CHILD MIND INSTITUTE

PROBLEMATIC INTERNET USE

TEAM MEMBER

Ha Tien Dong

22028111

Nguyen Thi Thanh Nhan

22028114

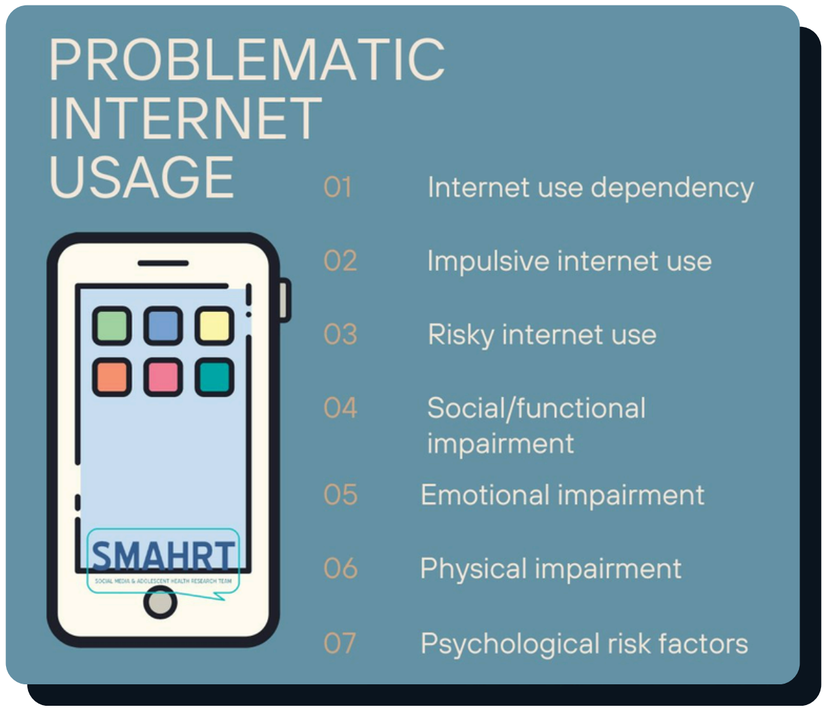
Nguyen Minh Quan

22028290

TABLE OF CONTENTS

1. Introduction
2. Exploratory data analysis
3. Data preprocessing
4. Training function
5. Model Improvement

INTRODUCTION



Problematic internet use among children and adolescents is a growing concern.

It has been increasingly associated with adverse mental health outcomes, including depression, anxiety, and social withdrawal.

INTRODUCTION

This competition challenges us to develop a predictive model capable of analyzing children's physical activity data to detect early indicators of PIU. The goal is to predict from this data a participant's Severity Impairment Index (sii)

Child Mind Institute

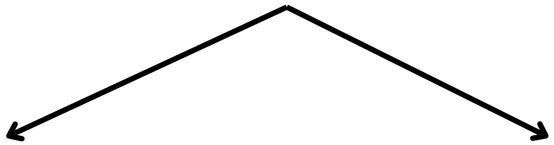
kaggle

EXPLORATORY DATA ANALYSIS

The data is derived from the Healthy Brain Network (HBN) study, which seeks to identify biological markers for mental health and learning disorders among children and adolescents.

EXPLORATORY DATA ANALYSIS

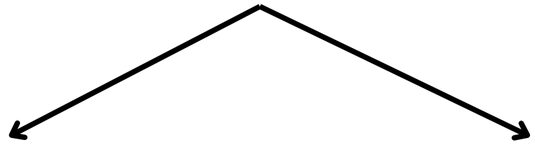
two primary data sources



physical activity data

internet usage behavior data

two key formats



tabular data (csv file)

time series data (parquet file)

TABULAR DATA

train.csv: 3960 examples, 82 features

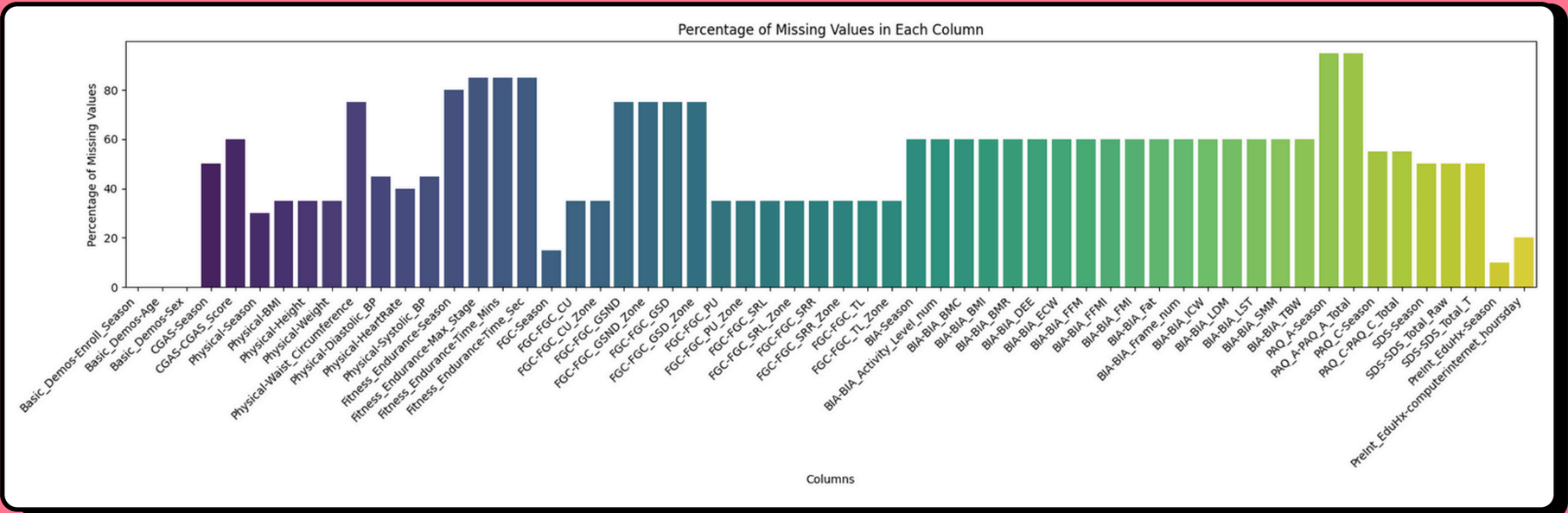
test.csv: 20 examples, 59 features

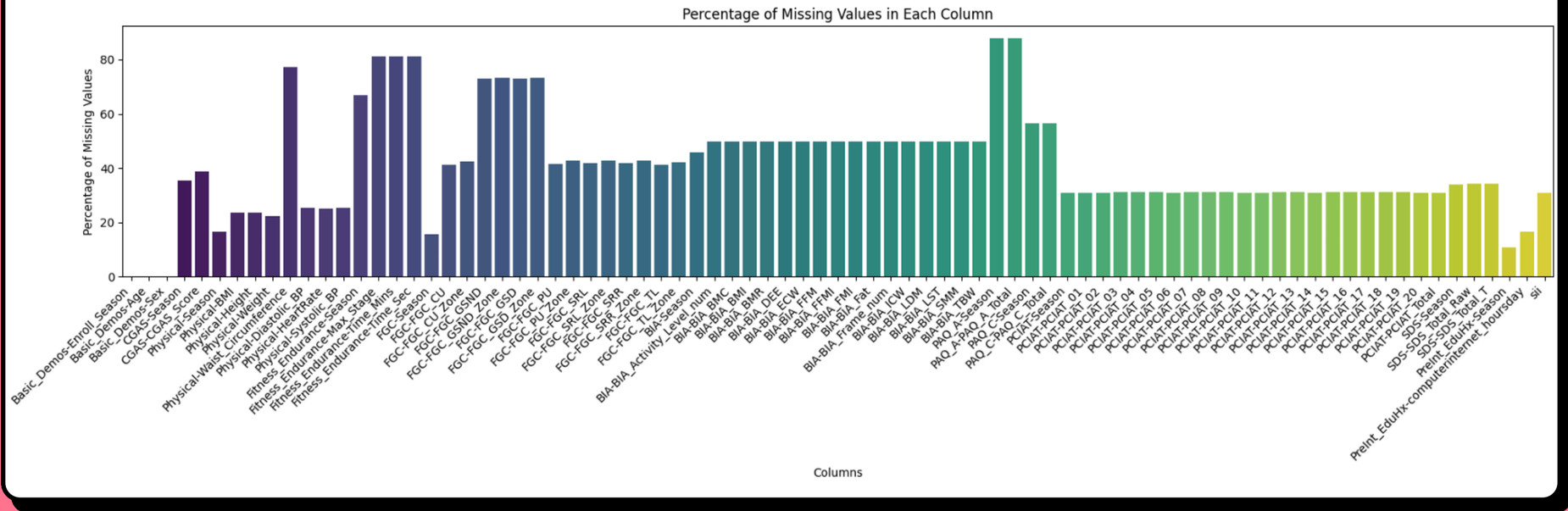
missing data

Comment: The features related to the target variable that are not present in the test set:

- PCIAT-Season: Values = Spring, Summer, Fall, Winter
- PCIAT-PCIAT_01 -> PCIAT-PCIAT_20: Values = 0,1,2,3,4,5
- PCIAT-PCIAT_Total
- Sii

Percentage of missing value in each column in the test dataset

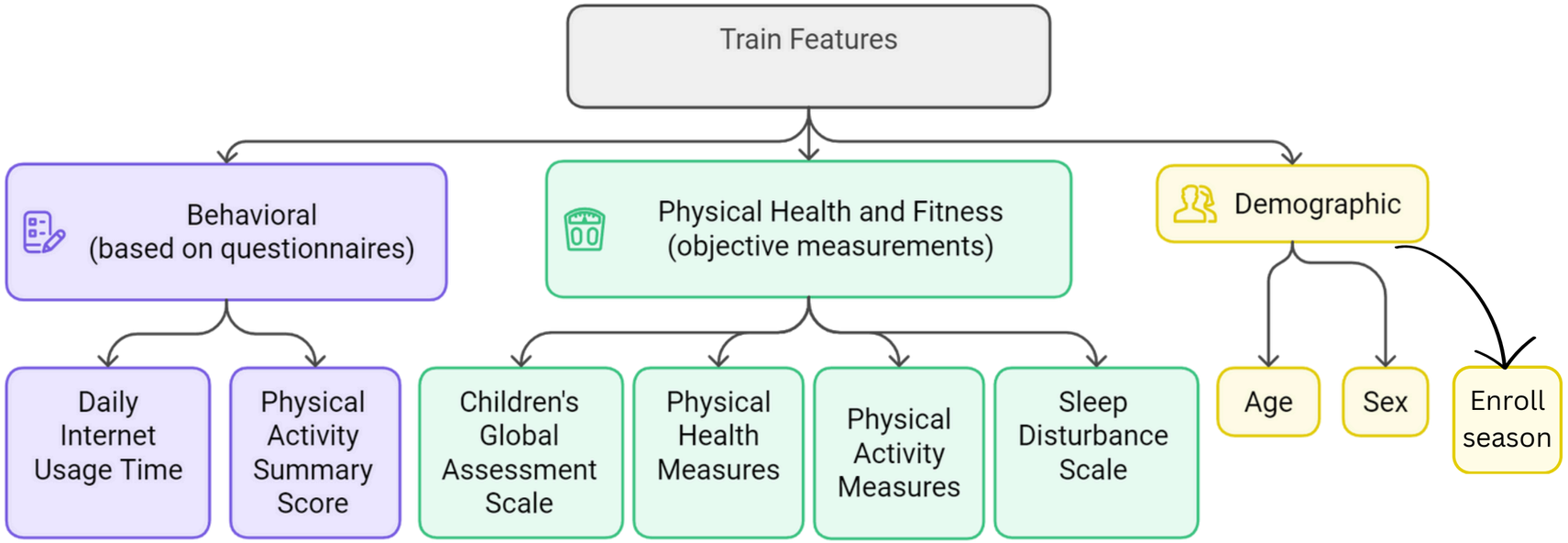




Percentage of missing value in each column in the train dataset

TABULAR DATA

Groups of features in the train data



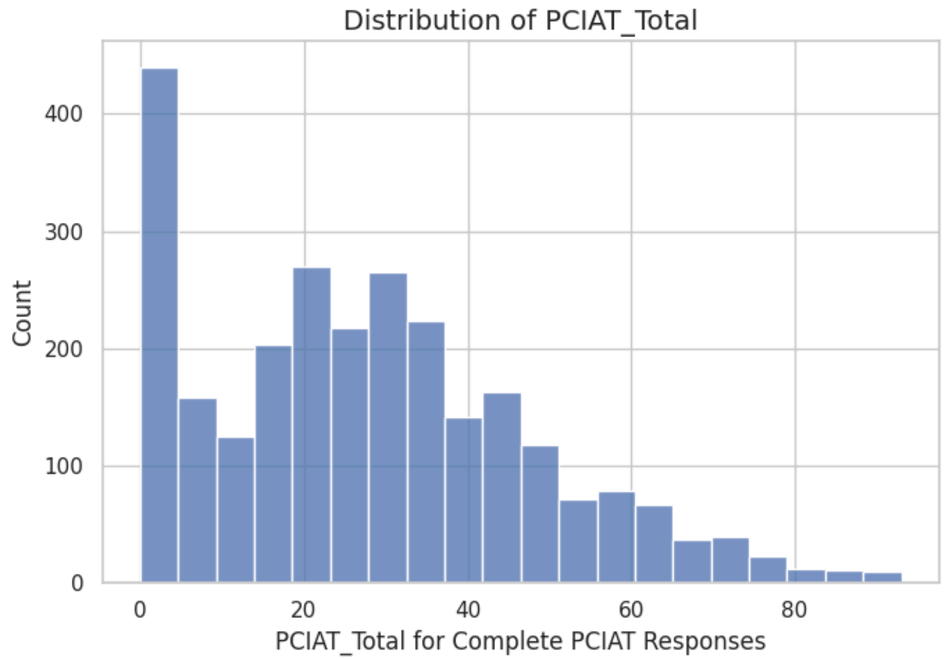
TABULAR DATA

Detailed analysis of some features

- Total Score (PCIAT-PCIAT_Total)
- SII Scores
- Internet Use (PreInt_ EduHx-computerinternet_ hoursday)
- Sex of Participant (Basic_Demos-Sex)
- Age group (Basic_Demos-Age)
- Correlations between the 58 features and the SII scores

TABULAR DATA

Overall distribution of PCIAT scores

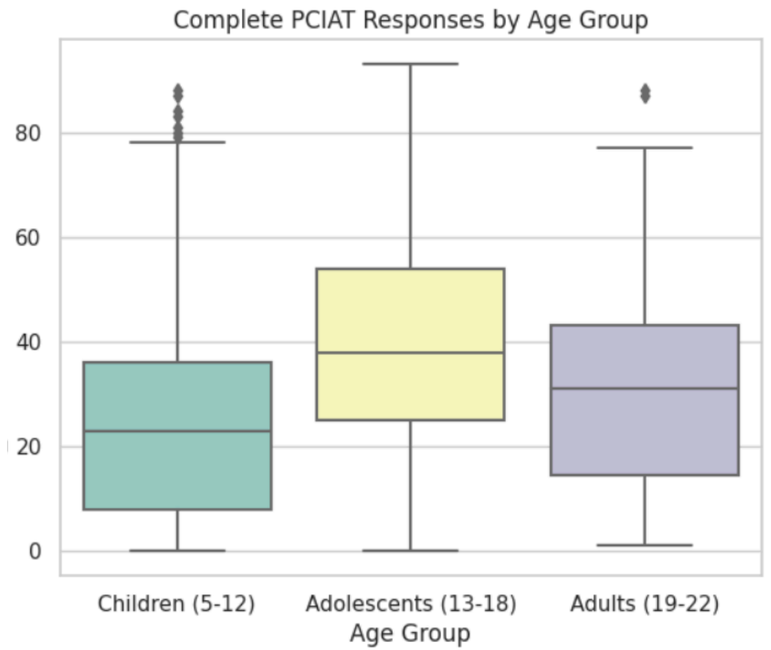


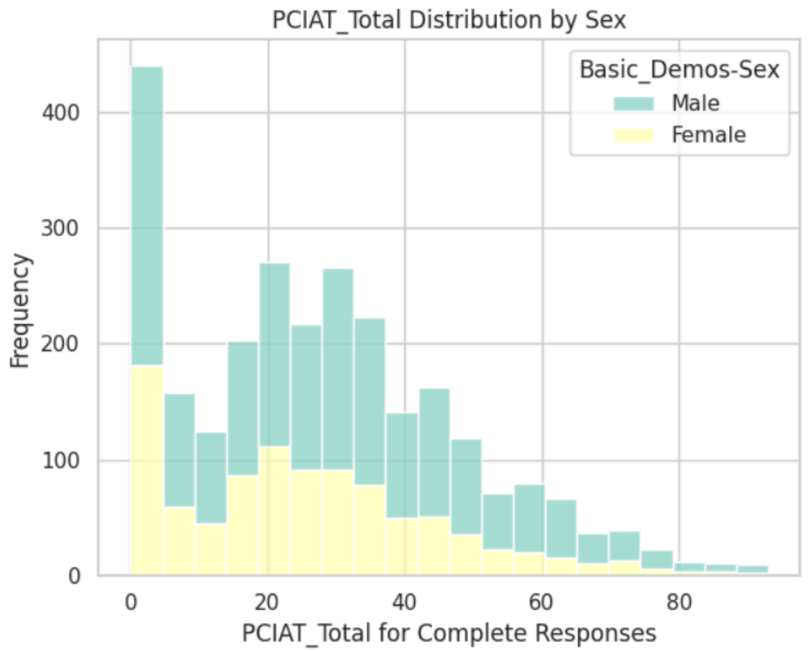
Right-skewed distribution: indicating most individuals have lower PCIAT_Total scores.

The frequency of responses gradually decreases as scores increase.

TABULAR DATA

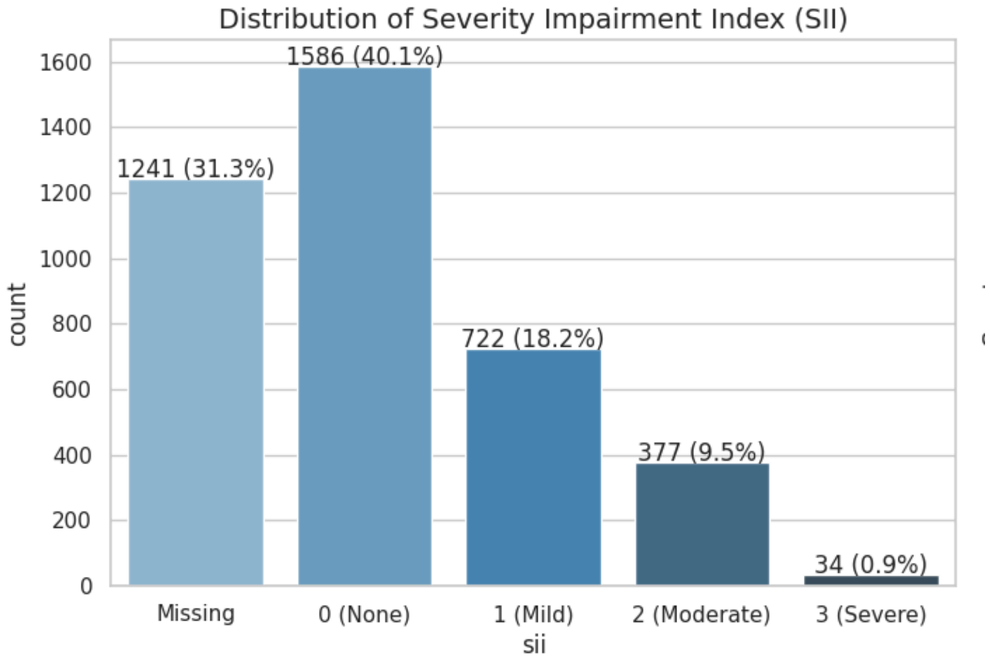
Total Score





TABULAR DATA

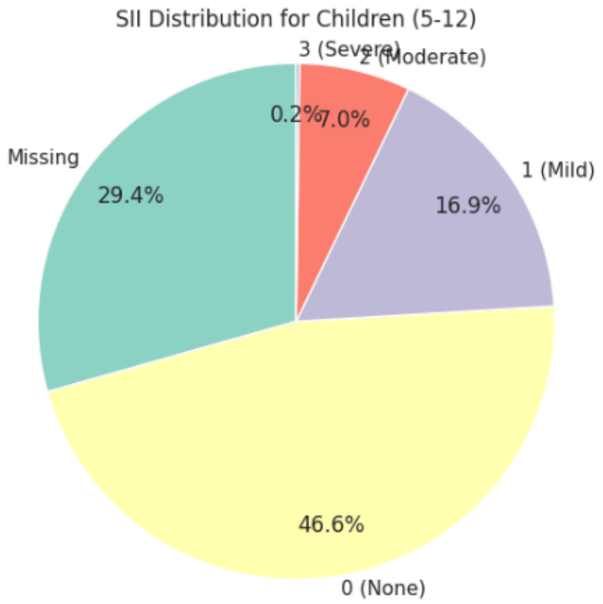
Overall distribution of SII scores

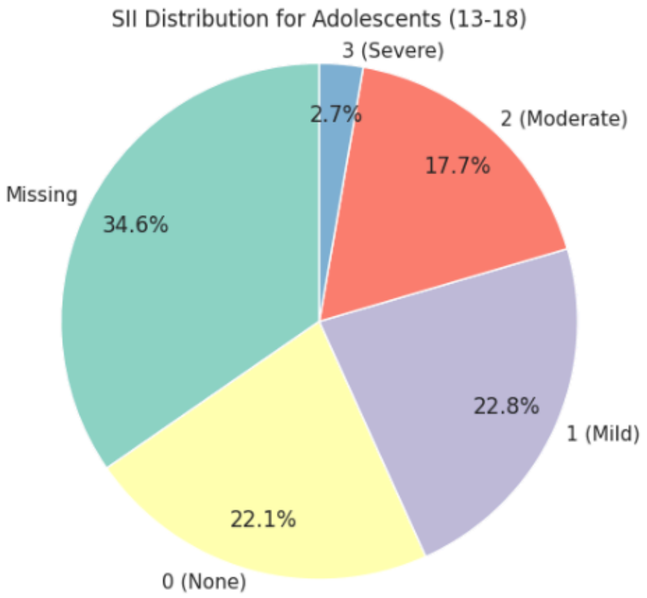


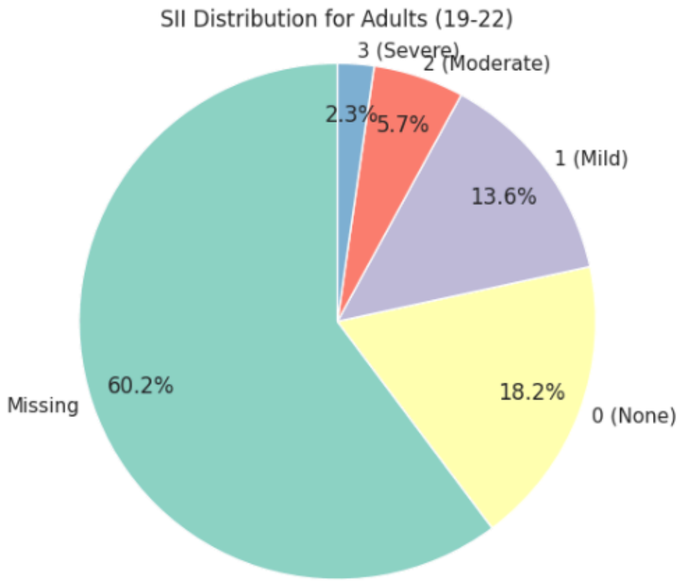
A significant proportion of the data is missing (31.3%, 1241 entries), the majority of individuals exhibit no impairment (40.1%), followed by mild (18.2%), moderate (9.5%), and severe impairments (0.9%).

TABULAR DATA

Distribution of SII scores by age groups

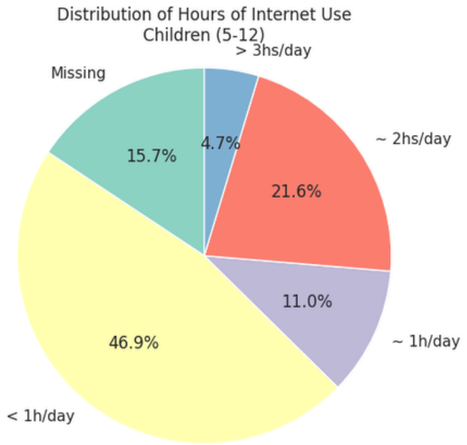


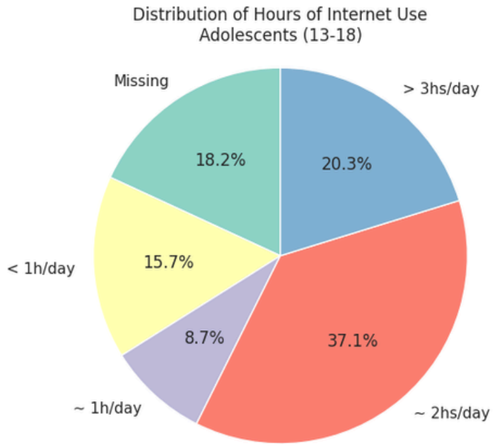


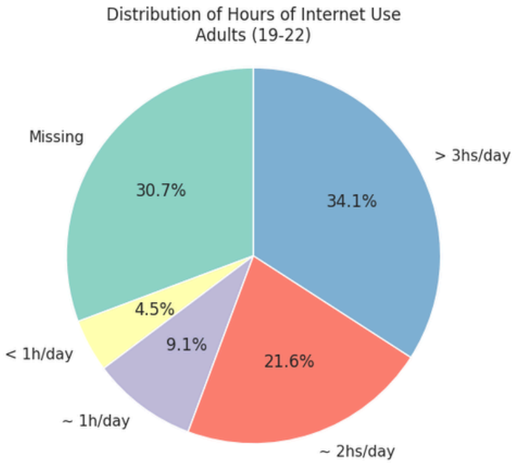


TABULAR DATA

Internet use







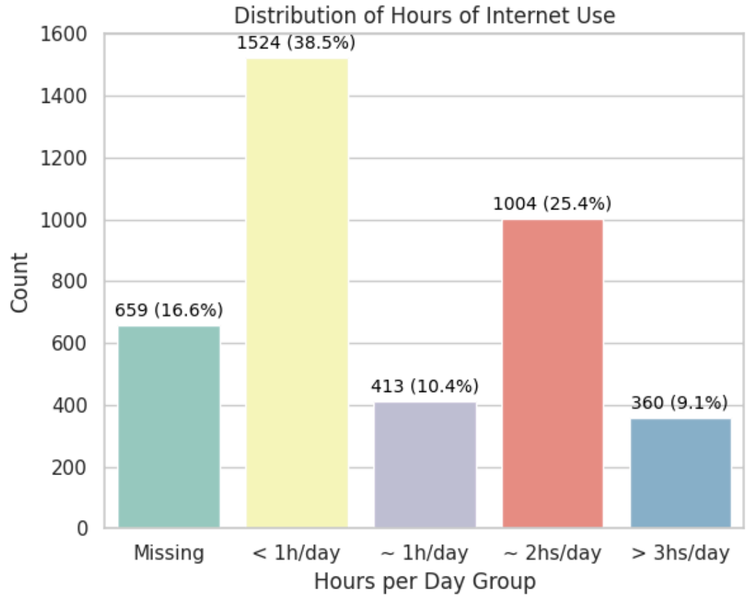
Internet use increases with age, with a growing share of individuals spending over 3 hours daily.

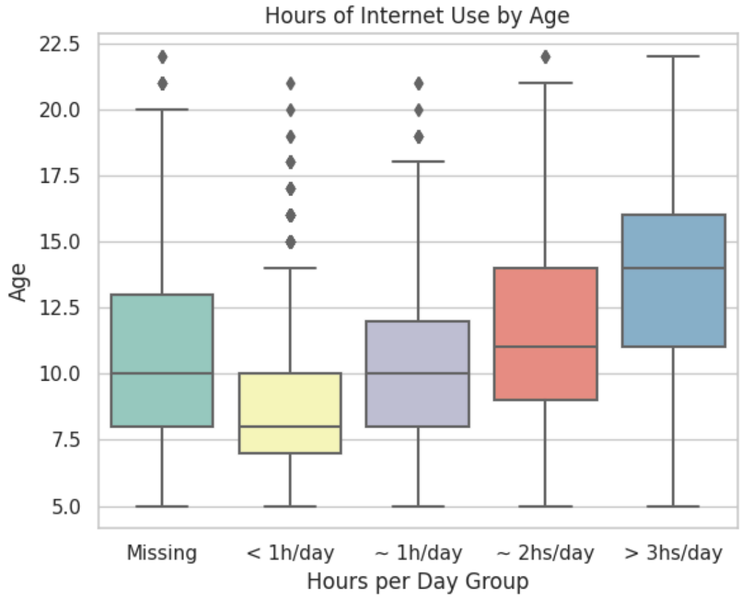
Missing data is substantial in all groups, especially among adults, which could impact the accuracy of the results.

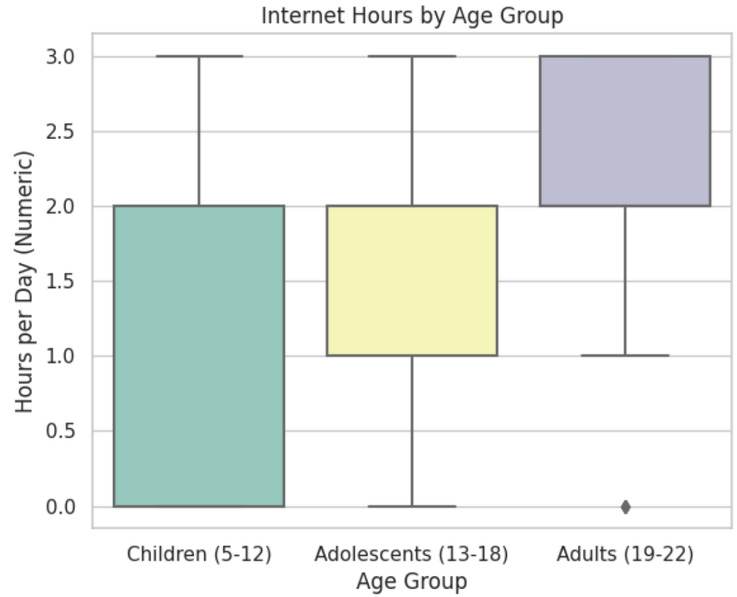
Younger age groups (children) are less likely to spend prolonged hours online compared to adolescents and adults.

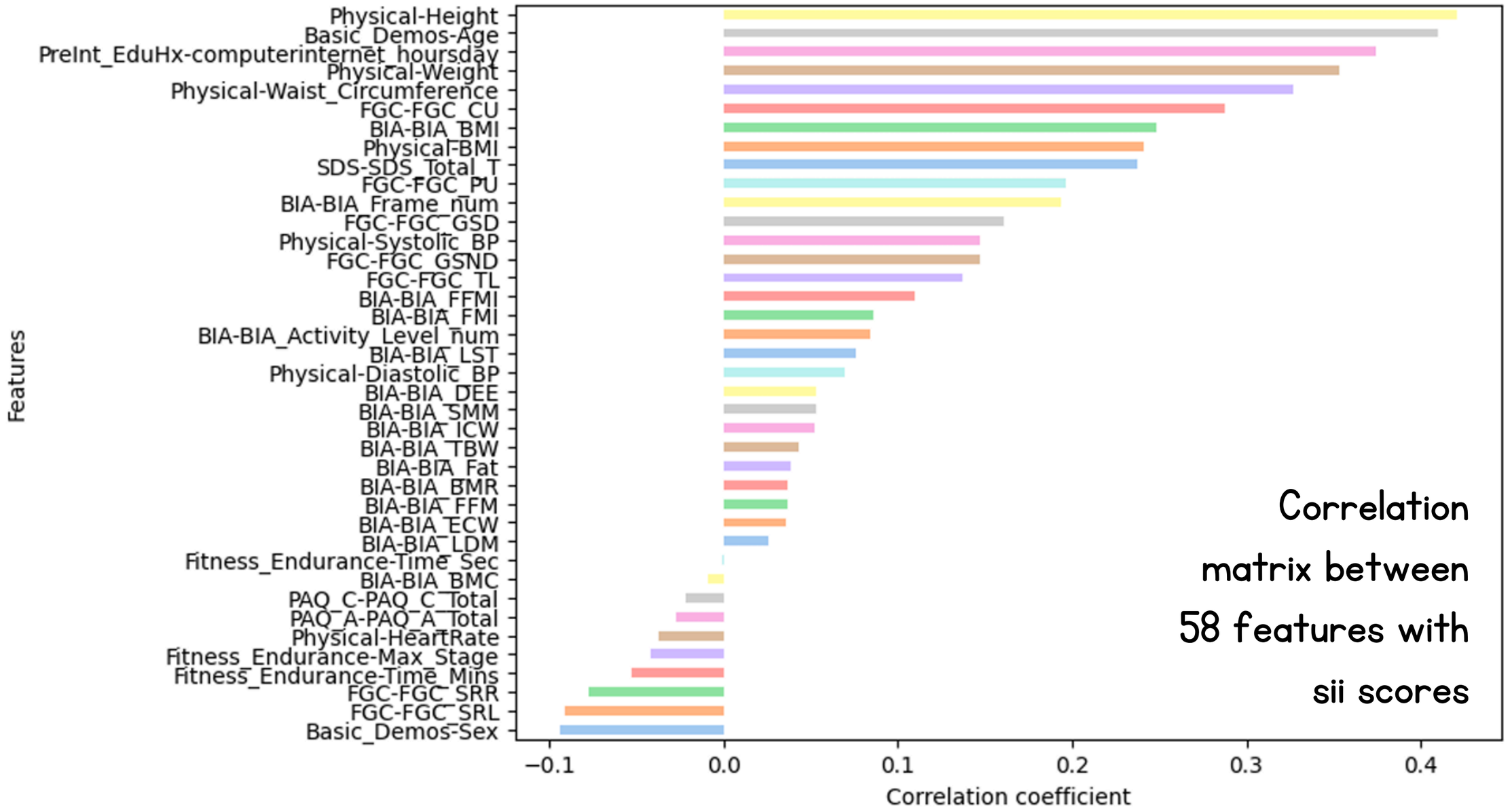
TABULAR DATA

Internet use









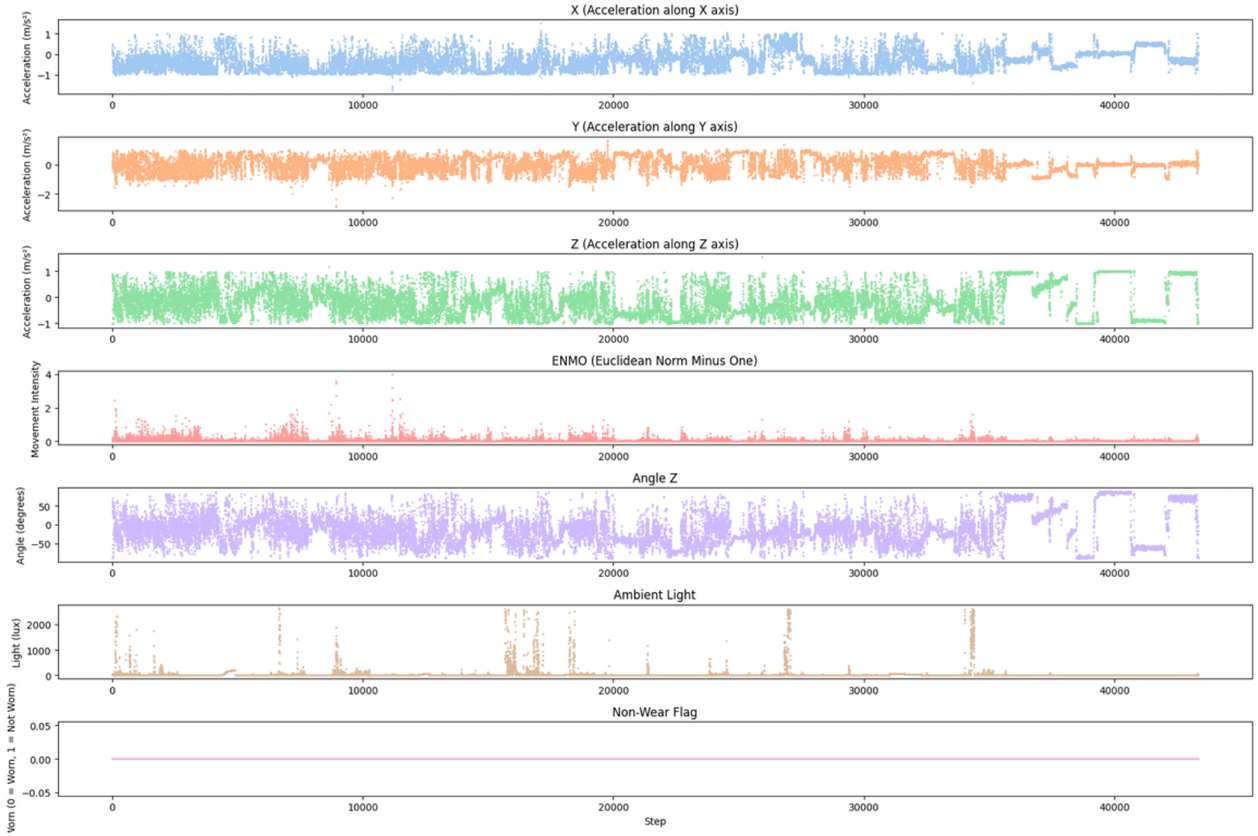
Correlation matrix between 58 features with sii scores

TIME-SERIES DATA

train dataset: 996 samples, 97 features

test dataset: 2 samples, 97 features

The time-series data provides detailed actigraphy metrics such as acceleration (X, Y, Z), ENMO, angle-Z, non-wear f lags, and contextual features like light levels, time of day, and weekday.



DATA PRE-PROCESSING

Data preprocessing is a key step to improve performance throughout the competition. Our team continuously refined the preprocessing pipeline across multiple versions and found the pipeline that performed best across different versions.

DATA PROCESS

Fill NaN values as "Missing"

- Select columns related to seasonal data
- Convert from String type to Category type
- Replace NaN elements with "Missing"
- Mapping column data to Int type

sii	PCIAT-PCIAT_Total min	PCIAT-PCIAT_Total max	count
i64	i64	i64	u32
null	null	null	1224
0	0	30	1594
1	31	49	730
2	50	79	378
3	80	93	34

DATA PROCESS

Rows with missing values in the sii and PCIAT columns were dropped

DATA PROCESS

Select important features

Select features that have high correlation with Sii and PCIAT, and remove the ID column

```
featuresCols = ['Basic_Demos-Enroll_Season', 'Basic_Demos-Age', 'Basic_Demos-Sex',
                'CGAS-Season', 'CGAS-CGAS_Score', 'Physical-Season', 'Physical-BMI',
                'Physical-Height', 'Physical-Weight', 'Physical-Waist_Circumference',
                'Physical-Diastolic_BP', 'Physical-HeartRate', 'Physical-Systolic_BP',
                'Fitness_Endurance-Season', 'Fitness_Endurance-Max_Stage',
                'Fitness_Endurance-Time_Mins', 'Fitness_Endurance-Time_Sec',
                'FGC-Season', 'FGC-FGC_CU', 'FGC-FGC_CU_Zone', 'FGC-FGC_GSND',
                'FGC-FGC_GSND_Zone', 'FGC-FGC_GSD', 'FGC-FGC_GSD_Zone', 'FGC-FGC_PU',
                'FGC-FGC_PU_Zone', 'FGC-FGC_SRL', 'FGC-FGC_SRL_Zone', 'FGC-FGC_SRR',
                'FGC-FGC_SRR_Zone', 'FGC-FGC_TL', 'FGC-FGC_TL_Zone', 'BIA-Season',
                'BIA-BIA_Activity_Level_num', 'BIA-BIA_BMC', 'BIA-BIA_BMI',
                'BIA-BIA_BMR', 'BIA-BIA_DEE', 'BIA-BIA_ECW', 'BIA-BIA_FFM',
                'BIA-BIA_FFMI', 'BIA-BIA_FMI', 'BIA-BIA_Fat', 'BIA-BIA_Frame_num',
                'BIA-BIA_ICW', 'BIA-BIA_LDM', 'BIA-BIA_LST', 'BIA-BIA_SMM',
                'BIA-BIA_TBW', 'PAQ_A-Season', 'PAQ_A-PAQ_A_Total', 'PAQ_C-Season',
                'PAQ_C-PAQ_C_Total', 'SDS-Season', 'SDS-SDS_Total_Raw',
                'SDS-SDS_Total_T', 'PreInt_EduHx-Season',
                'PreInt_EduHx-computerinternet_hoursday', 'sii']
```

DATA PROCESS

Feature Engineering

Calculating related metrics reduces the number of features and eliminates columns that store seasonal information.

```
def feature_engineering(df):
    season_cols = [col for col in df.columns if 'Season' in col]
    df = df.drop(season_cols, axis=1)
    df['BMI_Age'] = df['Physical-BMI'] * df['Basic_Demos-Age']
    df['Internet_Hours_Age'] = df['PreInt_EduHx-computerinternet_hoursday'] * df['Basic_Demos-
Age']
    df['BMI_Internet_Hours'] = df['Physical-BMI'] * df['PreInt_EduHx-computerinternet_hoursda
y']
    df['BFP_BMI'] = df['BIA-BIA_Fat'] / df['BIA-BIA_BMI']
    df['FFMI_BFP'] = df['BIA-BIA_FFMI'] / df['BIA-BIA_Fat']
    df['FMI_BFP'] = df['BIA-BIA_FMI'] / df['BIA-BIA_Fat']
    df['LST_TBW'] = df['BIA-BIA_LST'] / df['BIA-BIA_TBW']
    df['BFP_BMR'] = df['BIA-BIA_Fat'] * df['BIA-BIA_BMR']
    df['BFP_DEE'] = df['BIA-BIA_Fat'] * df['BIA-BIA_DEE']
    df['BMR_Weight'] = df['BIA-BIA_BMR'] / df['Physical-Weight']
    df['DEE_Weight'] = df['BIA-BIA_DEE'] / df['Physical-Weight']
    df['SMM_Height'] = df['BIA-BIA_SMM'] / df['Physical-Height']
    df['Muscle_to_Fat'] = df['BIA-BIA_SMM'] / df['BIA-BIA_FMI']
    df['Hydration_Status'] = df['BIA-BIA_TBW'] / df['Physical-Weight']
    df['ICW_TBW'] = df['BIA-BIA_ICW'] / df['BIA-BIA_TBW']

    return df
```

DATA PROCESS

Time-series data processing

- Time-series data is normalized using StandardScaler
- The AutoEncoder model is used to reduce the dimensionality of data and reconstruct data from encoded vectors.
- Reducing the number of features from 97 to 60

DATA PROCESS

Imputed missing values

- Use SimpleImputer to handle missing values.
- Input parameter strategy='median': replace NaN values with the median of the column

Models

1. RandomForest
2. LGBMRegressor
3. XGBRegressor
4. CatBoostRegressor
5. GradientBoosting
6. VotingRegressor

Models

For our final model, we employed a stacking approach combining 5 high-performing models: CatBoost, LightGBM, XGBoost, RF, GradientBoost. We train our models in 5 folds of data, and then optimized sii decision rounding threshold

MODELS

LightGBM

- Builds trees sequentially, with each tree minimizing the errors of the previous ones using gradient descent.
- Leaf-wise Growth: LightGBM grows trees leaf-wise. splits the leaf with the maximum loss reduction, leading to deeper trees and better optimization.

MODELS

LightGBM

- Histogram-based Splitting: Instead of evaluating every possible split, LightGBM uses histograms of feature values, making the model much faster and less memory-intensive.
- Handles categorical features natively without the need for one-hot encoding.
- Optimized for large datasets and high-dimensional data.

MODELS

CatBoost

- CatBoost is a gradient-boosting algorithm specifically designed to handle categorical features effectively
- CatBoost processes categorical features without requiring preprocessing steps like one-hot encoding.
- CatBoost grows symmetric (balanced) trees, where splits are made in a predefined symmetric structure.

MODELS

XGBoost

- XGBoost builds an ensemble of decision trees sequentially, with each tree aiming to minimize the errors of the previous ones.
- Uses a technique called max_depth for pruning, stopping the tree growth early when no further gain is achieved.
- Natively supports datasets with missing values by learning the best direction for missing data during training.

MODELS

GradientBoost

- The GradientBoostingRegressor is a regression algorithm available in libraries like Scikit-learn that uses gradient boosting to predict continuous values.
- It combines the predictions of several weak learners (usually decision trees) to minimize a specified loss function.
- Particularly effective for datasets with complex relationships between features and the target variable.

MODELS

RandomForest

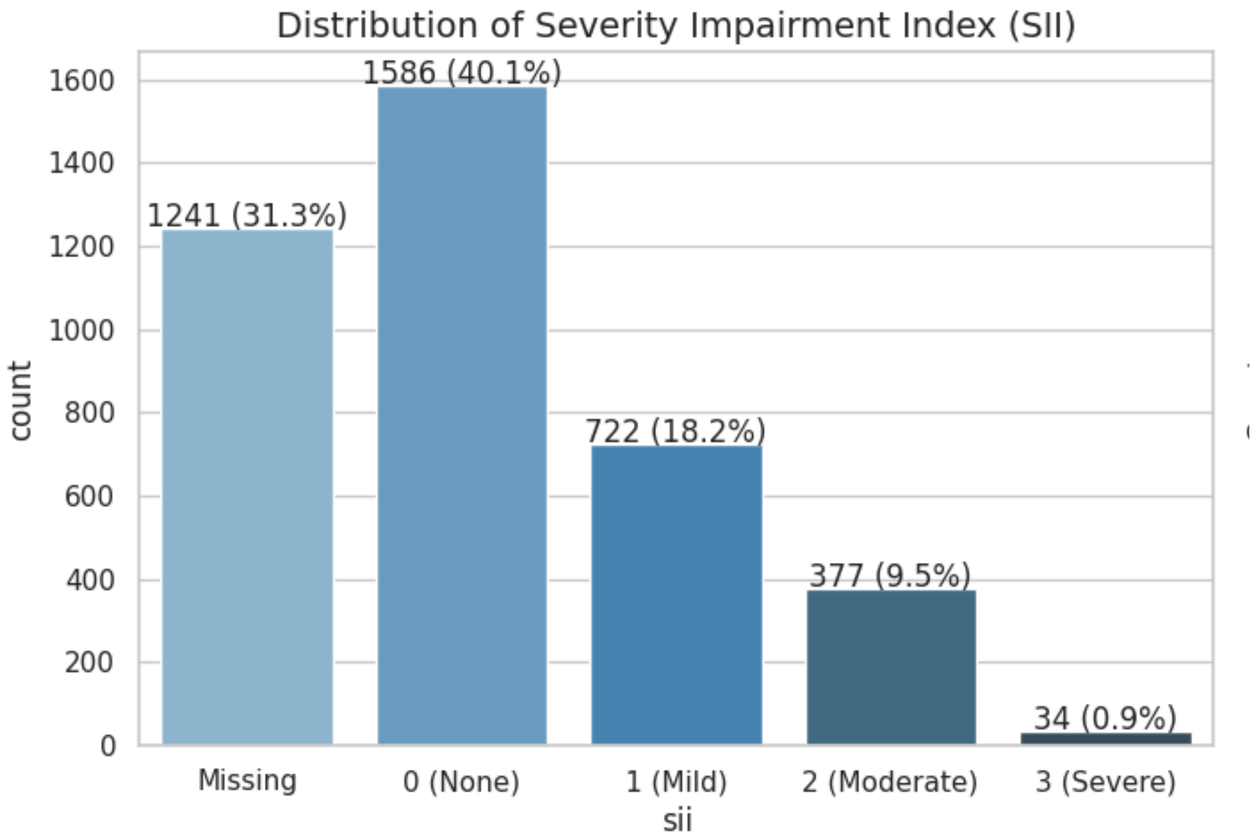
- Random Forest is an ensemble method that combines multiple models (decision trees) to improve performance .
- The model builds many decision trees, each trained on a random subset of the data.
- When creating a tree,random subset of features is considered for splitting at each node, reducing the correlation between trees
- For regression problems, the prediction is the average of the individual tree predictions.

MODELS

Voting Regressor

- The Voting Regressor is an ensemble learning technique that combines predictions from multiple regression models to improve accuracy and robustness.
- It works by aggregating the predictions of its constituent models (base regressors), either through averaging or weighted averaging, to produce the final prediction.

TRAINING FUNCTION



StratifiedKFold

This dataset is imbalanced. Half of the samples are in class 0, while very few in class 3.

=> Use StratifiedKFold to split the train data into 2 parts: train and validate

TRAINING

- Set the number of K-Fold iterations = 5
  - Split the training file into 2 parts train and val using StratifiedKFold
  - Fit model
  - Predict x_train and x_val
  - Calculating Performance Metrics using the Kappa formula
  - Predict x_test
  - Save the results to the array
- Optimizes thresholds to maximize QWK.
- Applying Optimized Thresholds to rounder test_preds
- Test Predictions with Tuned Thresholds

TRAINING

Set parameters for LGBM, XGBoost, CatBoost :

```
Params = {
    'learning_rate': 0.046,
    'max_depth': 12,
    'num_leaves': 478,
    'min_data_in_leaf': 13,
    'feature_fraction': 0.893,
    'bagging_fraction': 0.784,
    'bagging_freq': 4,
    'lambda_l1': 10,
    'lambda_l2': 0.01
}
```

```
XGB_Params = {
    'learning_rate': 0.05,
    'max_depth': 6,
    'n_estimators': 200,
    'subsample': 0.8,
    'colsample_bytree': 0.8,
    'reg_alpha': 1,
    'reg_lambda': 5,
    'random_state': SEED
}
```

```
CatBoost_Params = {
    'learning_rate': 0.05,
    'depth': 6,
    'iterations': 200,
    'random_seed': SEED,
    'cat_features': cat_c,
    'verbose': 0,
    'l2_leaf_reg': 10
}
```

TRAINING

VotingRegressor with the following parameters:

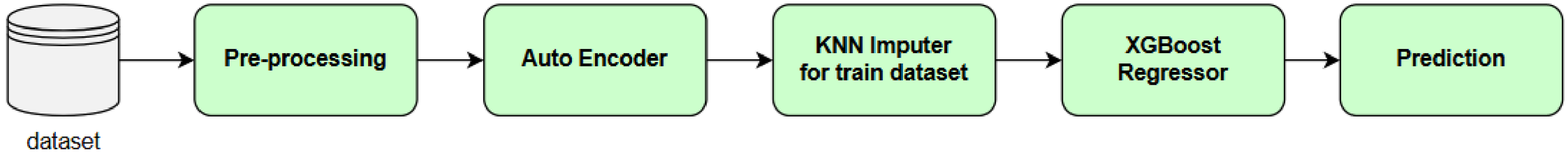
```
voting_model = VotingRegressor(estimators=[
    ('lightgbm', Light),
    ('xgboost', XGB_Model),
    ('catboost', CatBoost_Model),
    ('rf', Pipeline(steps=[('imputer', imputer), ('regressor', RandomForestRegressor(random_st
ate=SEED))])),
    ('gb', Pipeline(steps=[('imputer', imputer), ('regressor', GradientBoostingRegressor(rando
m_state=SEED))]))
])
```

TRAINING

Run training function and save results to csv file

```
Submission3 = TrainML(sample, voting_model, train = train_3, test_data = test_3)
Submission3.to_csv('submission.csv', index=False)
Submission3
```

Model Improvement



Mean Train QWK --> 1.0000

Mean Validation QWK ---> 0.458

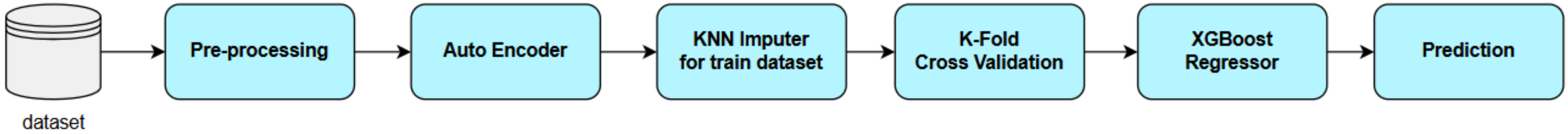
----> ll Optimized QWK SCORE :: 0.458

RESULT

Public score: 0.099

Private score: 0.075

version 1



Mean Train QWK --> 1.0000

Mean Validation QWK ---> 0.4605

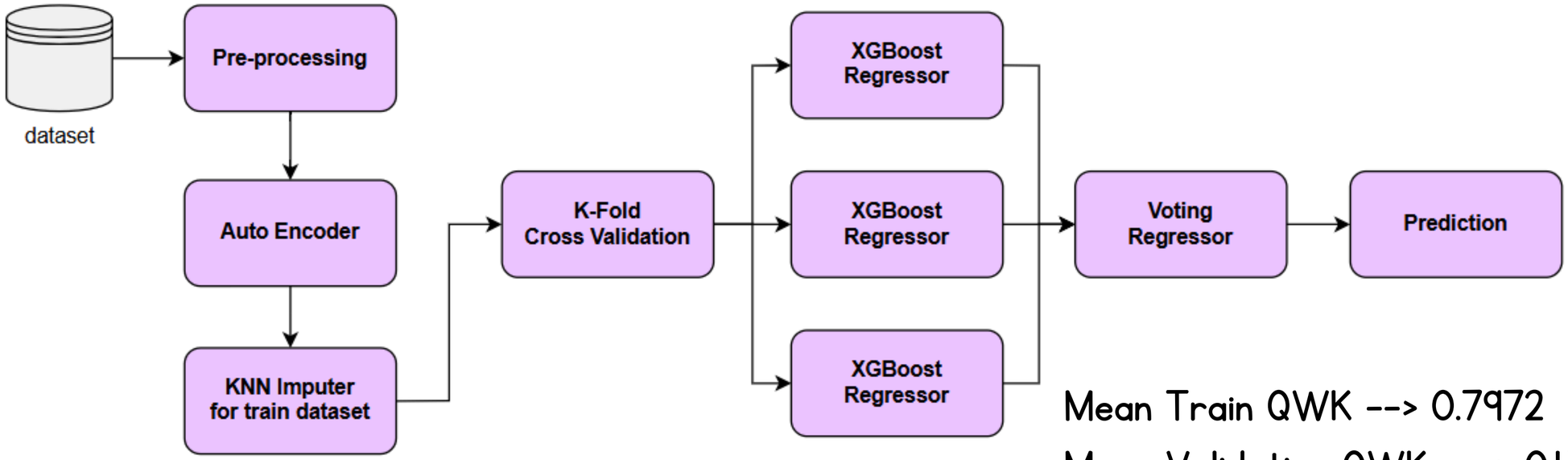
----> II Optimized QWK SCORE :: 0.460

RESULT

Public score: 0.227

Private score: 0.077

version 4



Mean Train QWK --> 0.7972

Mean Validation QWK ---> 0.4940

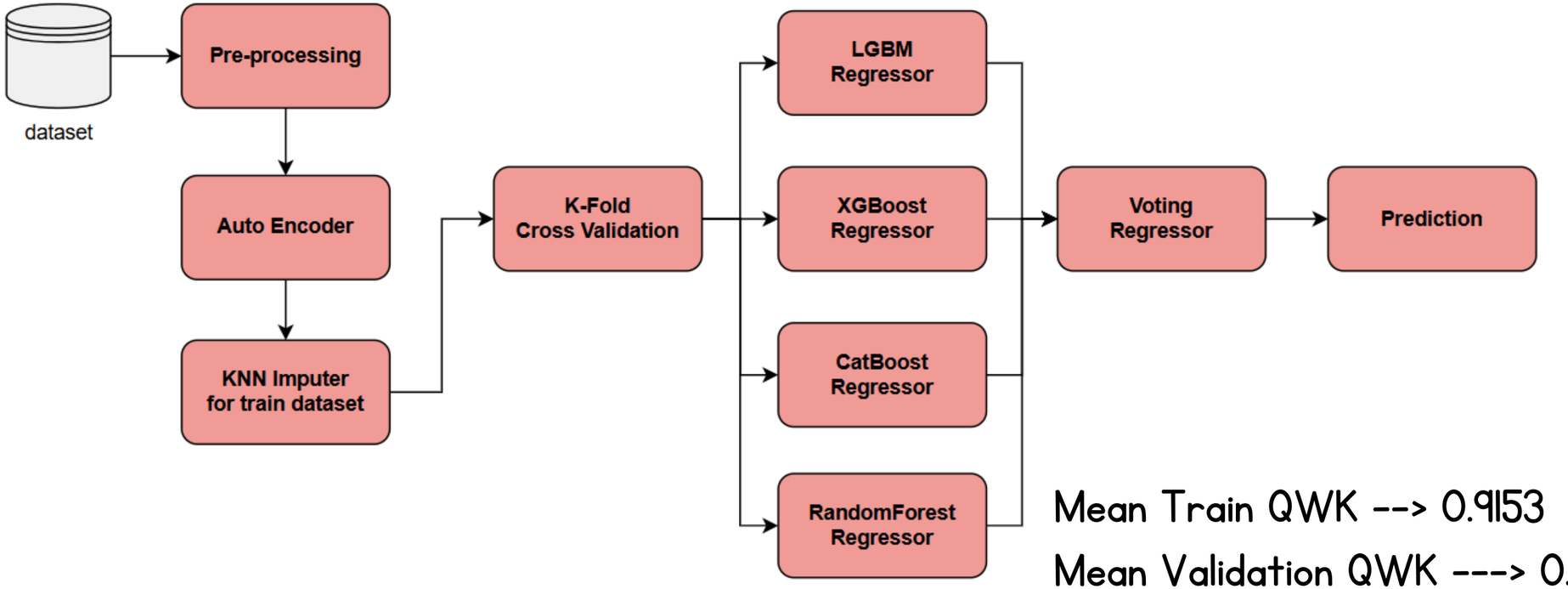
----> ll Optimized QWK SCORE :: 0.541

RESULT

Public score: 0.315

Private score: 0.145

version 10



Mean Train QWK --> 0.9153

Mean Validation QWK ---> 0.5033

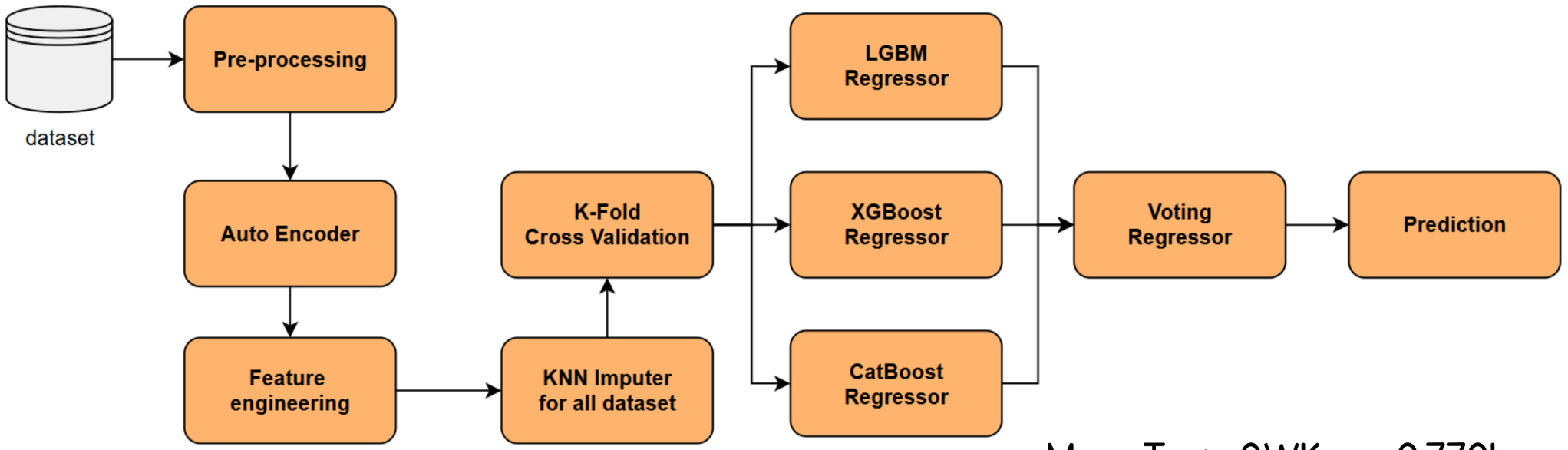
----> ll Optimized QWK SCORE :: 0.543

RESULT

Public score: 0.358

Private score: 0.205

version 12



Mean Train QWK --> 0.7701

Mean Validation QWK ---> 0.4857

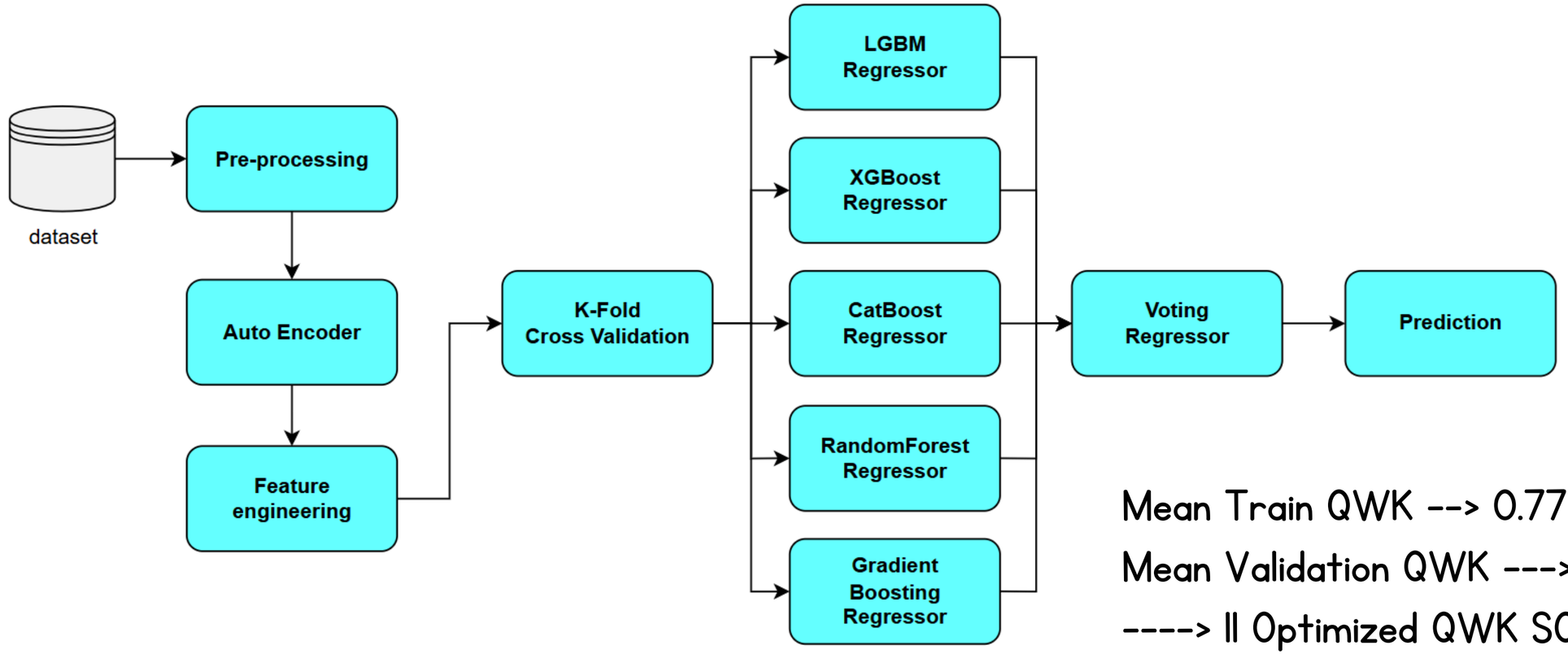
----> ll Optimized QWK SCORE :: 0.540

RESULT

Public score: 0.443

Private score: 0.407

version 15



Mean Train QWK --> 0.7775

Mean Validation QWK ---> 0.3805

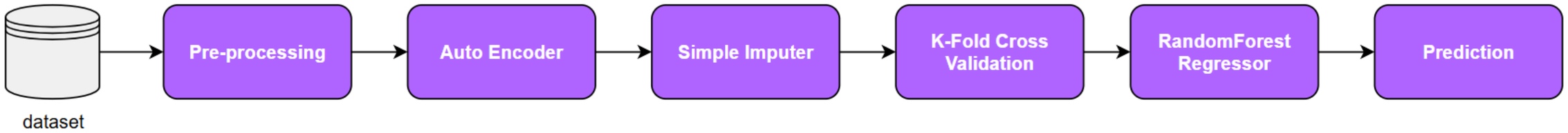
----> ll Optimized QWK SCORE :: 0.456

RESULT

Public score: 0.465

Private score: 0.449

version 16



Mean Train QWK --> 0.7585

Mean Validation QWK ---> 0.3594

----> ll Optimized QWK SCORE :: 0.430

RESULT

Public score: 0.446

Private score: 0.465

version 25

THANK YOU