Machine Learning

Problematic Internet Use
Childmind Institute

Lecturer : Dr. Le Duc Trong

Group 13

- **Trần Thái Dương - 22028061**
- **Nguyễn Mạnh Quân - 22028171**
- **Trần Huy Hoàng - 22028183**

Contents

1.Introduction

Child Mind Institute

Problem: Excessive internet use among children is linked to mental health issues like depression and anxiety, but current assessment methods are complex and inaccessible.

Proposal: Use physical fitness indicators (e.G., Posture, diet, activity levels) as proxies to predict problematic internet use.

1.Introduction

Goal: develop a predictive model to detect early signs of unhealthy digital habits, enabling timely interventions for healthier online behavior.

Impact: promote a balanced and responsible digital lifestyle for children.

1.Introduction

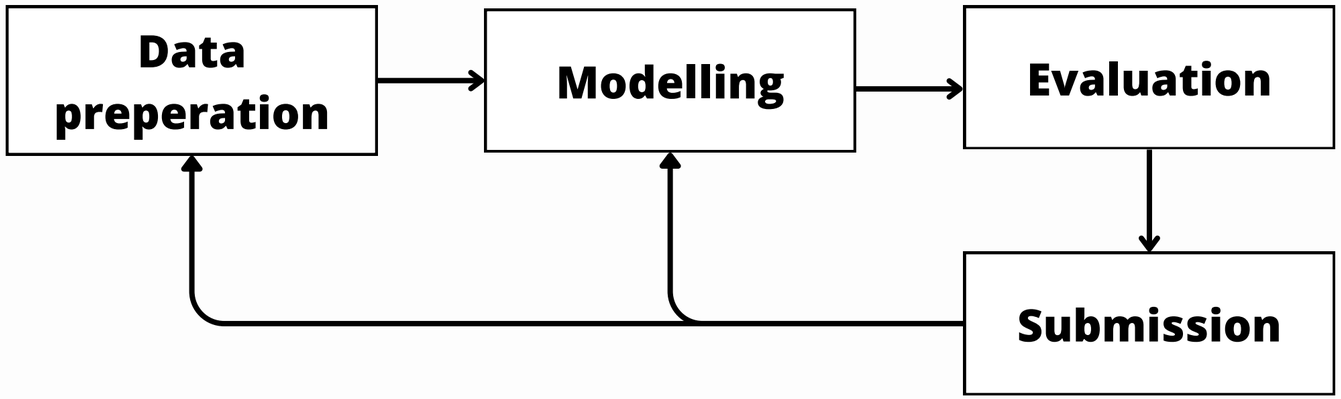
Out Target

Our goal is to predict the Severity Impairment Index (SII), a standardized measure of problematic internet use. The SII represents the severity level of internet-related issues for each individual.

Severity	SII
None	0
Mild	1
Moderate	2
Severe	3

2. General approach

The workflow

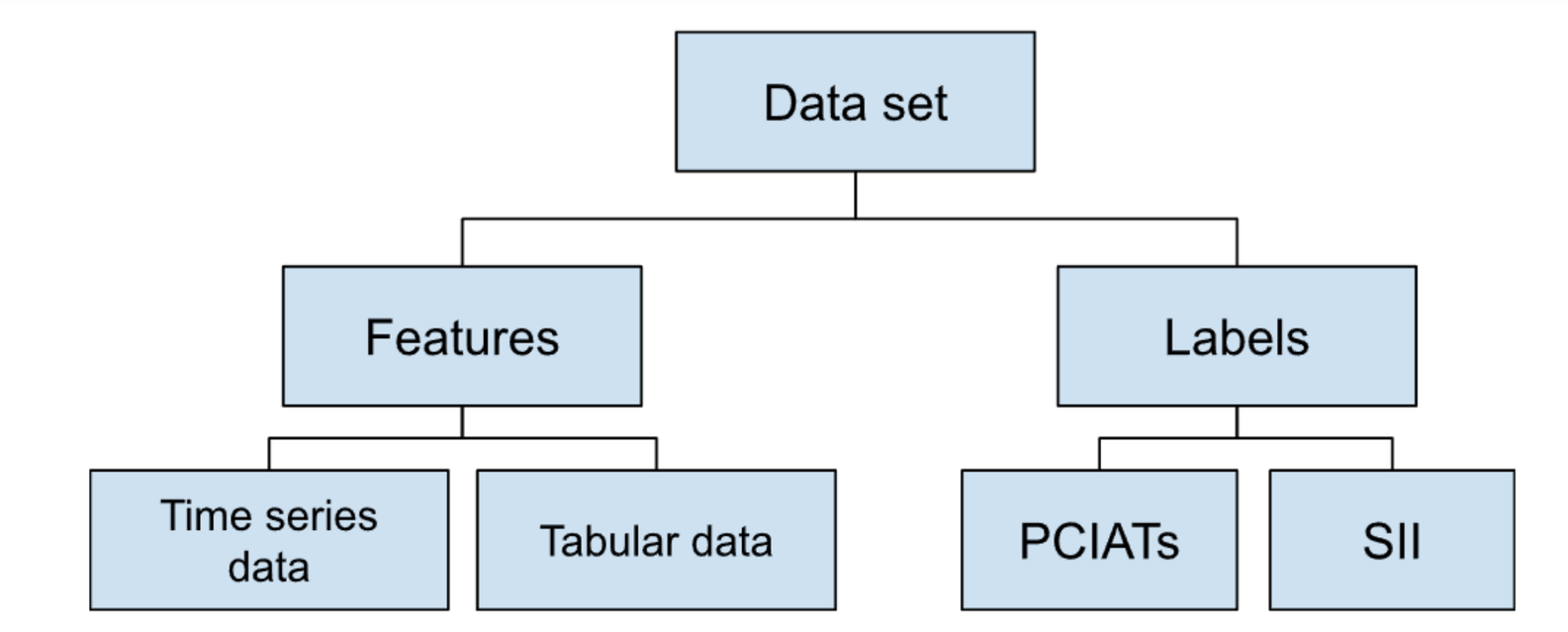


2. General approach

The pipeline

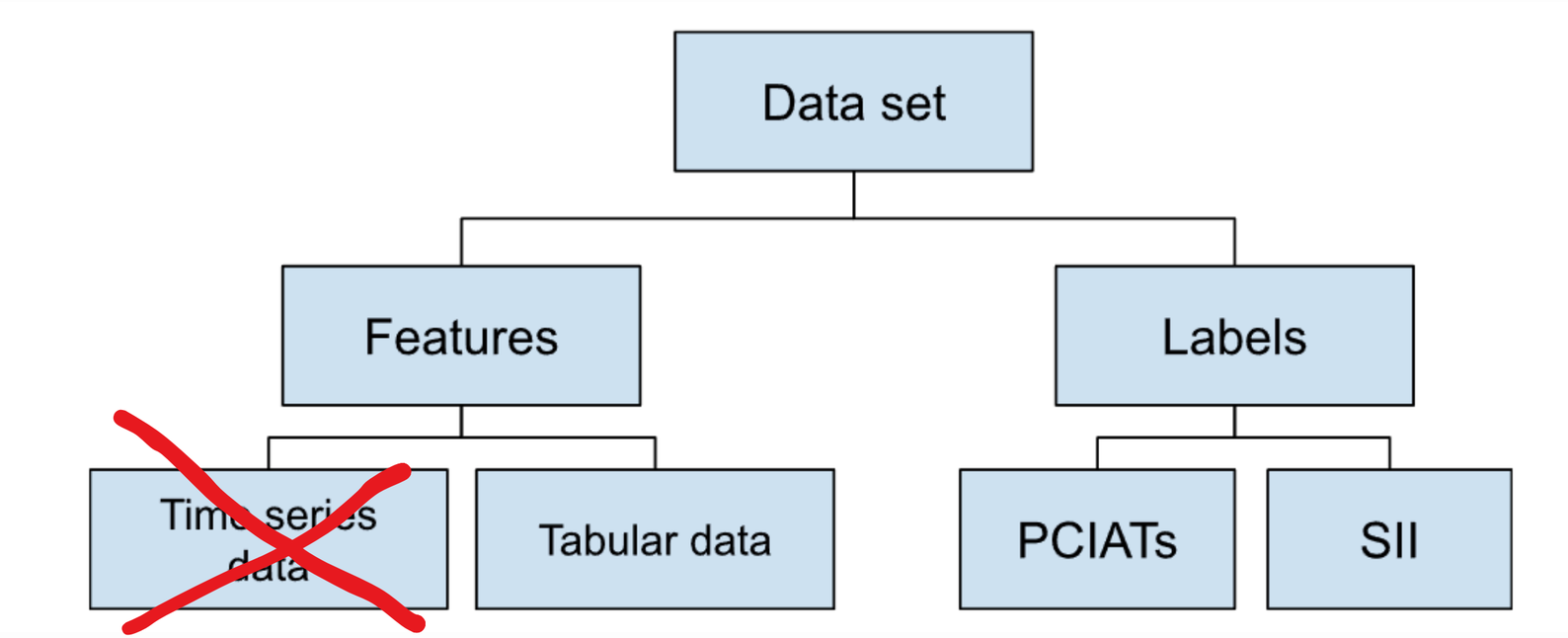
3. Data Overview

3.Data Overview



Classifying the given data based on their purpose and type

3.Data Overview

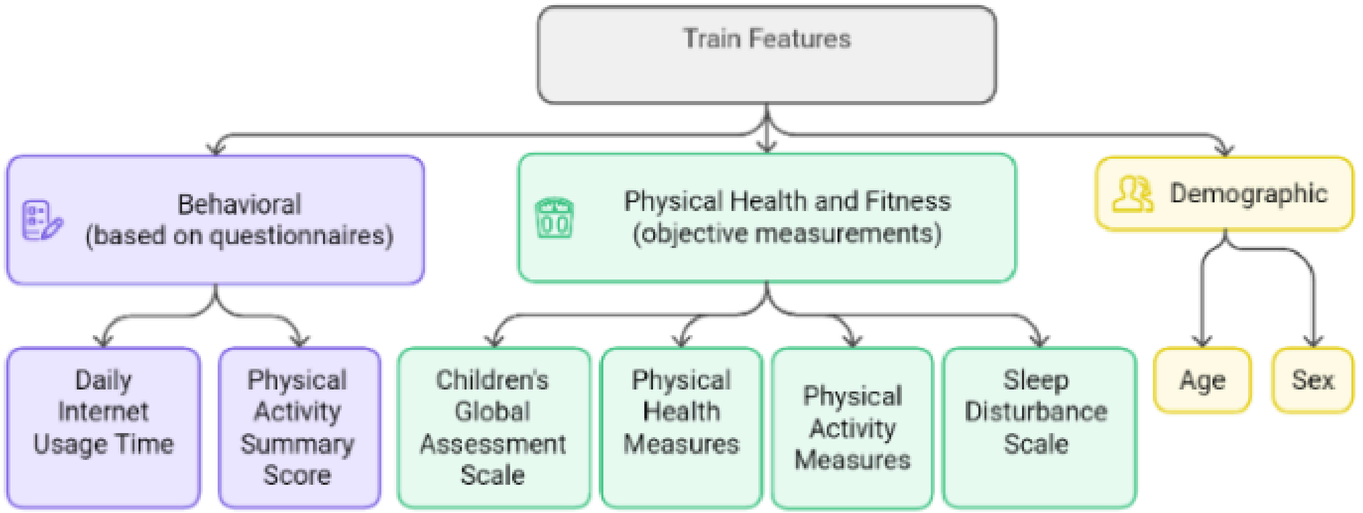


Time series data is not used in every version of our models

3.Data Overview

Features

- 60 Features
- The included data types:
  - categorical int
  - int
  - string
  - float



Grouping the features by type and measurement method

3.Data Overview

Labels

Severity	SII	PCIAT range
None	0	0-30
Mild	1	31-49
Moderate	2	50-79
Severe	3	80-100

Parent child addiction test - Question 1:

How often does your child disobey time limits you set for online use?

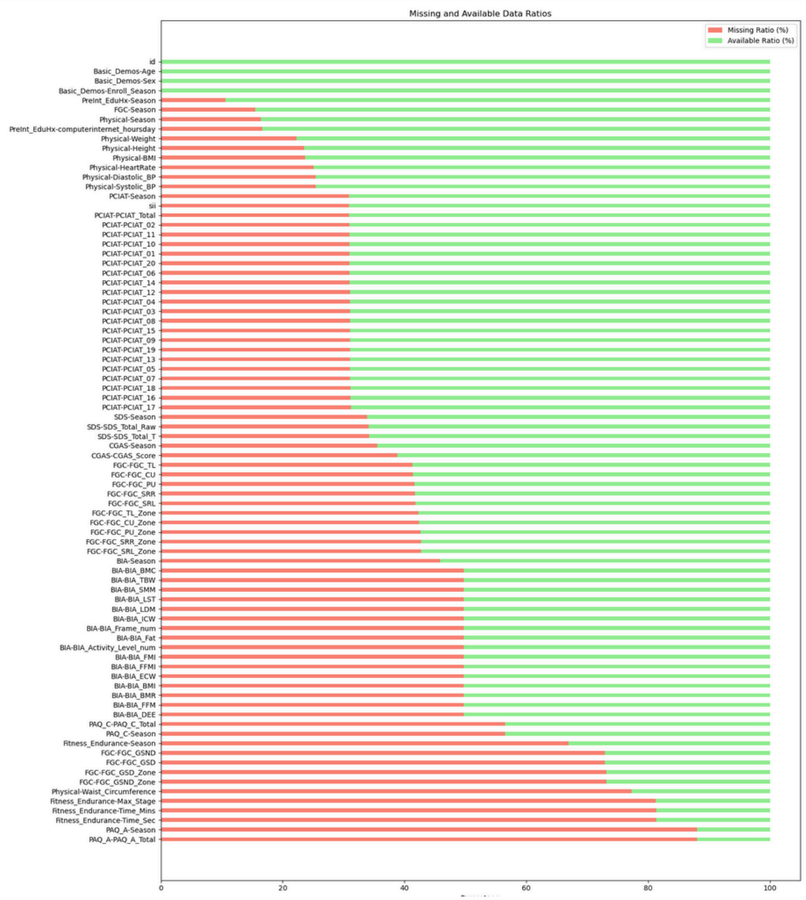
0=Does Not Apply, 1=Rarely, 2=Occasionally, 3=Frequently, 4=Often, 5=Always

4. Data processing

- We will show some data processing methods that was used in all of our models.
- The method applied in each models may have different variations.
- But generally, they holds the same idea.

4. Data processing

Handling missing labels



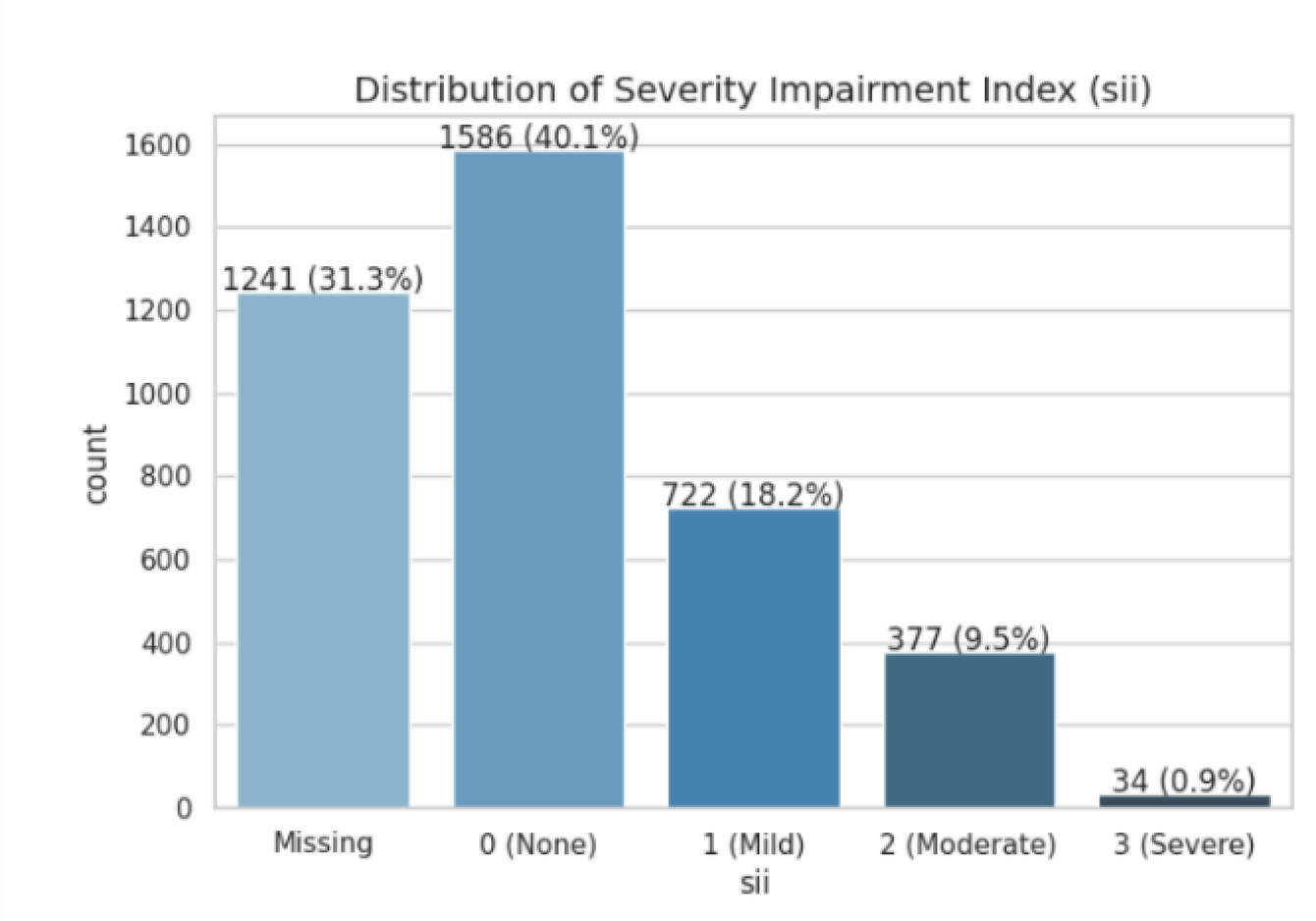
The missing/available data ratio graph

4. Data processing

Handling missing labels

4.Data processing

SII distribution

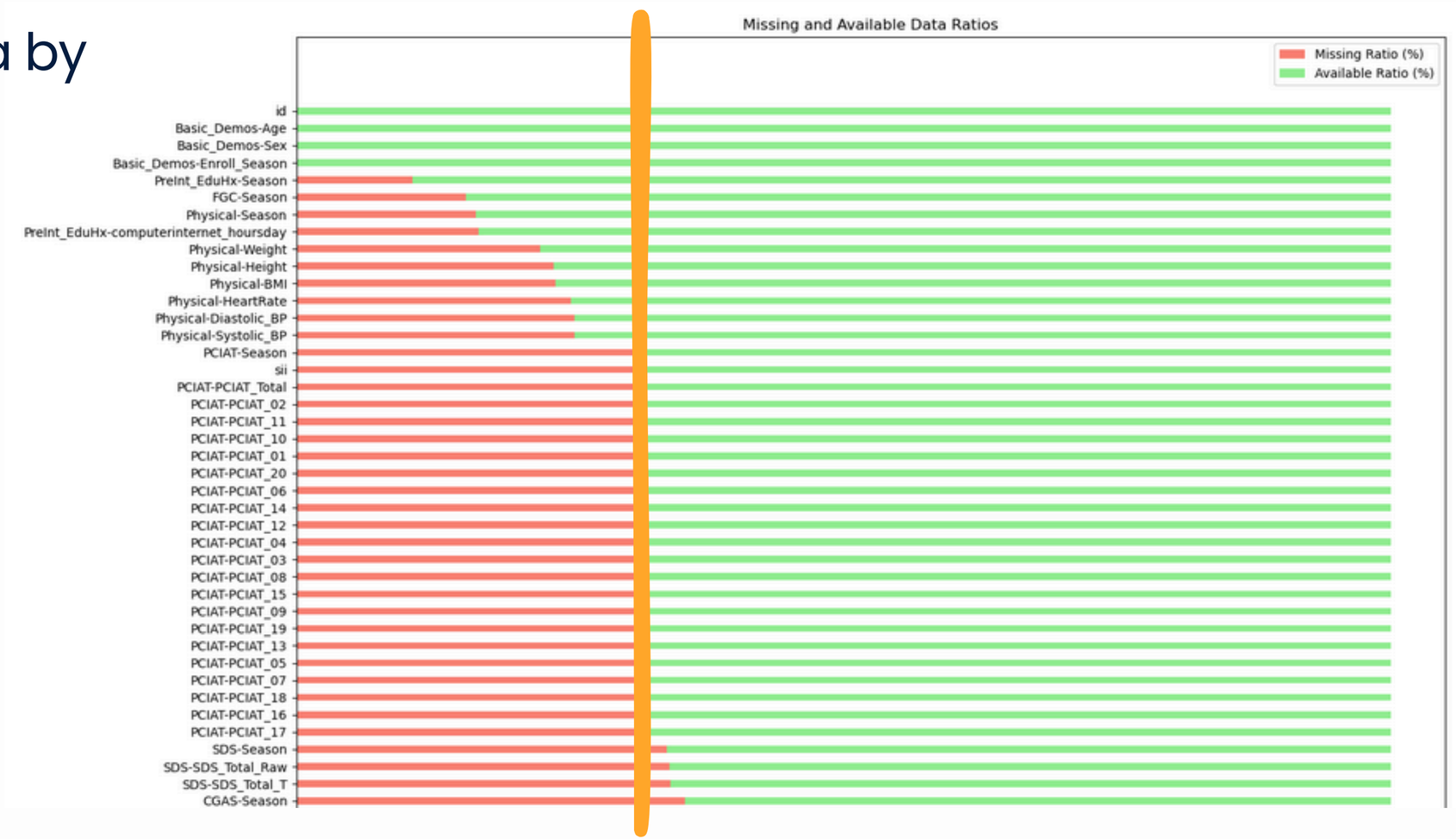


4. Data processing

Handling missing labels

Trim the data by

SII and PCIAT



4. Data processing

Handling missing labels

Before trimming:

3960 rows

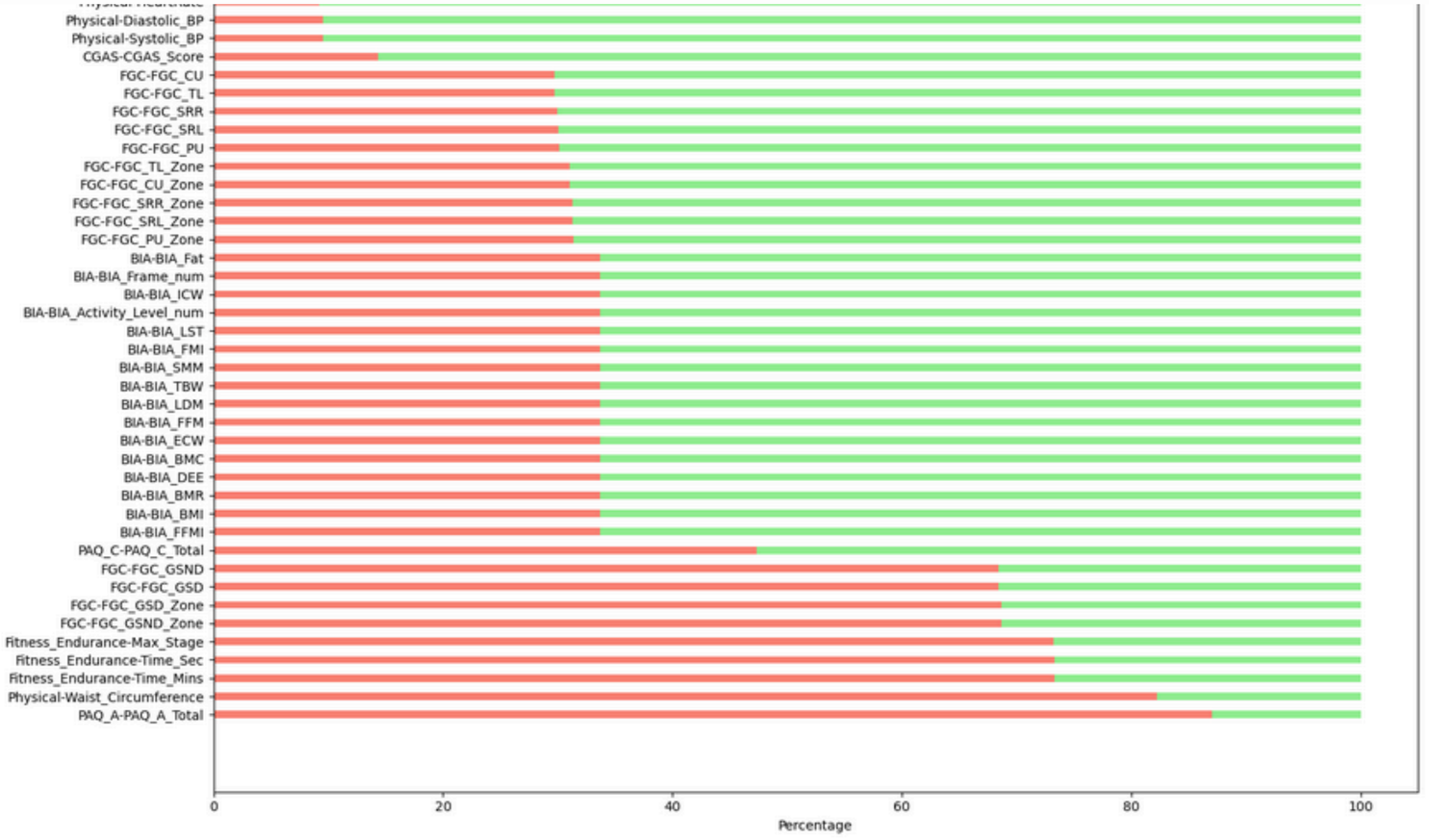
After trimming:

2671 rows

4. Data processing

Handling missing feature data

All features with high missing data ratio are dropped



Imputation leads to high bias.

All features with high missing ratio >= 40% of total data after trimming by PCIAT and SII.

4. Data processing

Handling string data

All string features, except for id, are seasonal data

['Basic_Demos-Enroll_Season', 'CGAS-Season', 'Physical-Season', 'Fitness_Endurance-Season', 'FGC-Season', 'BIA-Season', 'PAQ_A-Season', 'PAQ_C-Season', 'PCIAT-Season', 'SDS-Season', 'PreInt_EduHx-Season']

Either **ecode** the string features or **drop** them

4. Data processing

Handling noise

To simplify the data preprocessing, the extreme values of each feature will be capped at its 1st and 99th quantiles. Specifically:

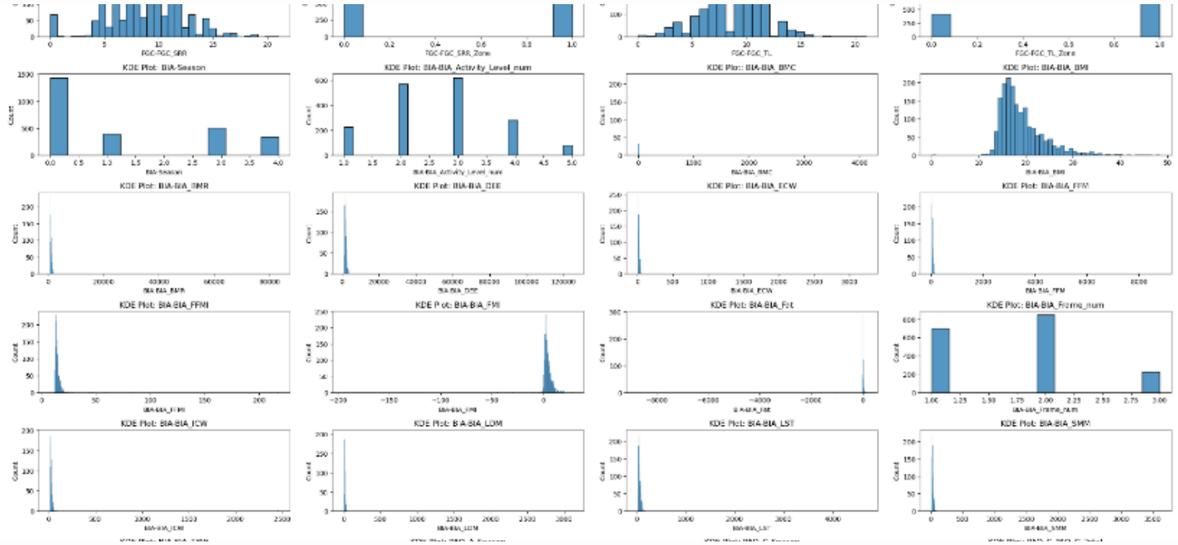
- Values below the 1st quantile will be set to the 1st quantile value.
- Values above the 99th quantile will be set to the 99th quantile value.

This ensures that all feature values during training, validation, and testing remain within the central 98% of the training dataset's distribution.

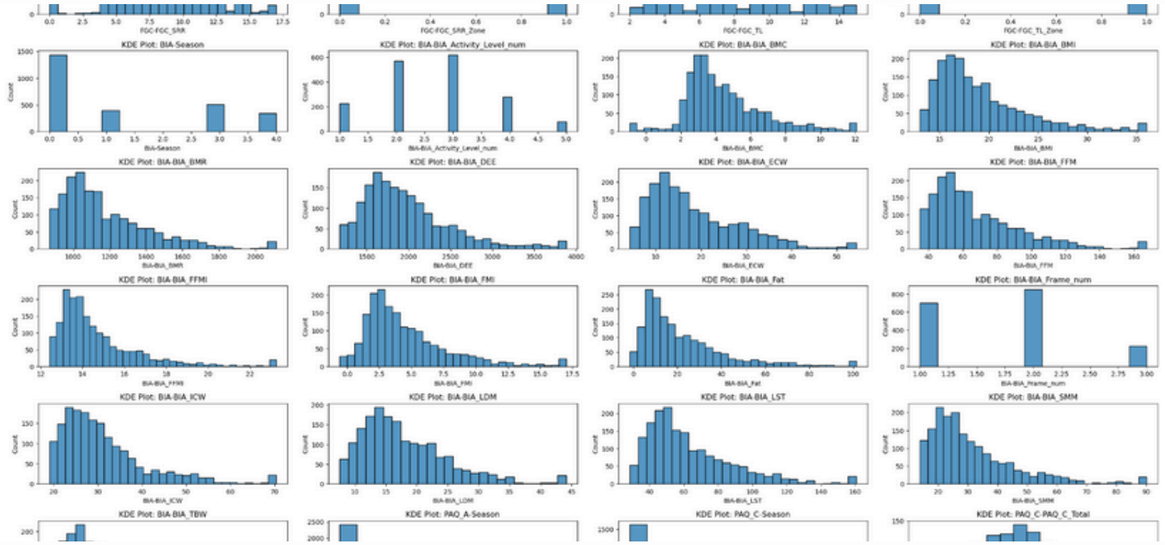
4. Data processing

Handling noise

Before capping

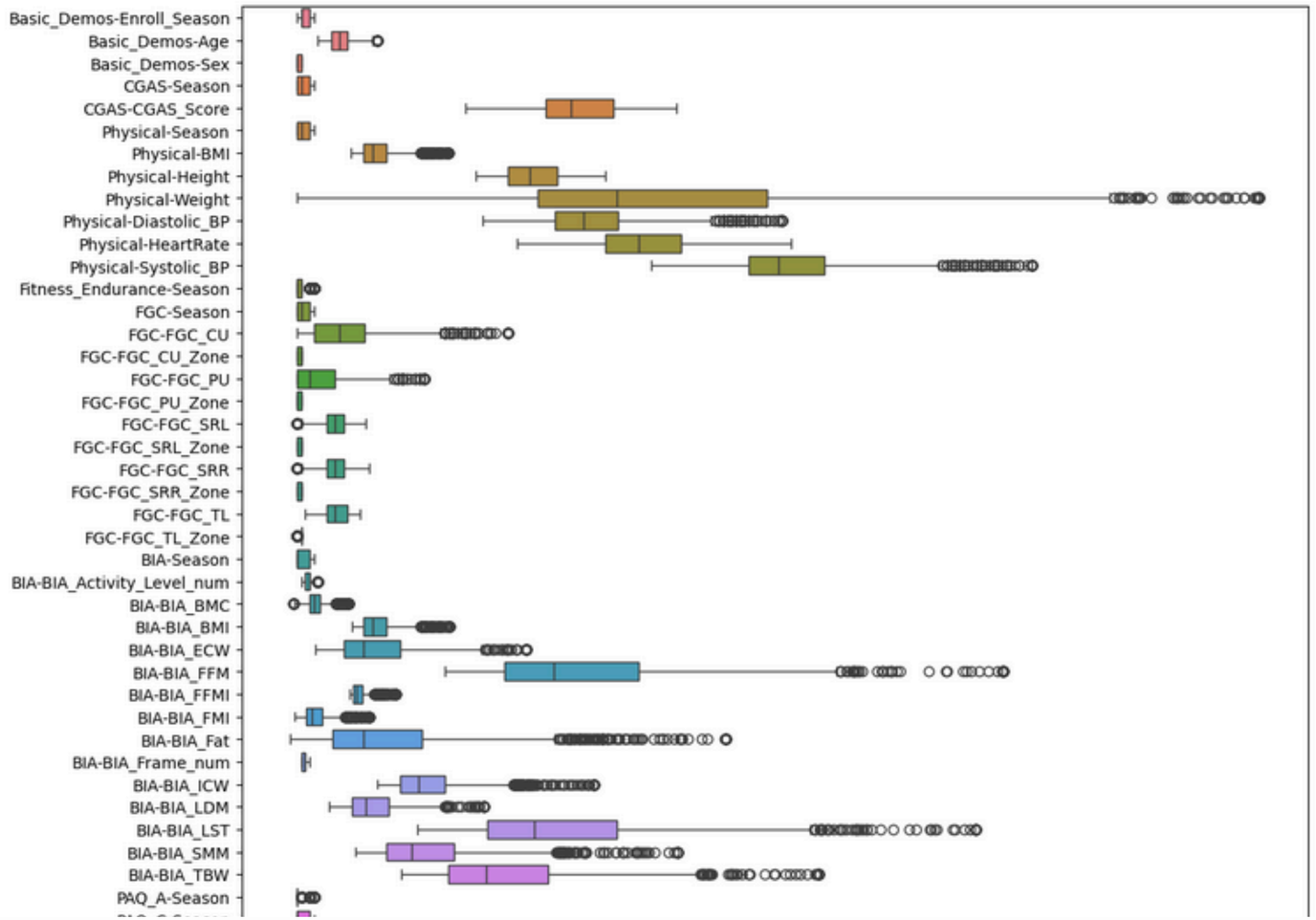


After capping



4. Data processing

Handling noise

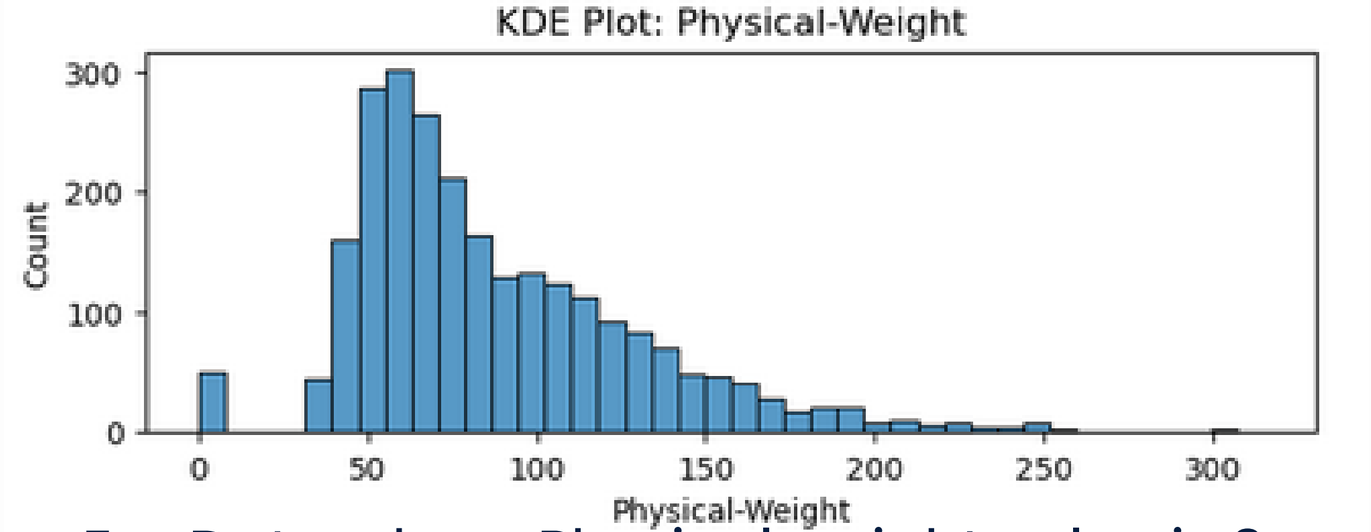


- Not all noise are removed
- Some noise may reflect real world special cases

4. Data processing

Handling noise

All invalid values are marked as NaN and will be recalculated using imputatation. E.g. Data where Physical weight value is 0



E.g. Data where Physical weight value is 0

5. Model selection

5. Model selection

Models taken into consideration

Criteria	SVM	Logistic Regression	Naive Bayes	Random forest	XGBoost
Computational Efficiency	High for non-linear kernels; moderate for linear	High	Very High	Moderate	High
Multi-class Support	Needs strategies like one-vs-one or one-vs-all	Native support	Native support	Native support	Native support
Performance on Complex Data	Excellent with non-linear kernels	May struggle without transformations	Poor with complex data	Excellent	Excellent
Preprocessing Needs	Requires careful scaling of features	Requires scaling for consistent results	Minimal	Minimal	Minimal
Robustness to Noise	Sensitive to outliers	Sensitive to outliers	Sensitive to feature independence assumptions	Robust	Robust
Suitability for Multi-Class (4 categories)	Feasible but computationally expensive	Simple and efficient	Feasible but assumes feature independence	Direct and efficient	Direct and efficient
Suitability for Small Dataset	Excellent with non-linear kernels (small training sets)	Works well with smaller datasets	Works well with small datasets	Works well with bootstrapping but can still overfit	Can work well, but may overfit on small datasets

6. Model training and validation

6. Model training and validation

Version 1.0

Predicting PCIAT values

Version 2.0

Predicting SII values

Version 3.0

Threshold Tuning

6. Model training and validation

Evaluation metric

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

Without the weights mistaking 1 with 2 is the same as mistaking 1 with 5

With the weights mistaking 1 with 2 is the less servere than mistaking 1 with 5

6.Model training and validation

Version 1.0 (0.237)

Predicts SII by using PCAIT

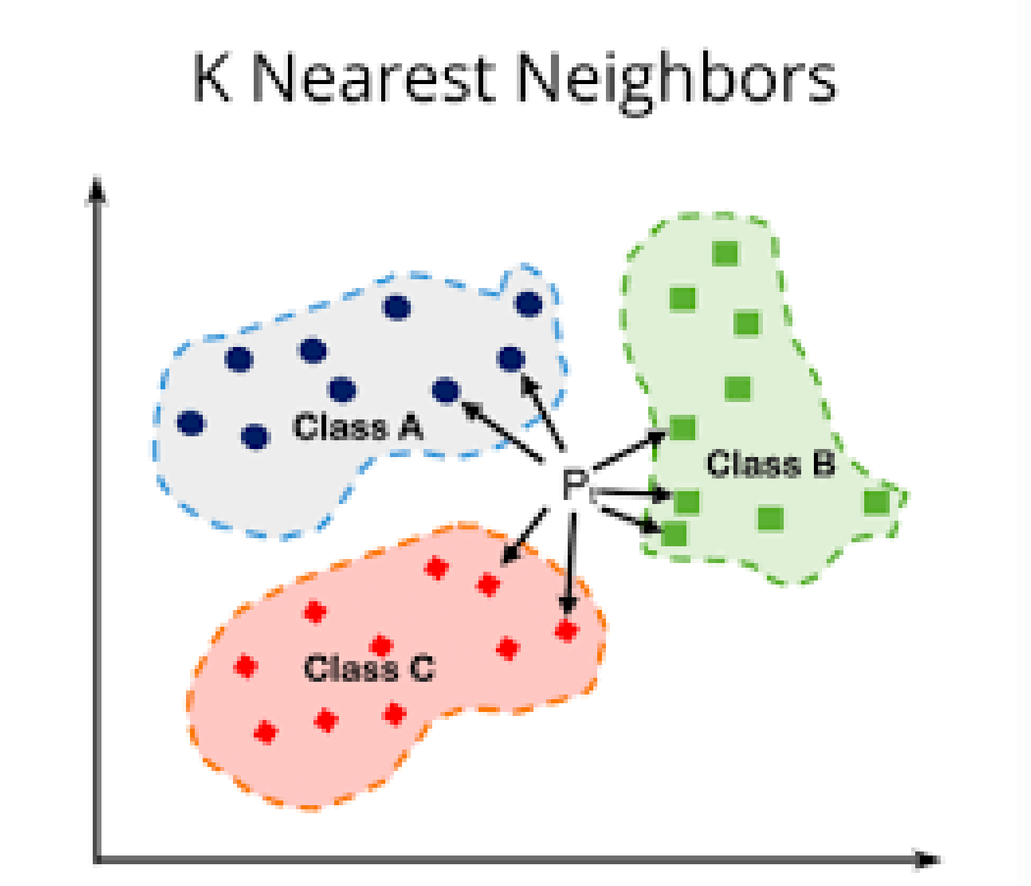
Using Random Forest classifier

No seasonal data used

6. Model training and validation

6.1. Version 1.0

Filling missing data using KNN imputer



6. Model training and validation

6.1. Version 1.0

We built a random forest model for each PCIAT question

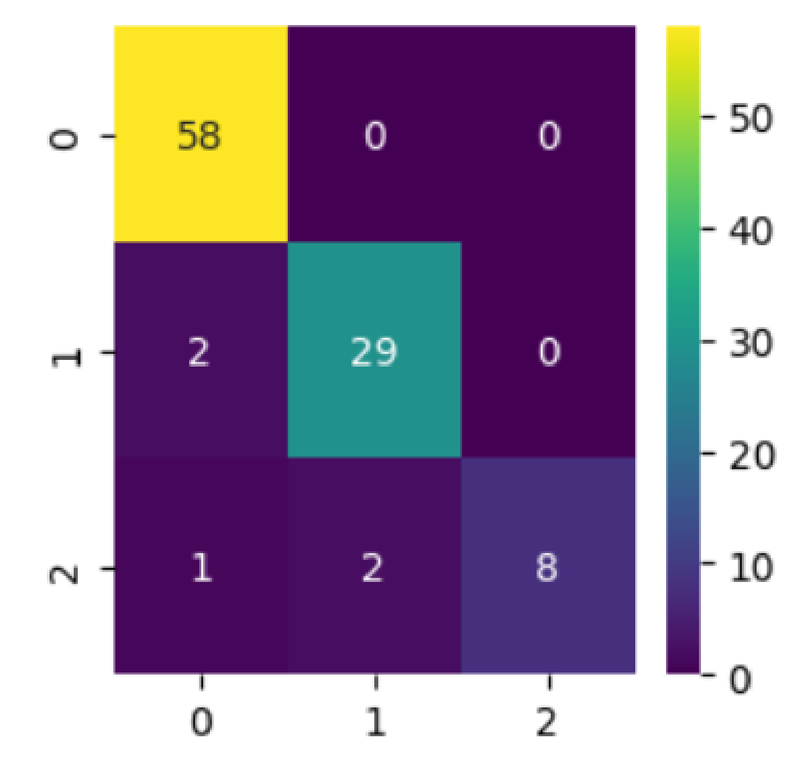
```
# Step 2: Build an ensemble tree for each PCIAT question
pciat_forest = {} # 20 forest for 20 questions, each forest contain 50 to 100 trees

for question in pciat_columns:
    if question == 'PCIAT-PCIAT_Total' or question == 'sii':
        continue

    forest = RandomForestClassifier(
        n_estimators=100,
        criterion='entropy',
        max_depth=10,
        random_state=0,
        bootstrap=True
    )

    forest.fit(np.array(X),np.array(np.array(Y[question])))
    pciat_forest[question] = forest
```

100 trees for 20 questions, which is 2000 trees!

6. Model training and validation

6.1. Version 1.0

Test set QWK 0.943



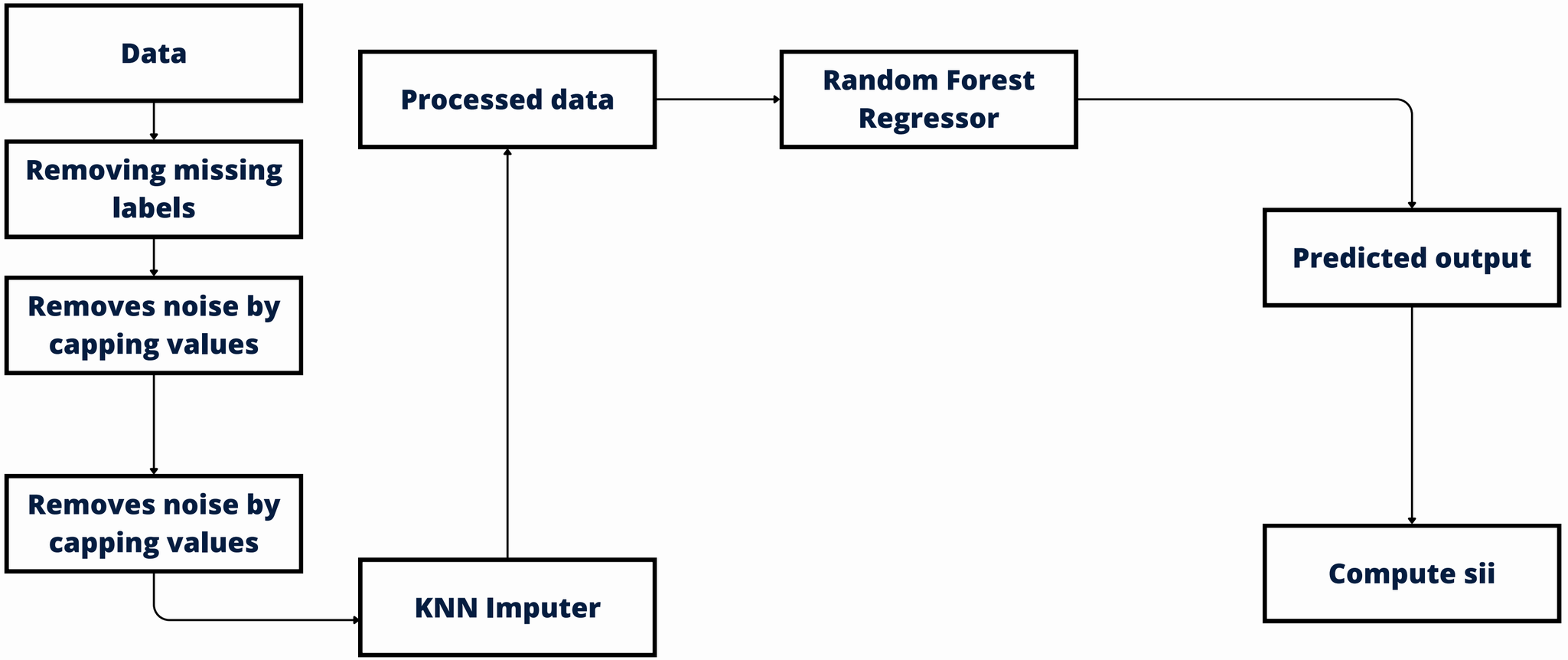
Cause of problem

```
test_quantity = 100
X = df[feature_columns]
Y = df[pciat_columns]

X_train = X[:-test_quantity]
X_test = X[-test_quantity:]
Y_train = Y[:-test_quantity]
Y_test = Y[-test_quantity:]
```

Test score: 0.237

Conclusion

Overview of our version 1.0



Data processing pipeline

6.Model training and validation

Version 2.0 (0.310)

Predict SII directly

Using Cross validated Grid search

Using QWK scrore

Using stratified sampling

6. Model training and validation

6.2. Version 2.0

Changes from version 1.0

- Instead of indirectly predicting SII values through PCIAT, we directly predict SII.
- No change was made in the data processing step.
- We tried better train/validation split strategy.
- Used Cross validatiion Grid search to optimize our parameters.

6. Model training and validation

6.2. Version 2.0

We used stratified sampling for better validation result accuracy

```
X = df[feature_columns]
Y = df['sii']

X_train, X_val, Y_train, Y_val = train_test_split(X, Y, train_size=0.8, test_size=0.2, stratify=Y, random_state=42)
```

6. Model training and validation

6.2. Version 2.0

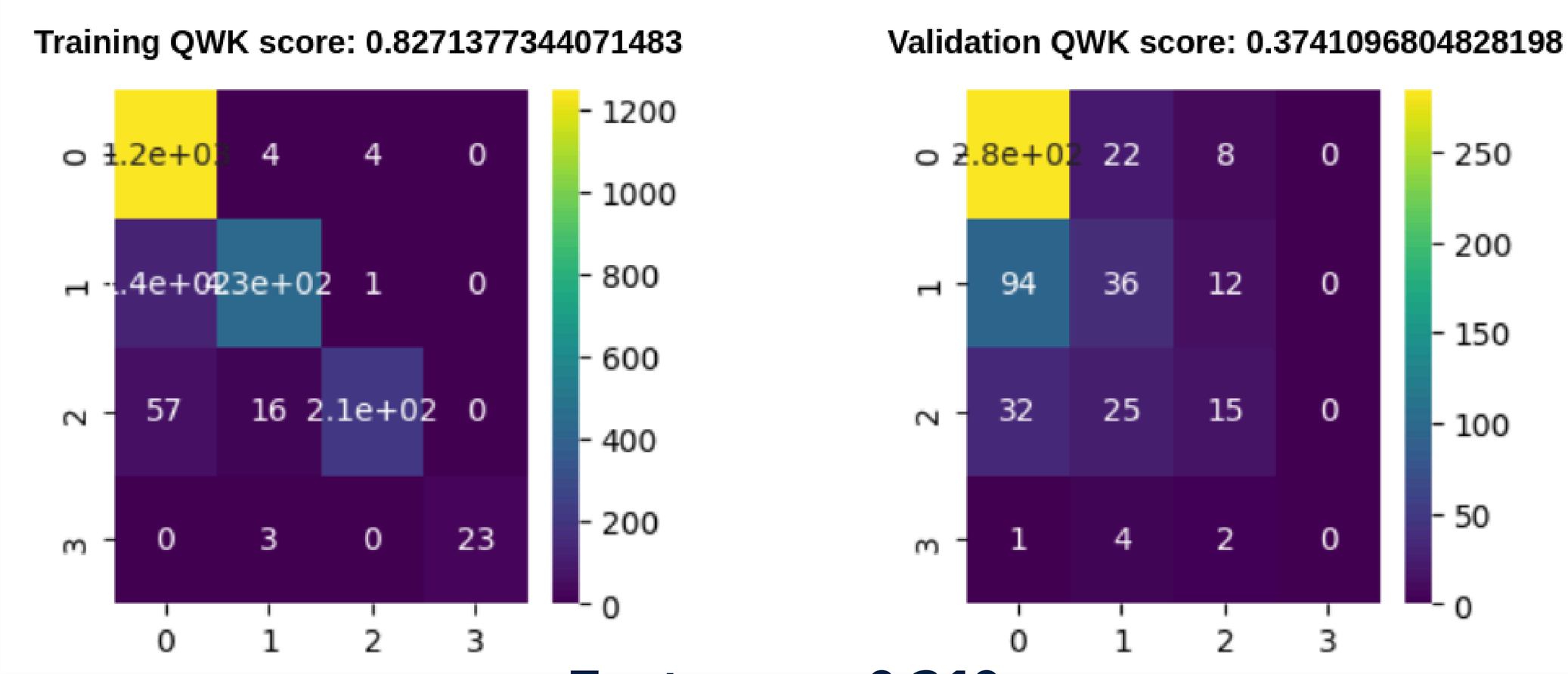
```
param_grid = {
        'n_estimators': [100],
        'max_depth' : [6, 8],
        'random_state': [0,1, 42, 30, 35],
        'bootstrap': [True],
        'criterion': ['entropy'],
        'max_features': [None, 'sqrt', 'log2']
}
```

Grid search guided by Quadratic Kappa Score

```
forest = GridSearchCV(RandomForestClassifier(), rf_param, scoring=kappa_scorer, cv=5, verbose=3)
forest.fit(X_train, Y_train)
```

6. Model training and validation

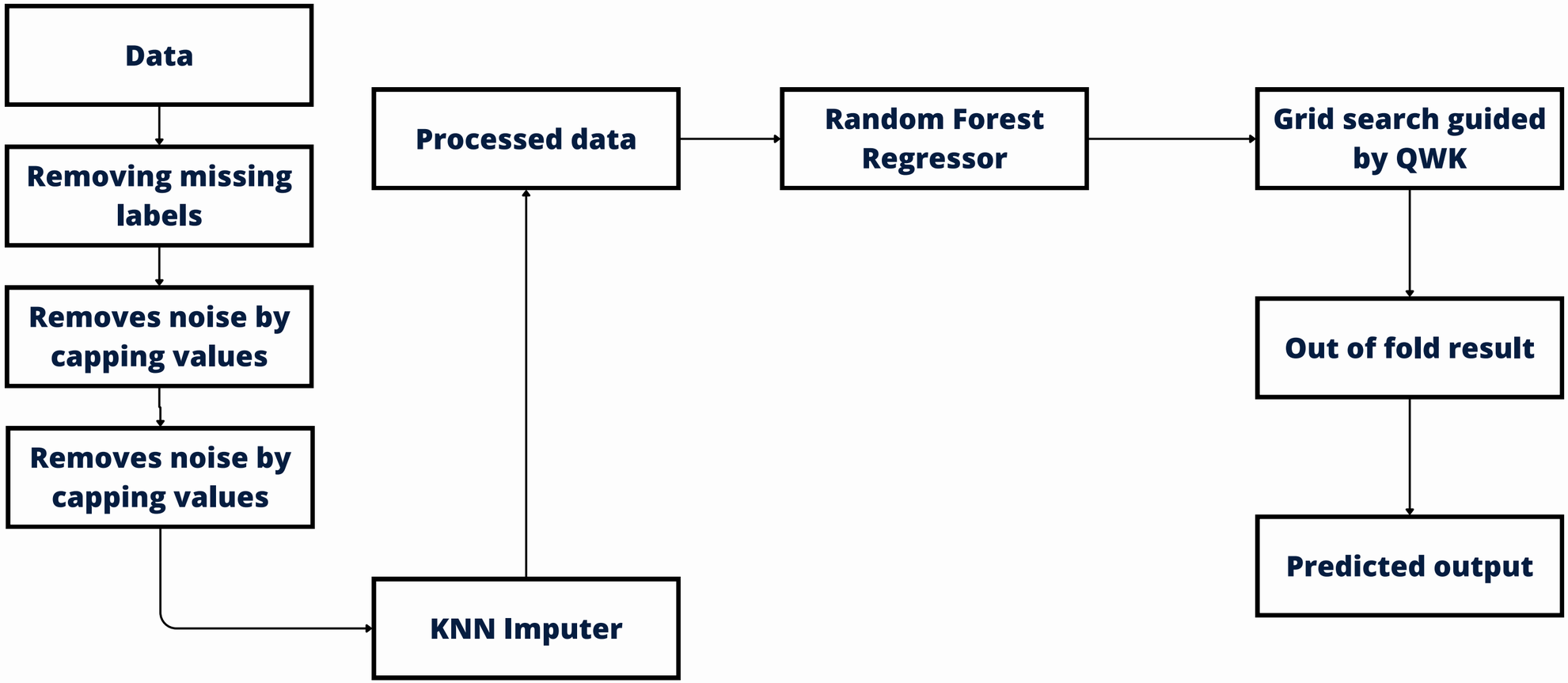
6.2. Version 2.0



Test score: 0.310

Conclusion

Overview of our version 2.0



Data processing pipeline

6.Model training and validation

Version 2.1 (0.403)

Including string data type

Balanced weight

6. Model training and validation

6.3. Version 2.1

Better grid search parameters and balanced weight parameter

sii	
0.0	1568
1.0	711
2.0	359
3.0	33

```
# Using grid search
kappa_scorer = make_scorer(cohen_kappa_score, weights='quadratic')

rf_param = {
        'n_estimators': [100],
        'max_depth'   : [6, 8],
        'random_state': [42],
        'bootstrap'   : [True],
        'criterion'   : ['entropy'],
        'max_features': [None, 'sqrt', 'log2'],
        'class_weight': [None, 'balanced']
}

forest = RandomForestClassifier
```

6. Model training and validation

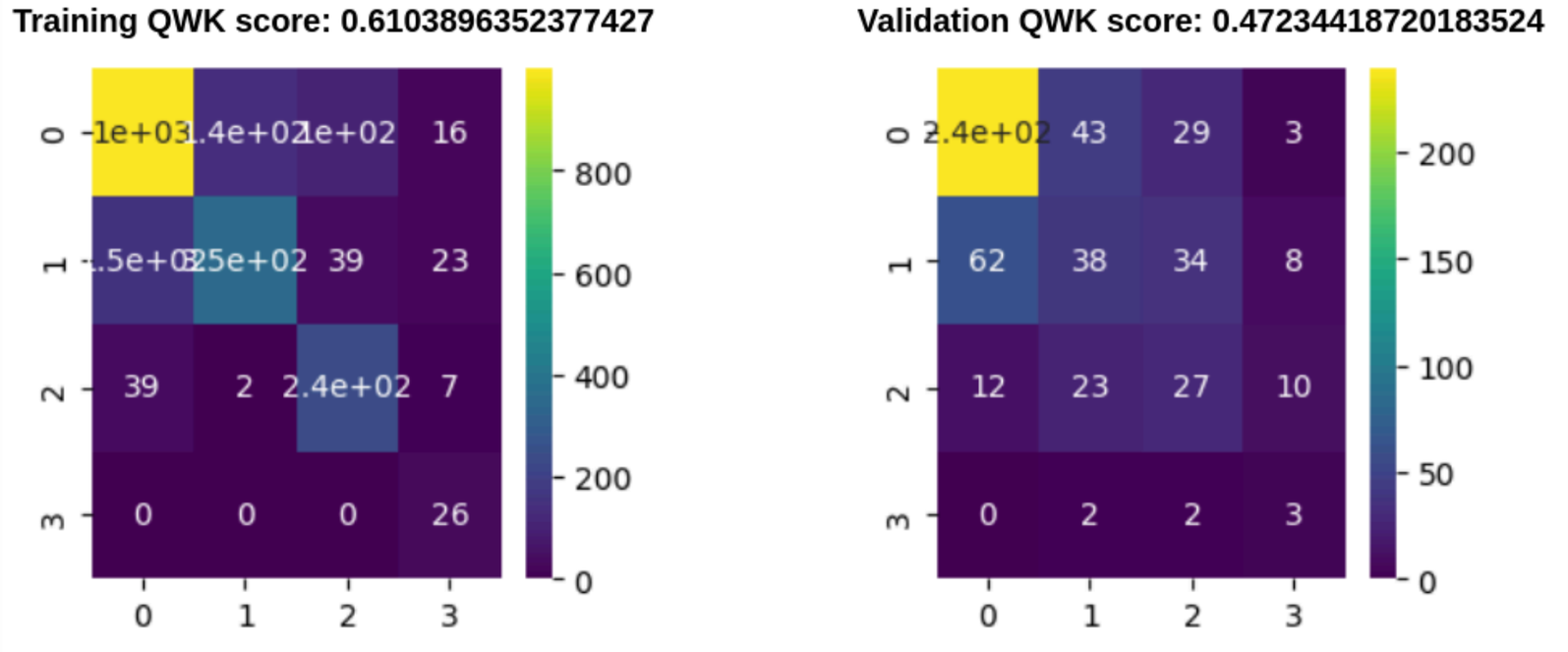
6.3. Version 2.1

```
# Encode
season_mapping = {
    'Spring' : 1,
    'Sumemr' : 2,
    'Fall'   : 3,
    'Winter' : 4
}
```

We chose ordinal mapping because it’s simple and tree-based models handle ordinal and nominal data similarly

6. Model training and validation

6.3. Version 2.1



Test score: 0.403

6.Model training and validation

Version 3.0 (0.421)

Add more nosie

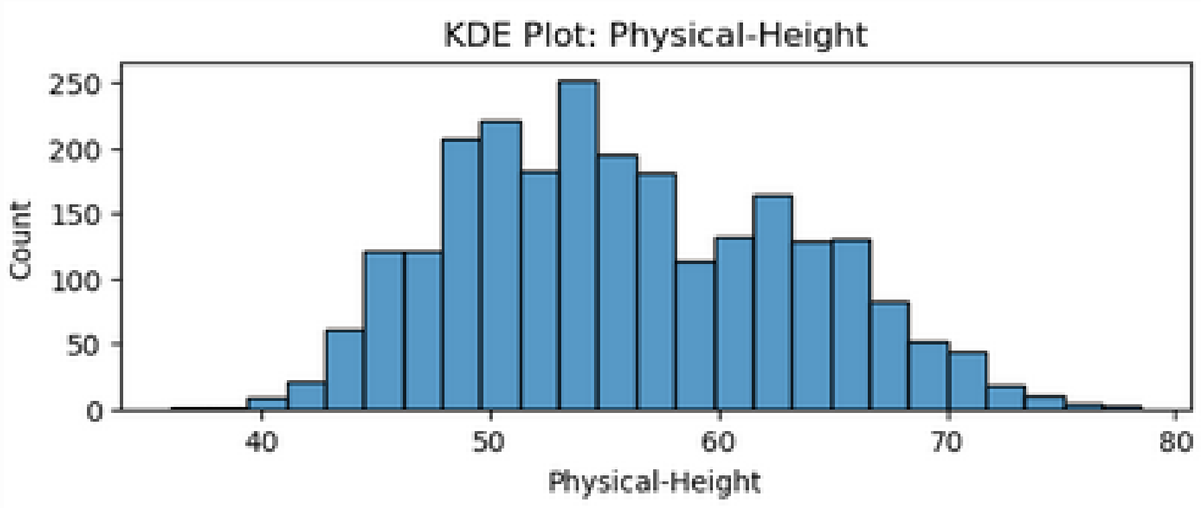
Optimizing SII boundary

Using XGBOOST regressor instead of classifier

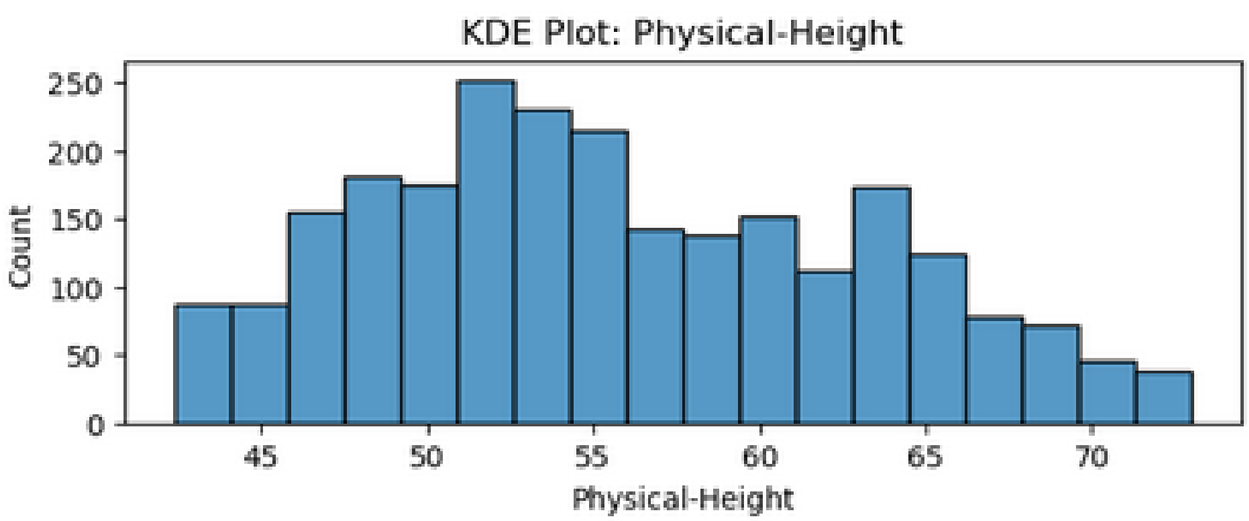
Using K-Fold cross validation

6. Model training and validation

6.4. Version 3.0



Before capping



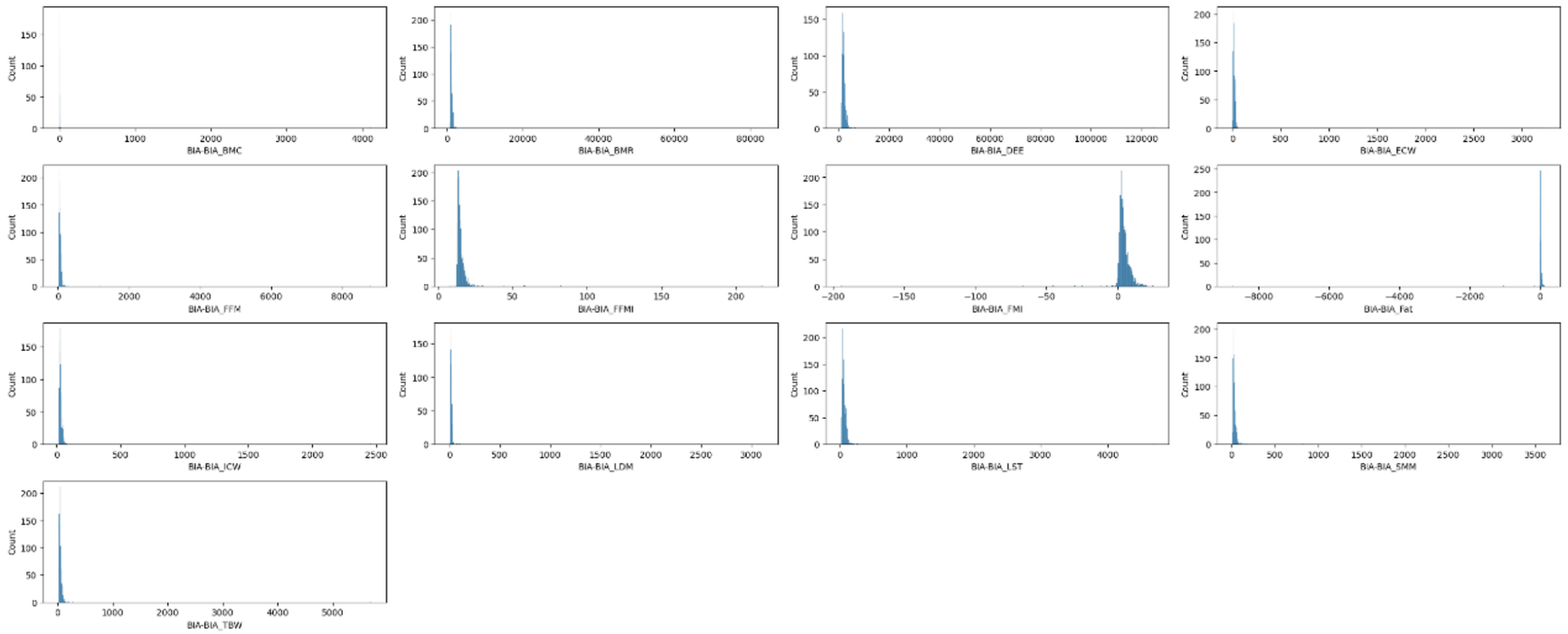
After capping

Quantile capping all features might not be a good idea

6. Model training and validation

6.4. Version 3.0

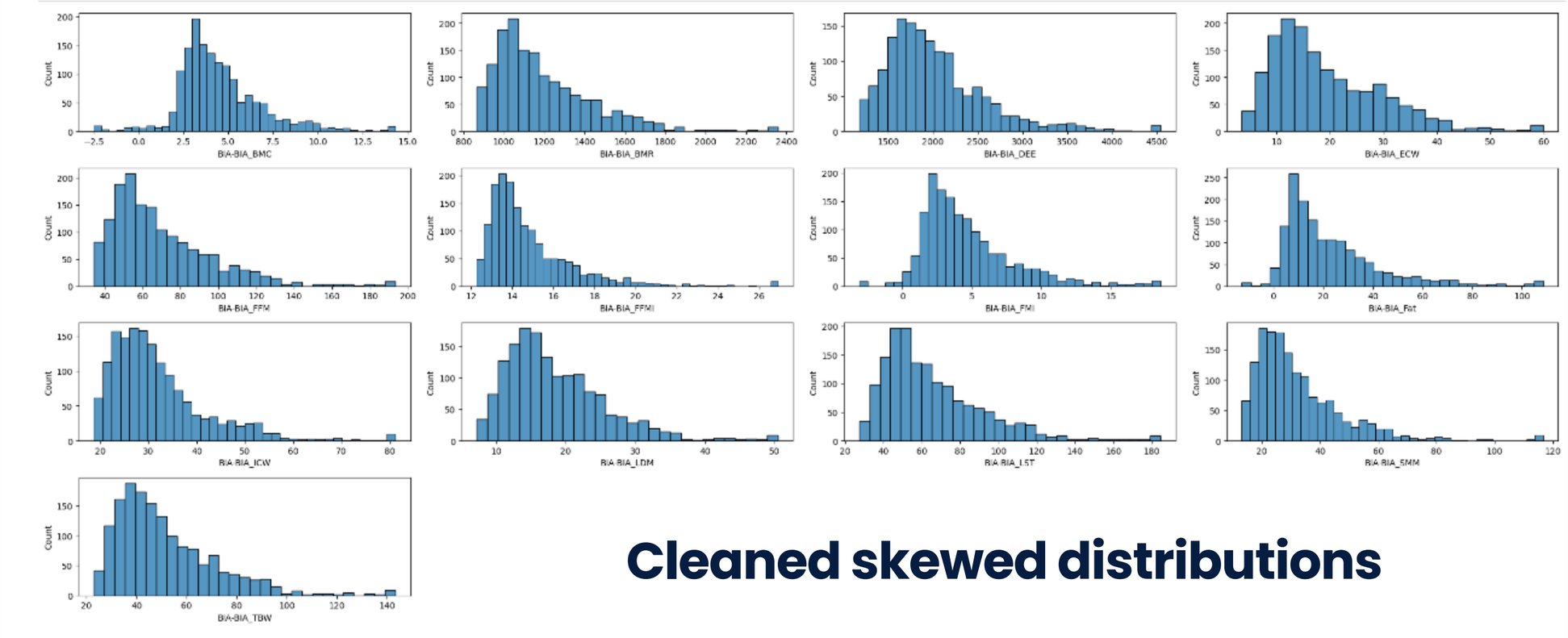
```
# Handling skewed data
skewed_data = ['BIA-BIA_BMC', 'BIA-BIA_BMR', 'BIA-BIA_DEE', 'BIA-BIA_ECW', 'BIA-BIA_FFM', 'BIA-BIA_FFMI',
               'BIA-BIA_FMI', 'BIA-BIA_Fat', 'BIA-BIA_ICW', 'BIA-BIA_LDM', 'BIA-BIA_LST', 'BIA-BIA_SMM', 'BIA-BIA_TBW']

plotFeaturesKDE(train_data[skewed_data])
```



Instead, we should only remove noise of extremely skewed distributions

6. Model training and validation

6.4. Version 3.0



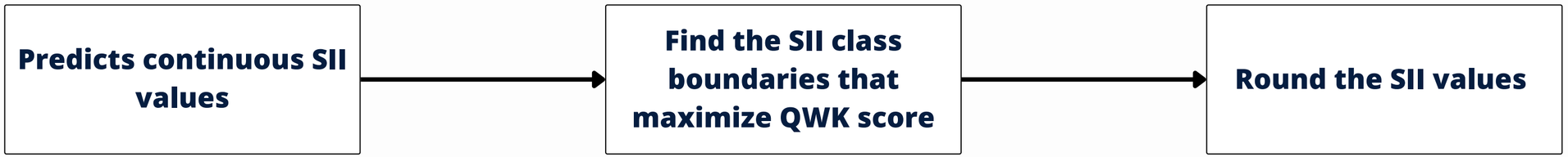
Cleaned skewed distributions

Instead, we should only remove noise of extremely skewed distributions

6. Model training and validation

6.4. Version 3.0

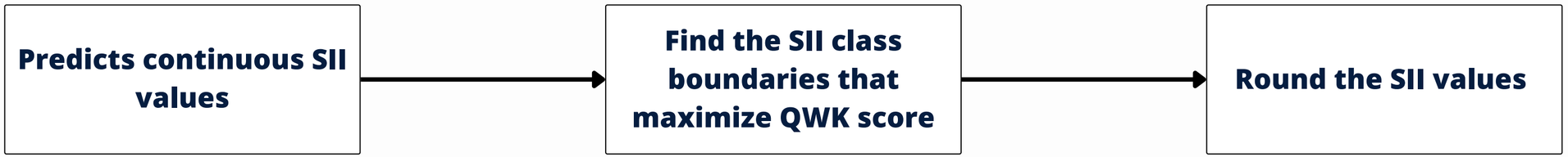
Method reference:
https://www.kaggle.com/code/kuosys/cmi-reproducible-results-fixseed-lgb-cpu-lb-494

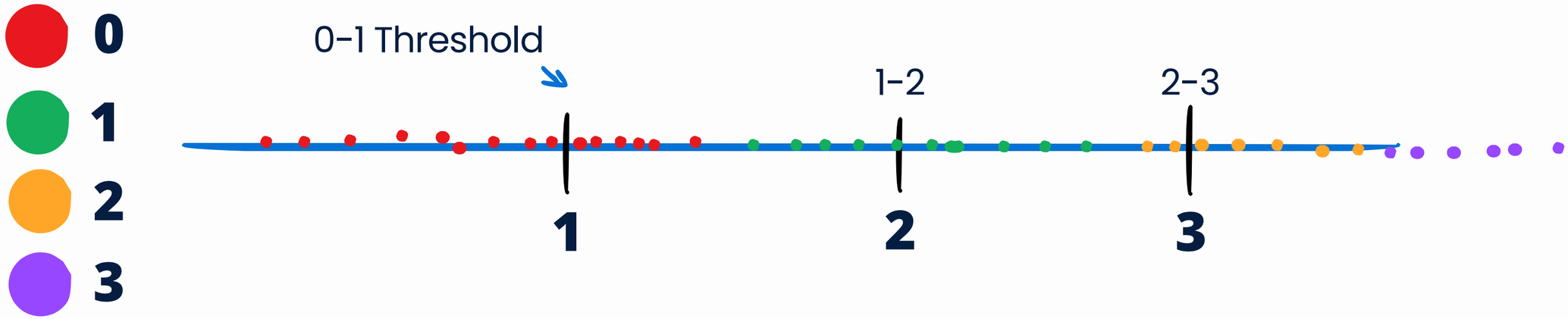


6. Model training and validation

6.4. Version 3.0

Method reference:
https://www.kaggle.com/code/kuosys/cmi-reproducible-results-fixseed-lgb-cpu-lb-494

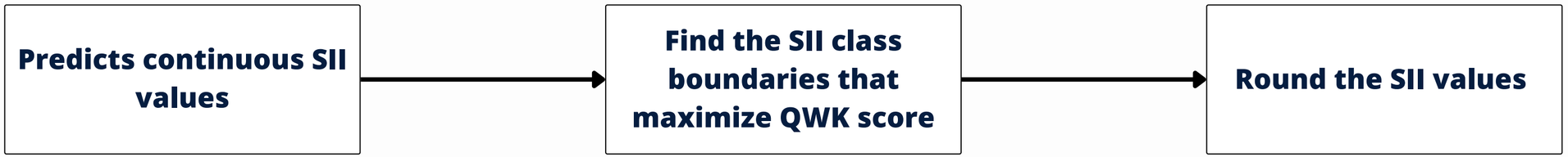


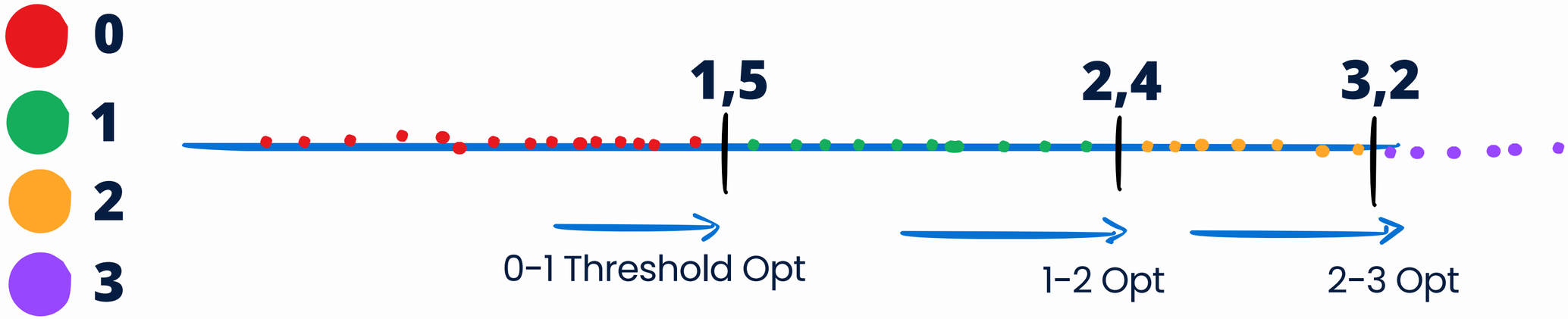


6. Model training and validation

6.4. Version 3.0

Method reference:
https://www.kaggle.com/code/kuosys/cmi-reproducible-results-fixseed-lgb-cpu-lb-494





6. Model training and validation

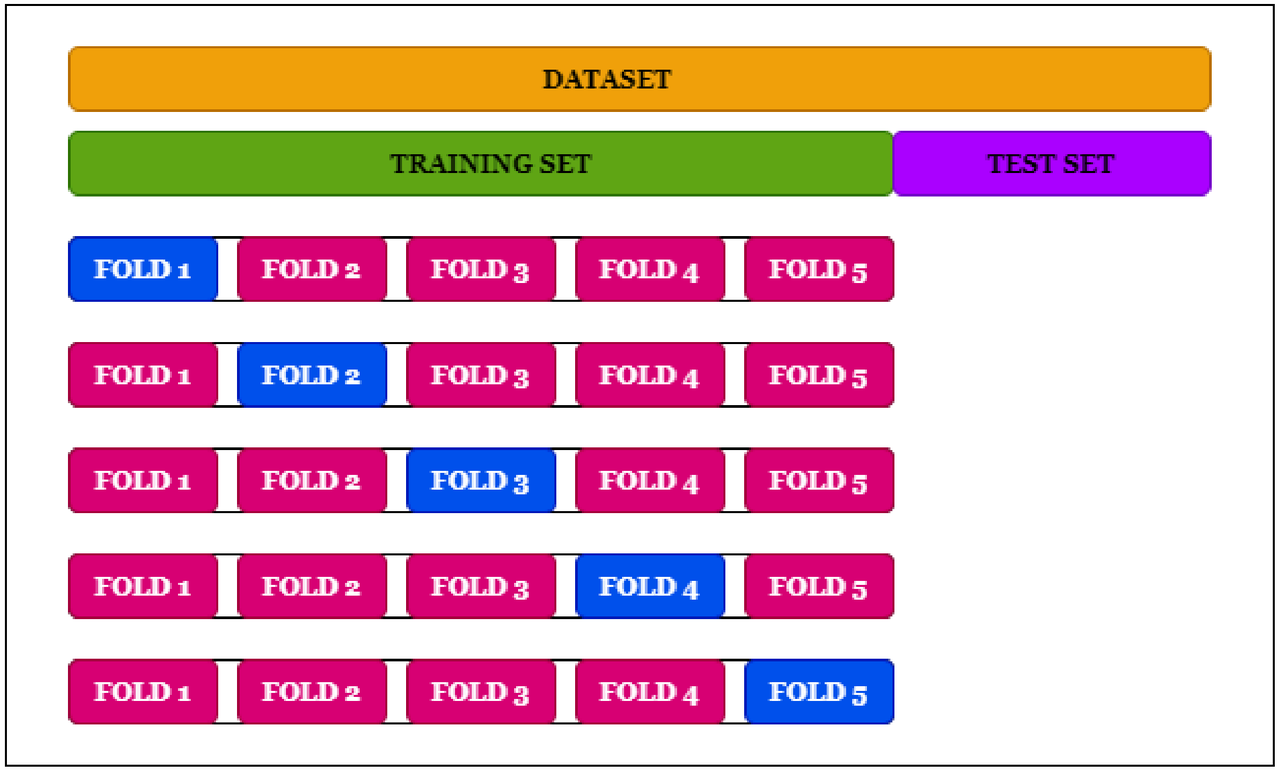
6.4. Version 3.0

Using regressor instead of classifier

```
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor

XGB_Params = {
    'learning_rate': 0.05,
    'max_depth': 6,
    'n_estimators': 200,
    'subsample': 0.8,
    'colsample_bytree': 0.8,
    'reg_alpha': 1,  # Increased from 0.1
    'reg_lambda': 5,  # Increased from 1
    'random_state': 42
}

xgb_reg = XGBRegressor(**XGB_Params)
```

6. Model training and validation

6.4. Version 3.0



6. Model training and validation

6.4. Version 3.0

Train QWK	**0.9231**
Validation QWK	**0.3869**
Optimized Validation QWK	**0.450**
Test QWK	**0.421**

The model is overfitting but we can improve!

6.Model training and validation

Version 3.1 (0.445)

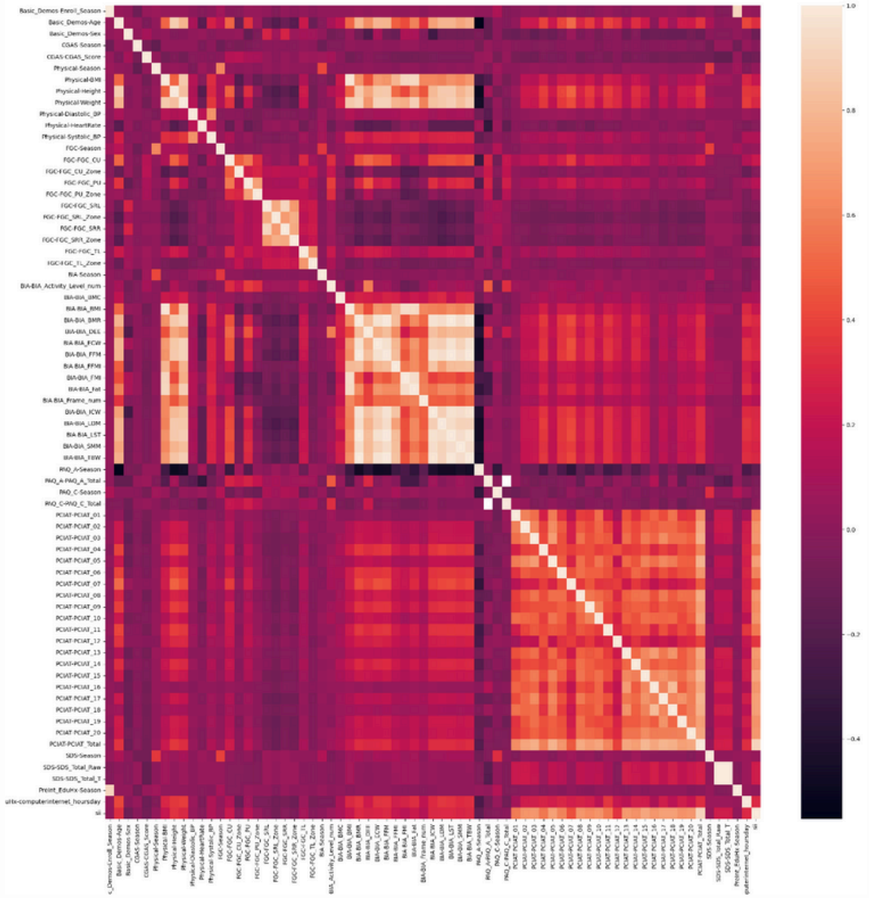
Ensembling RF and XGBoost

Feature engineering

6. Model training and validation

6.4. Version 3.0

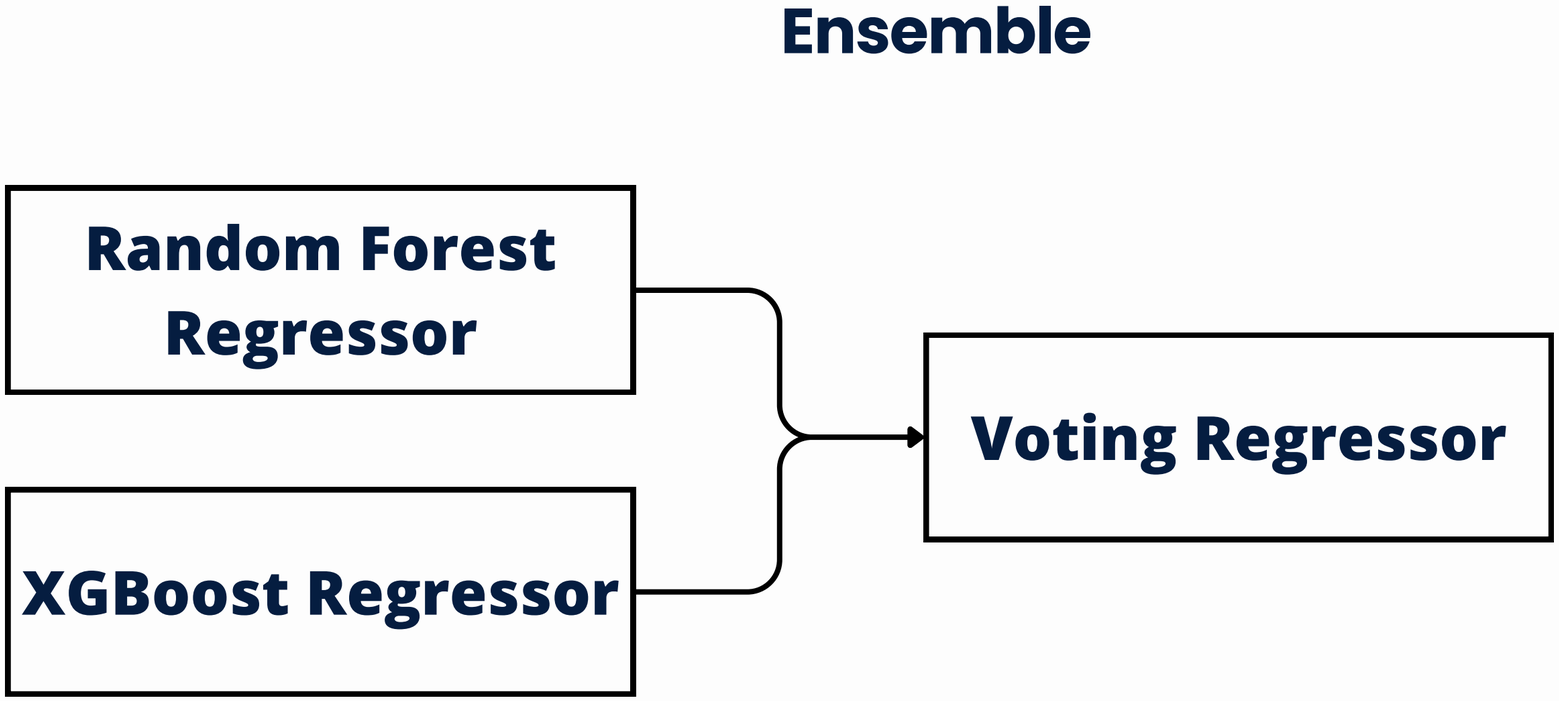
Based on correlation and domain knowledge



```python
def feature_engineering(df):
    # BIA
    df['ECW_TBW_Ratio'] = df['BIA-BIA_ECW'] / df['BIA-BIA_TBW']
    df['ICW_TBW_Ratio'] = df['BIA-BIA_ICW'] / df['BIA-BIA_TBW']
    df['Fat_Muscle_Ratio'] = df['BIA-BIA_Fat'] / df['BIA-BIA_SMM']
    df['BMI_Activity'] = df['BIA-BIA_BMI'] * df['BIA-BIA_Activity_Level_num']
    df['BMR_DEE_Interaction'] = df['BIA-BIA_BMR'] * df['BIA-BIA_DEE']
    df['TBW_Per_FFM'] = df['BIA-BIA_TBW'] / df['BIA-BIA_FFM']
    df['SMM_Per_FFM'] = df['BIA-BIA_SMM'] / df['BIA-BIA_FFM']

    # FGC
    df['healthy_zone'] = np.sum(df[['FGC-FGC_CU_Zone', 'FGC-FGC_PU_Zone',
                                    'FGC-FGC_SRL_Zone', 'FGC-FGC_SRR_Zone', 'FGC-FGC_TL_Zone']], axis=1)

    # Internet hour

    # Interacting features
    # Conver pounds to Kg
    df['Physical-Weight'] = df['Physical-Weight'] * 0.453592
    df['skeletal_muscle_ratio'] = df['BIA-BIA_SMM'] / df['Physical-Weight']
    df['lean_dry_ratio']  = df['BIA-BIA_LDM'] / df['Physical-Weight']
    df['lean_soft_ratio'] = df['BIA-BIA_LST'] / df['Physical-Weight']
    df['fat_free_ratio'] = df['BIA-BIA_FFM'] / df['Physical-Weight']
    df['mineral_ratio'] = df['BIA-BIA_BMC'] / 1000 / df['Physical-Weight'] # Gram to KG
    df['fat_ratio'] = df['BIA-BIA_Fat'] / 100

    df['internet_physical_bmi'] = df['PreInt_EduHx-computerinternet_hoursday'] * df['Physical-BMI']
    df['intertet_healthy_zone'] = df['PreInt_EduHx-computerinternet_hoursday'] * df['healthy_zone']
    df['internet_bia_bmi']  = df['PreInt_EduHx-computerinternet_hoursday'] * df['BIA-BIA_BMI']
    df['internet_activity'] = df['PreInt_EduHx-computerinternet_hoursday'] * df['BIA-BIA_Activity_Level_num']

    # Activity questionaire
    df['PAQ_Total'] = df[['PAQ_A-PAQ_A_Total', 'PAQ_C-PAQ_C_Total']].mean(axis=1, skipna=True)

    return df
```

6. Model training and validation

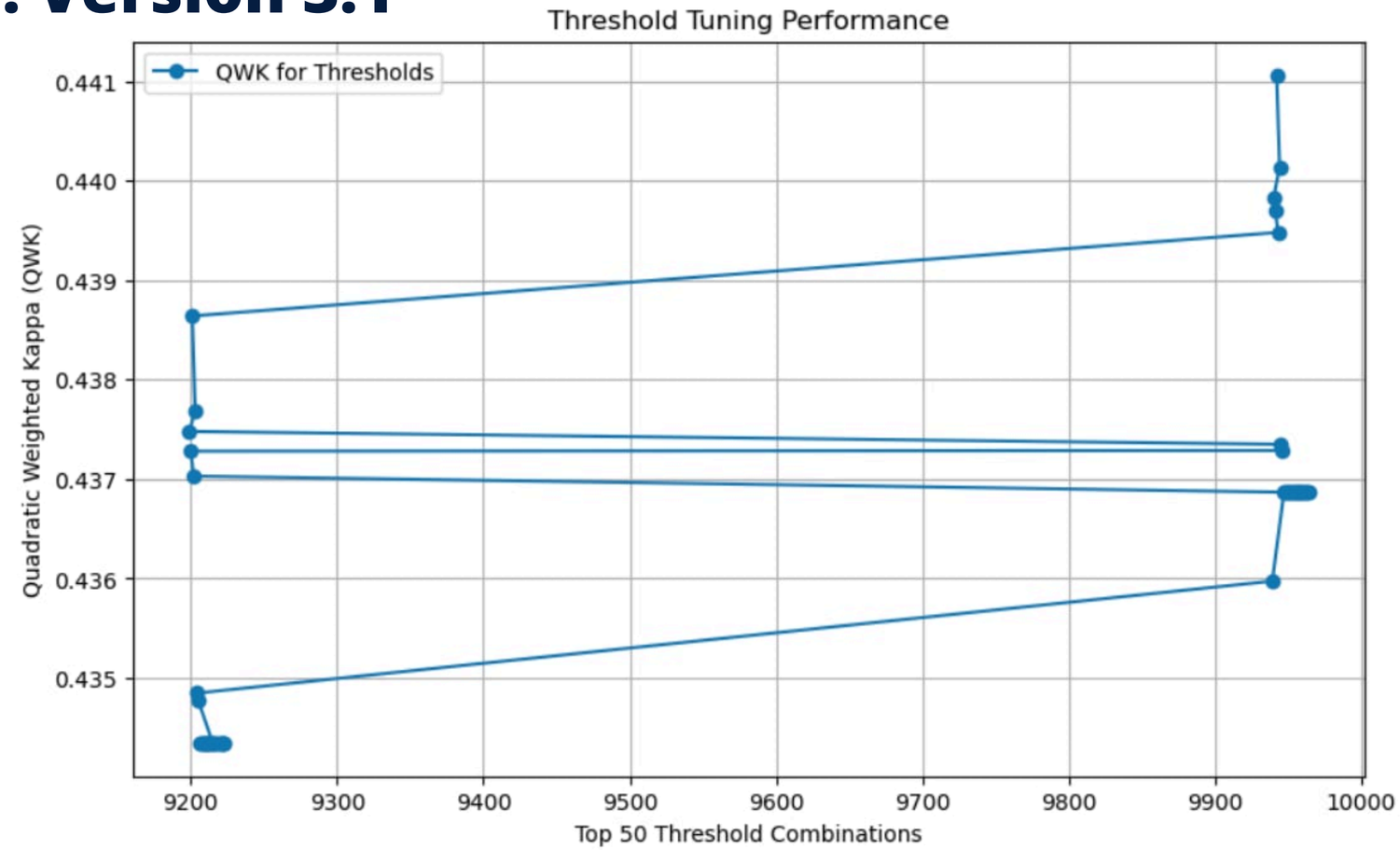
6.5. Version 3.1



Random Forest excels in stability and interpretability but may struggle with highly complex patterns.
XGBoost captures intricate relationships and fine-tunes performance but is prone to overfitting; combining them balances robustness and precision.

6. Model training and validation

6.5. Version 3.1

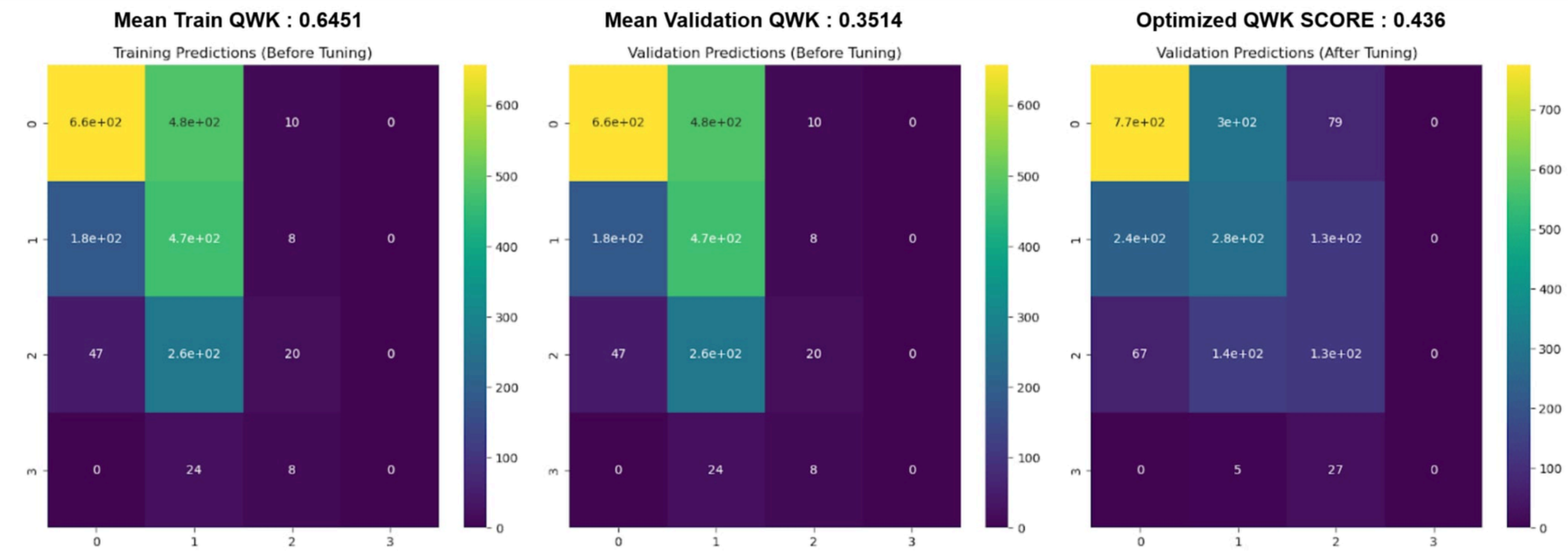


Threshold before tuning
[0.5, 1.5, 2.5]
Threshold after tuning
[0.60160294 1.0514105 2.71777567]

Using Nelder-Mead to minimize negative QWK

6. Model training and validation

6.4. Version 3.1

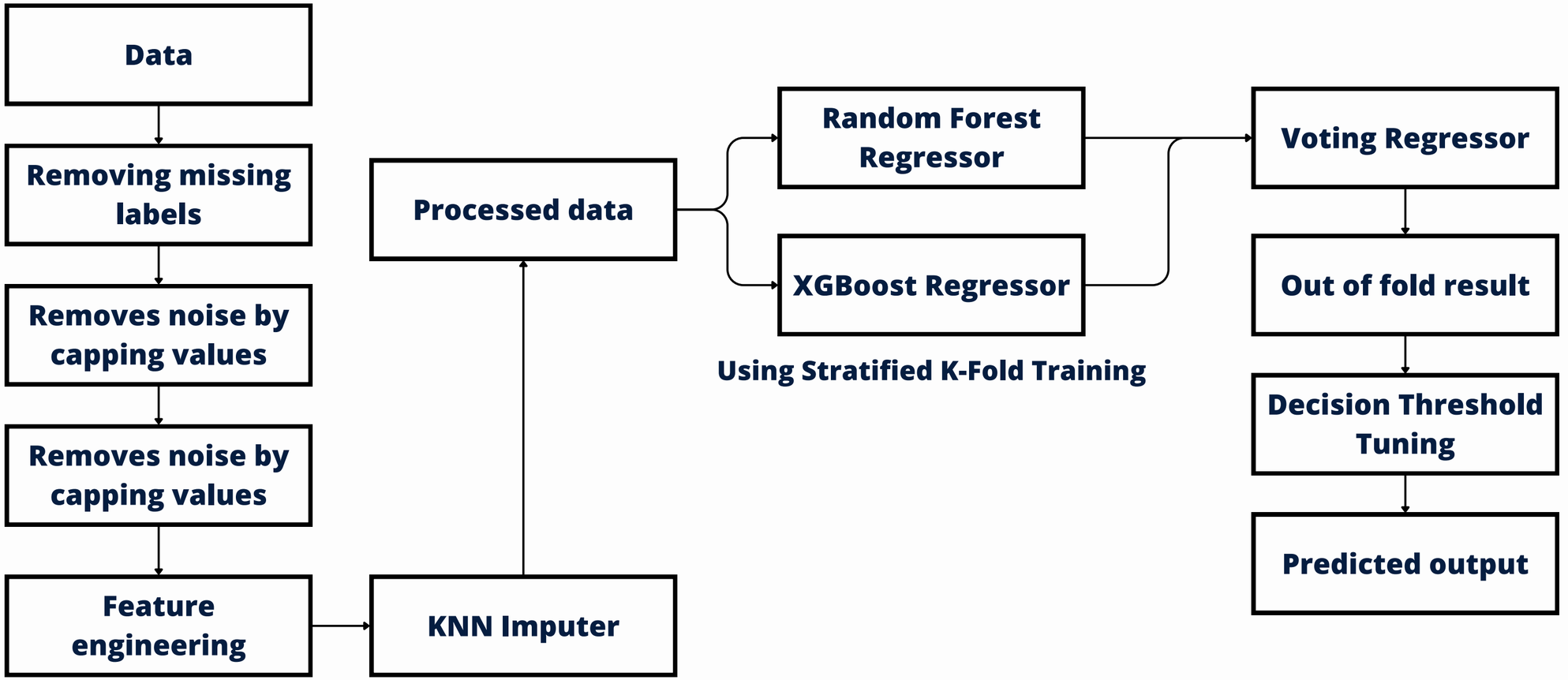


Tuning the decision threshold with QWK minimization

Test score: 0.445

Conclusion

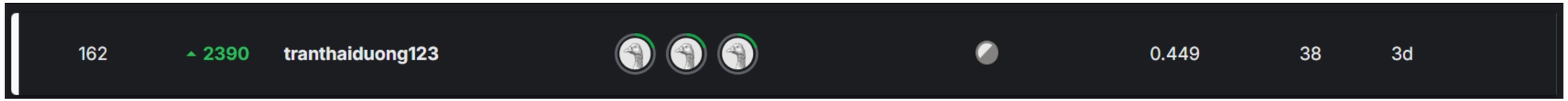
Overview of our best model



Conclusion

Version	Data							Model				Public Score	Private Score
	Missing feature imputation		Directly predicting sii		Season features		Feature engineering	XGBoost	Random Forest	Voting	Decision Threshold Tuning		
	Median	KNN	NO	YES	Remove	Encoding							
ver-1.0		x	x		x				classifier			0.237	0.238
ver-1.2.1		x	x		x				classifier			0.175	0.206
ver-1.2.2		x	x		x				classifier			0.216	0.248
ver-1.2.3		x	x		x				classifier			0.216	0.248
ver-1.2.4		x	x		x				classifier			0.212	0.243
ver-2.0		x		x	x				classifier			0.310	0.312
ver-2.1		x		x		x			classifier			0.403	0.450
ver-2.2	x			x		x	x	classifier				0.410	0.387
ver-3.0		x		x		x		regressor			x	0.415	0.421
ver-3.1		x		x		x	x	regressor	regressor	regressor	x	0.445	0.449

Final standing



Thanks for your listening 💕