

- **X, Y, Z**: Acceleration in 3 axes (g)
- **enmo**: Euclidean Norm Minus One (motion intensity)
- **anglez**: Arm angle relative to the horizontal plane
- **non-wear_flag**: Indicates if device was not worn
- **light**, **battery_voltage**, **time_of_day**
- **relative_date_PCIAT**: Days relative to the PCIAT assessment

Challenges

Key Challenges

- **High missingness** in tabular data and SII values.
- Complex time-series actigraphy data processing.
- Need for robust feature engineering.

Opportunities

- Remove rows with excessive NaN values that cannot be reasonably imputed
- Apply supervised learning for missing data.
- Integrate physical activity and internet usage for better predictions.

Table of Contents

Remove rows with missing SII (target) values that cannot be reliably estimated or predicted

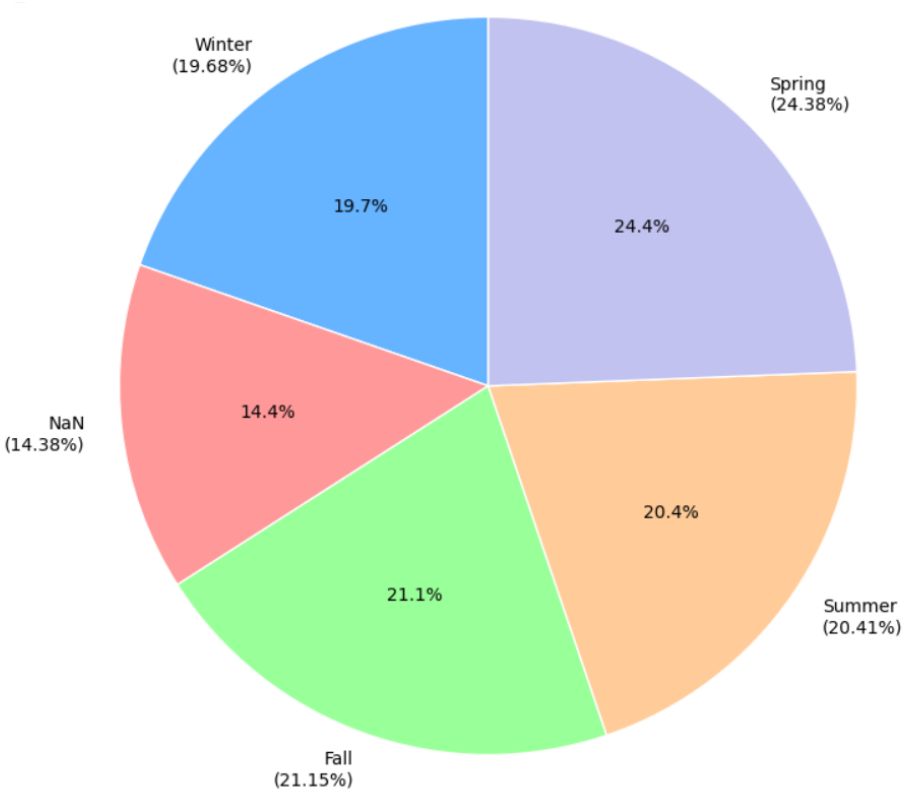
```
na_total_rows = train[train['sii'].isna()]
na_total_rows
```

IAT-T_15	PCIAT-PCIAT_16	PCIAT-PCIAT_17	PCIAT-PCIAT_18	PCIAT-PCIAT_19	PCIAT-PCIAT_20	PCIAT-PCIAT_Total	SDS-Season	SDS-SDS_Total_Raw	SDS-SDS_Total_T	PreInt_EduHx-Season	PreInt_EduHx-computerinternet_hoursday	sii	recalc_sii
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Fall	2.0	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Summer	2.0	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Spring	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	Spring	49.0	68.0	Spring	2.0	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Spring	0.0	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Winter	0.0	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Spring	1.0	NaN	NaN

Figure: 1224 row cannot be reliably estimated or predicted

Solution: deleting these rows.

Impute Missing 'Season' Values Using Existing Distribution



Solution: Since the temporal data depends on survey respondents, we can only impute missing values based on the existing distribution.

Figure: CGAS-Season

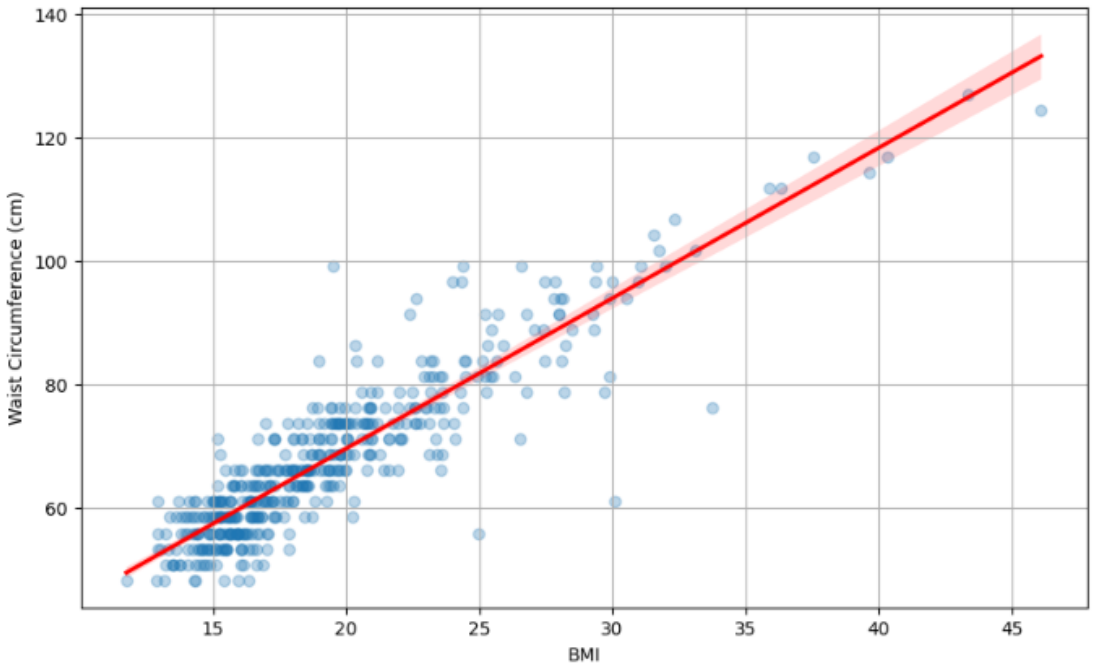
Physical Measures

	count	mean	std	min	25%	50%	75%	max	missing
Physical-BMI	2513.0	19.131766	4.914565	0.0	15.780022	17.823815	21.173921	46.102914	206
Physical-Height	2516.0	55.885688	7.389588	36.0	50.000000	55.000000	61.637500	78.500000	203
Physical-Weight	2557.0	87.859378	43.356403	0.0	57.200000	75.800000	111.400000	315.000000	162
Physical-Waist_Circumference	480.0	26.631250	5.225146	19.0	23.000000	26.000000	29.000000	50.000000	2239

Predict Missing Height and Weight with KNN

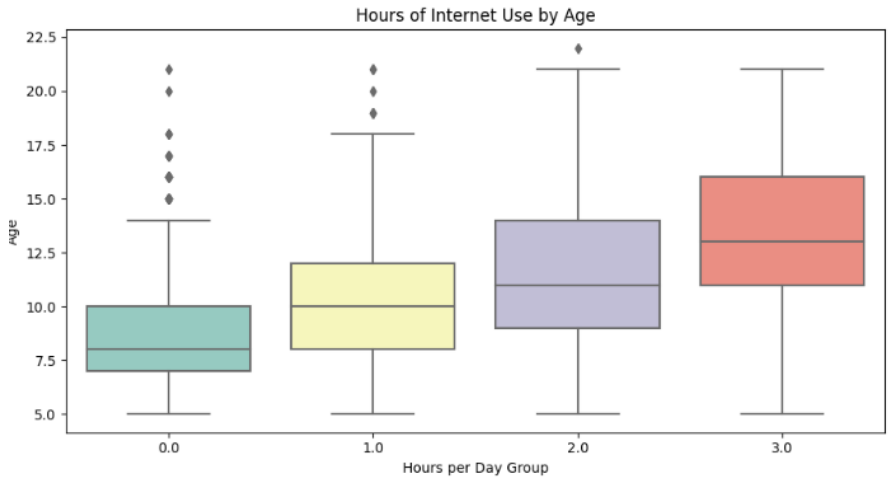
- It can handle **multidimensional and non-linear** data.
- Preserves the **natural variability** of the data.
- Leverages information from **multiple relevant variables** (season, age, gender).

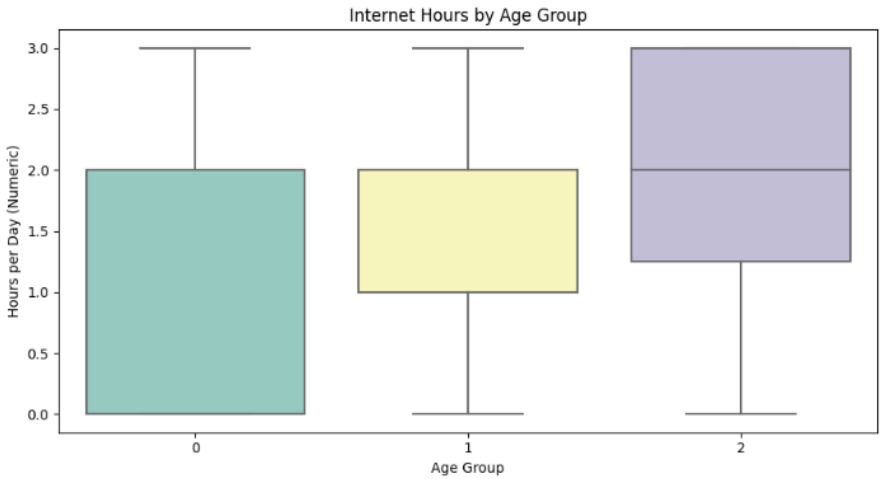
Predict Missing Waist Circumference Values with Linear Regression

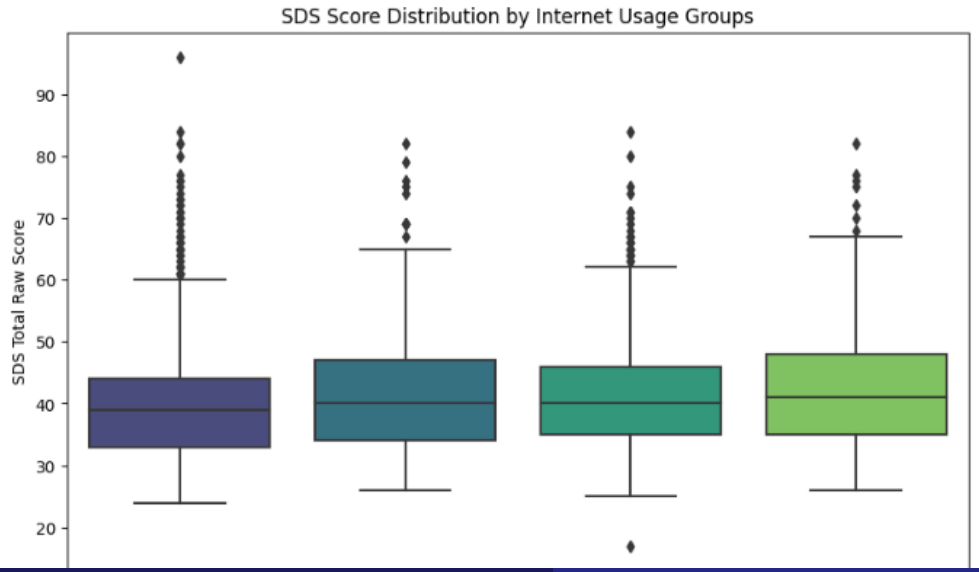


A **linear correlation** was observed between waist circumference and BMI. Predictive modeling was then performed using **linear regression** with both BMI and weight as independent variables.

Predict Missing computerinternet hoursday with logistic regression







Solution: Logistic regression was selected as the prediction model due to the **categorical** nature of the target variable and the presence of meaningful patterns in the related features

Manual Tuning and Feature Selection

- Hyperparameters were manually tuned based on:
  - Kaggle leaderboard scores.
  - Insights from prior experiments and relevant literature.
- Key tuned hyperparameters include:
  - Learning rate (0.001) , number of training epochs (150 - 200), lamda (5 - 10)
- After extensive experimentation, the following 5 input features were selected:
  1. `feature_1` — SDS_Score_Weighted
  2. `feature_2` — Internet_Hours_Age
  3. `feature_3` — PhysicalHeight_Age
  4. `feature_4` — Basic_Demos-Sex
  5. `feature_5` — PreInt_FGC_CU_PU

Introduction

Next, we concentrate on choosing suitable methods and partially show why we choose them. Overall, as Gradient Boosting Decision Tree (GDBT) framework has been dominant in the field of tabular data, we adopt two implementations of it with different philosophy, namely the XGBoost and LightGBM algorithms, to solve our problems.

Preliminaries

- Gradient boosting libraries (XGBoost, LightGBM) are widely used for structured data.
- Both focus on speeding up training and improving accuracy of tree ensembles.
- We examine their novel algorithmic techniques and mathematical foundations.
- Emphasis on each library's contributions and performance differences.

Gradient Boosting Overview

- Ensemble of trees: predictions updated as $\hat{y}^{(t)} = \hat{y}^{(t-1)} + f_t(x)$.
- Use gradients $g_i = \frac{\partial L}{\partial \hat{y}_i}$ and Hessians $h_i = \frac{\partial^2 L}{\partial \hat{y}_i^2}$ for optimization.
- Each new tree f_t minimizes an approximate objective (see block below).
- Splitting decisions maximize reduction in loss using accumulated gradients in leaves.

General Objective

$$Obj = \sum_i L(y_i, \hat{y}_i) + \sum_{t=1}^{T} \Omega(f_t).$$

XGBoost: Key Innovations (1/2)

- **Regularized Objective:** uses $\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_j w_j^2$ (leaf count T, weights w_j) to penalize complexity.
- **Shrinkage (Learning Rate):** multiplies new tree outputs by $\nu < 1$ to reduce each step's impact.
- **Column Subsampling:** sample a fraction of features per tree/level to speed training and reduce correlation.
- **Sparsity-aware:** handles missing or zero values by learning optimal default directions for splits.

XGBoost Objective (t-th Tree)

$$Obj^{(t)} = \sum_i l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t),$$
$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_j w_j^2.$$

XGBoost: Key Innovations (2/2)

- **Weighted Quantile Sketch:** approximates feature quantiles for fast split finding on large/weighted data.
- **Split Gain Formula:** computes loss reduction using gradient sums $G = \sum g_i$, Hessians $H = \sum h_i$.
- **Second-order Optimization:** uses Hessians to compute optimal leaf weights (Newton step).
- **Distributed/GPU Training:** optimized C++ implementation with multi-threading and GPU support for large-scale learning.

Split Gain (XGBoost)

$$Gain = \frac{1}{2}\left(\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right) - \gamma,$$

where G_L, G_R and H_L, H_R are gradient/hessian sums of left/right child.

LightGBM: Key Innovations (1/2)

- **Leaf-wise Growth:** splits the leaf with maximum loss reduction (best-first), yielding potentially deeper trees.
- **Histogram Binning:** discretizes continuous features into buckets, reducing memory and speeding up split search.
- **Exclusive Feature Bundling (EFB):** merges mutually exclusive sparse features into one to reduce dimensionality.
- **Gradient-based One-Side Sampling (GOSS):** keeps high-gradient instances; detailed on next slide.

LightGBM: Key Innovations (2/2)

- **GOSS Sampling:** keep the top $a\%$ of instances with largest gradients, randomly sample $b\%$ of the rest.
- **GOSS Weighting:** multiply gradients of sampled small-gradient instances by $\frac{1-a}{b}$ to maintain unbiased distribution.
- **Leaf-wise vs Level-wise:** more accurate splits but deeper trees; controlled by parameters `num_leaves` and `max_depth`.
- **Efficient Implementation:** multi-threaded, GPU (histogram-based) support, and native categorical feature handling.

GOSS Sampling Weights

Keep top $a\%$ by $|g|$, sample $b\%$ of others; multiply sampled gradients by $\frac{1-a}{b}$.

Mathematical Formulations (1/2)

- **Taylor Expansion:** for a new tree f_t, approximate loss by $\sum_i [g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2] + \Omega(f_t)$.
- **Optimal Leaf Weight:** for leaf with gradient sum G and Hessian sum H: $w^* = -\frac{G}{H+\lambda}$.
- **Learning Rate:** scales leaf outputs by ν, effectively $w^* \leftarrow \nu w^*$.
- **Gain-based Splitting:** XGBoost uses the above gain formula to choose splits.

Optimal Leaf Weight

$w^* = -\frac{\sum_{i \in I} g_i}{\sum_{i \in I} h_i + \lambda}$, for instances I in a leaf.

Mathematical Formulations (2/2)

- **Histogram Splitting (LightGBM):** continuous features binned; compute $G_b = \sum_{i \in \text{bin}} g_i$, $H_b = \sum_{i \in \text{bin}} h_i$ per bin.
- **GOSS Reweighting:** after sampling, use scaled gradients $(1 - a)/b$ in aggregated sums for splitting.
- **EFB Concept:** exclusive features are offset and treated as one feature (no overlap in non-zero entries).
- **Leaf-wise Gain:** always split the current leaf with highest gain (global max reduction).

Histogram Aggregation

Aggregate gradients/hessians per bin: $G_b = \sum_{i \in bin} g_i$, $H_b = \sum_{i \in bin} h_i$. Use these to evaluate each potential split.

Comparison: XGBoost vs LightGBM

Aspect	**XGBoost**	**LightGBM**
Training Speed	Slower (level-wise trees)	Faster (histogram & GOSS optimizations)
Memory Usage	Higher (full gradients, quantile sketch)	Lower (histogram bins, feature bundling)
Tree Structure	Level-wise (balanced)	Leaf-wise (deeper)
Split Finding	Exact or sketch method	Histogram-based (approximate)
Sampling	Row/column subsampling	GOSS (rows), EFB (columns)
Regularization	Explicit L1/L2 leaf penalties	Implicit L2 on weights (no leaf count penalty)
Categorical Feats.	Requires one-hot encoding	Native categorical support
Parallelism	Multi-threaded, GPU, distributed	Multi-threaded, GPU (histogram), distributed

Implementation Considerations

- **Preprocessing:** LightGBM accepts categorical features natively (specify as categories); XGBoost requires one-hot or label encoding.
- **Hyperparameters:** XGBoost uses `max_depth`, `eta`, `subsample`, etc.; LightGBM uses `num_leaves`, `learning_rate`, `feature_fraction`, etc.
- **GPU vs CPU:** Both support GPU acceleration; XGBoost's GPU uses exact/hist split methods, LightGBM's GPU uses histogram.
- **Distributed Training:** XGBoost integrates with Spark/Dask; LightGBM uses MPI or direct network communication for parallel learning.

Key Takeaways

- XGBoost and LightGBM are both powerful GBM implementations with complementary strengths.
- XGBoost emphasizes regularized objectives and precise split finding (good for robustness).
- LightGBM emphasizes training speed and scalability (good for large datasets).
- LightGBM often outperforms on very large data; XGBoost offers finer control via regularization.
- Choice depends on data size, sparsity, and performance vs. accuracy requirements.

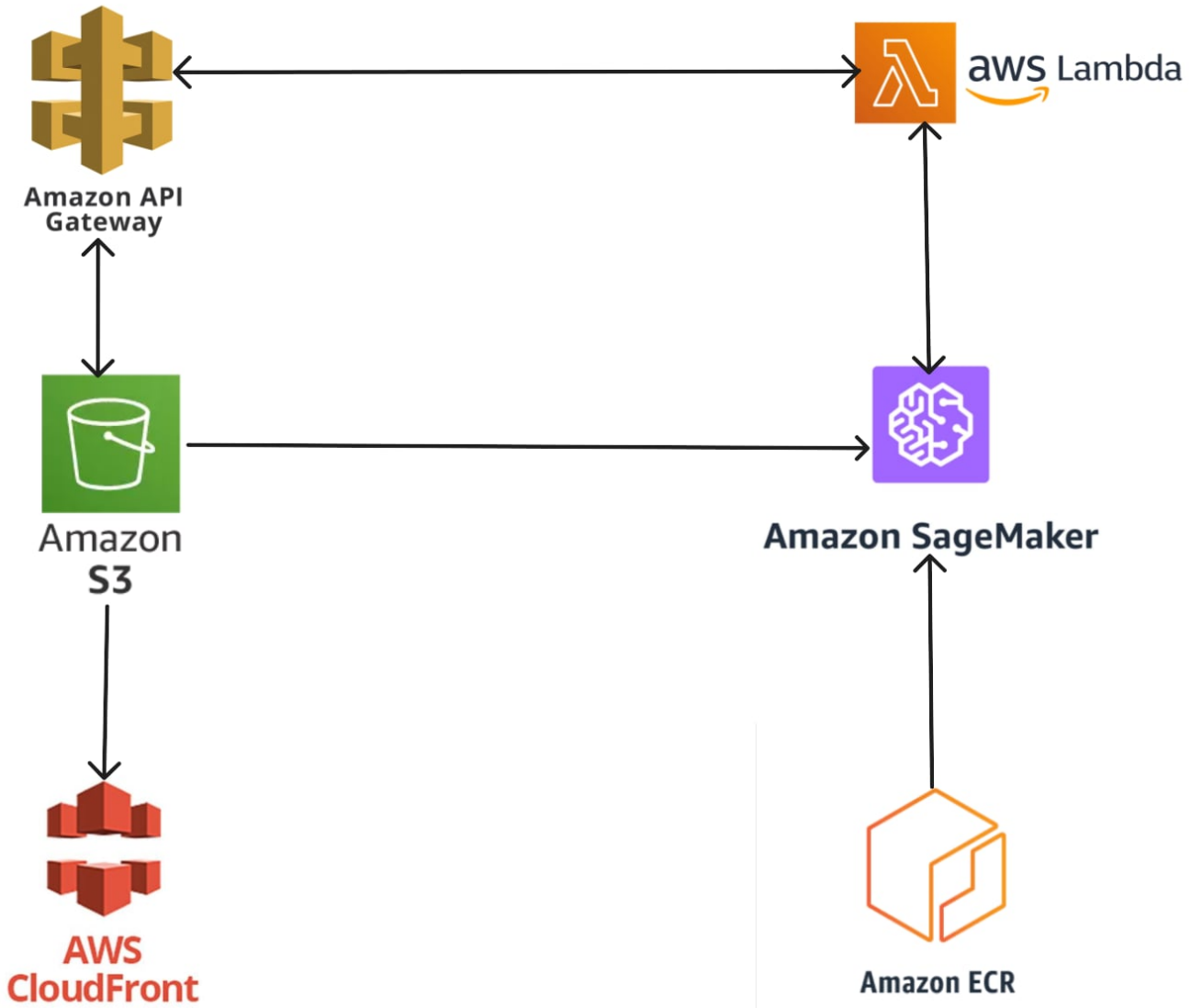
Table of Contents

Comparison of Model Performance

Model	Implementation	Private Score	Runtime
LightGBM	Library	0.476	2m 34s
XGBoost	Library	0.472	2m 22s
LightGBM	From scratch	0.466	1h 31m 23s
XGBoost	From scratch	0.455	38m 52s

- Custom implementations provide deeper insight but are computationally expensive.

Bonus: AWS serverless inference architecture



Team Contribution Summary

Name	Student ID	Contribution (%)
Nguyễn Đình Thiên Quang	23521285	33.33%
Đồng Quốc Thắng	23521421	33.33%
Nguyễn Thiện Nhân	23521083	33.33%

All members contributed equally to the project in terms of workload and responsibilities.