Course: INFO 531: Data Warehousing and Analytics in the Cloud
Term name and year: Spring 2025
Student's Full Name: Kai Shuen Neo

Week 16 Final Project Report

Brief description of project

This project aims to make a prediction on the level of problematic internet use exhibited by children by analysing their physical activity through fitness data. This aids in identifying early signs and indicators of problematic internet and technology use in children, allowing for prompt interventions in the earlier stages of recognising this problem that they might have and to eventually encourage and inculcate healthier digital habits in children. In the long-term, this better equips children in navigating the digital landscape responsibly.

Data source

The data that would be used would be the data collected from the Healthy Brain Network (HBN) dataset, a clinical sample containing the health data of 5000 individuals aged 5-22 years old who have undergone clinical and research screenings. The dataset consists of features that include individuals' physical activity data and internet usage behaviour data, with response being individuals' Severity Impairment Index (SII), a measure of the problematic internet use. The source of data is from Kaggle: https://www.kaggle.com/competitions/child-mind-institute-problematic-internet-use/data

Data

Data files would include a train.csv file and a test.csv file.

Tools

The tool that would be used would be a Jupyter Notebook (Anaconda) that runs on Python3.

Data Preparation Plan

Data preparation would involve data discovery, data cleaning and data transformation to preprocess data to allow it to be suitable for analysis and modelling.

In terms of data discovery, we would first need to understand the data collected. This would mean checking the data structure and data type of predictor and response variables, and thereafter, identifying data quality issues that could surface in the form of corrupted values, missing values, imbalanced and standardless data. It is critical that we filter them out at this step as data quality issues could potentially affect our analysis further on. We would also seek to understand our data better through the use of descriptive statistics and data visualizations by checking the distributions of variables and correlation between variables.

Subsequently, after the data discovery step, we would identify the data quality issues observed in the data cleaning step and fix them in the data transformation step. This would involve correcting, creating, converting and completing data. To handle missing values ('NaN' / 'NA' cells), we could consider omitting them completely, or replacing them using mean/median/mode imputation, depending on the amount of missing data so as to complete the dataset. In the event that data duplicates are present, we would remove them to prevent skewed results. As for data inconsistencies, we could do standardization and normalization to ensure consistent formatting across all variables so as to achieve data accuracy. For instance, we could map strings and time into numbers so that the data can be processed by the code more easily. This could also involve converting variable types from categorical to numerical. We could also delete variables that are meaningless and create new variables so that the data can be better generalized. We would also need to correct possible outliers, unacceptable data inputs and unreasonable values by replacing them with NaN for further processing.

Finally, upon the completion of data preprocessing, we can then use the cleaned and processed data for exploratory data analysis and statistical evaluation to test our hypotheses. We can also utilize several Machine Learning techniques to fit models that could aid in making our predictions.

Actual code and output generated from data preparation

[1]:
```
# topic: Relating physical activity to problematic internet use
# objective: predict sii using physical activity fitness data (via classification)
# important features include - Parent-Child Internet Addiction Test (PCIAT).
```

[3]:
```
# import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
from scipy import stats
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, fbeta_score, roc_auc_score, classificat
```

Figure 1: import libraries

[4]:
```
# import training dataset
train_df = pd.read_csv('train.csv')
train_df
```

[4]:

	id	Basic_Demos-Enroll_Season	Basic_Demos-Age	Basic_Demos-Sex	CGAS-Season	CGAS-CGAS_Score	Physical-Season	Physical-BMI	Physical-Height	Physical-Weight	...	PCIAT-PCIAT_18	PCIAT-PCIAT_19	PC PCIAT
0	00008ff9	Fall	5	0	Winter	51.0	Fall	16.877316	46.0	50.8	...	4.0	2.0	
1	000fd460	Summer	9	0	NaN	NaN	Fall	14.035590	48.0	46.0	...	0.0	0.0	
2	00105258	Summer	10	1	Fall	71.0	Fall	16.648696	56.5	75.6	...	2.0	1.0	
3	00115b9f	Winter	9	0	Fall	71.0	Summer	18.292347	56.0	81.6	...	3.0	4.0	
4	0016bb22	Spring	18	1	Summer	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3955	ff8a2de4	Fall	13	0	Spring	60.0	Fall	16.362460	59.5	82.4	...	1.0	1.0	
3956	ffa9794a	Winter	10	0	NaN	NaN	Spring	18.764678	53.5	76.4	...	NaN	NaN	
3957	ffcd4dbd	Fall	11	0	Spring	68.0	Winter	21.441500	60.0	109.8	...	1.0	0.0	
3958	ffed1dd5	Spring	13	0	Spring	70.0	Winter	12.235895	70.7	87.0	...	1.0	1.0	
3959	ffef538e	Spring	11	0	NaN	NaN	Winter	NaN	NaN	NaN	...	NaN	NaN	

3960 rows × 82 columns

Figure 2: import training dataset

```
[7]: # import test dataset
     test_df = pd.read_csv('test.csv')
     test_df
```

[7]:

	id	Basic_Demos-Enroll_Season	Basic_Demos-Age	Basic_Demos-Sex	CGAS-Season	CGAS-CGAS_Score	Physical-Season	Physical-BMI	Physical-Height	Physical-Weight	...	BIA-BIA_TBW	PAQ_A-Season	PAQ_PAQ_A_Tc
0	00008ff9	Fall	5	0	Winter	51.0	Fall	16.877316	46.00	50.8	...	32.6909	NaN	N
1	000fd460	Summer	9	0	NaN	NaN	Fall	14.035590	48.00	46.0	...	27.0552	NaN	N
2	00105258	Summer	10	1	Fall	71.0	Fall	16.648696	56.50	75.6	...	NaN	NaN	N
3	00115b9f	Winter	9	0	Fall	71.0	Summer	18.292347	56.00	81.6	...	45.9966	NaN	N
4	0016bb22	Spring	18	1	Summer	NaN	NaN	NaN	NaN	NaN	...	NaN	Summer	1
5	001f3379	Spring	13	1	Winter	50.0	Summer	22.279952	59.50	112.2	...	63.1265	NaN	N
6	0038ba98	Fall	10	0	NaN	NaN	Fall	19.660760	55.00	84.6	...	47.2211	NaN	N
7	0068a485	Fall	10	1	NaN	NaN	Fall	16.861286	59.25	84.2	...	50.4767	NaN	N
8	0069fbed	Summer	15	0	NaN	NaN	Spring	NaN	NaN	NaN	...	NaN	NaN	N
9	0083e397	Summer	19	1	Summer	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
10	0087dd65	Spring	11	1	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
11	00abe655	Fall	11	0	Summer	66.0	NaN	NaN	NaN	NaN	...	NaN	NaN	N
12	00ae59c9	Fall	13	0	NaN	NaN	Winter	21.079065	57.75	100.0	...	56.0118	NaN	N
13	00af6387	Spring	12	0	NaN	NaN	Spring	15.544111	60.00	79.6	...	NaN	NaN	N
14	00bd4359	Spring	12	0	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
15	00c0cd71	Winter	7	0	Summer	51.0	Spring	29.315775	54.00	121.6	...	NaN	NaN	N
16	00d56d4b	Spring	5	1	Summer	80.0	Spring	17.284504	44.00	47.6	...	NaN	NaN	N
17	00d9913d	Fall	10	1	NaN	NaN	Fall	19.893157	55.00	85.6	...	NaN	NaN	N
18	00e6167c	Winter	6	0	Spring	60.0	Winter	30.094649	37.50	60.2	...	38.7638	NaN	N
19	00ebc35d	Winter	10	0	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N

20 rows × 59 columns

Figure 3: import test dataset

```
[9]: # data exploration
```

```
[11]: train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3960 entries, 0 to 3959
Data columns (total 82 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   id                                      3960 non-null   object
 1   Basic_Demos-Enroll_Season               3960 non-null   object
 2   Basic_Demos-Age                         3960 non-null   int64
 3   Basic_Demos-Sex                         3960 non-null   int64
 4   CGAS-Season                             2555 non-null   object
 5   CGAS-CGAS_Score                         2421 non-null   float64
 6   Physical-Season                         3310 non-null   object
 7   Physical-BMI                            3022 non-null   float64
 8   Physical-Height                         3027 non-null   float64
 9   Physical-Weight                         3076 non-null   float64
 10  Physical-Waist_Circumference            898 non-null    float64
 11  Physical-Diastolic_BP                   2954 non-null   float64
 12  Physical-HeartRate                      2967 non-null   float64
 13  Physical-Systolic_BP                    2954 non-null   float64
 14  Fitness_Endurance-Season                1308 non-null   object
 15  Fitness_Endurance-Max_Stage             743 non-null    float64
 16  Fitness_Endurance-Time_Mins             740 non-null    float64
 17  Fitness_Endurance-Time_Sec              740 non-null    float64
 18  FGC-Season                              3346 non-null   object
 19  FGC-FGC_CU                              2322 non-null   float64
 20  FGC-FGC_CU_Zone                         2282 non-null   float64
 21  FGC-FGC_GSND                            1074 non-null   float64
 22  FGC-FGC_GSND_Zone                       1062 non-null   float64
 23  FGC-FGC_GSD                             1074 non-null   float64
 24  FGC-FGC_GSD_Zone                        1063 non-null   float64
 25  FGC-FGC_PU                              2310 non-null   float64
 26  FGC-FGC_PU_Zone                         2271 non-null   float64
 27  FGC-FGC_SRL                             2305 non-null   float64
 28  FGC-FGC_SRL_Zone                        2267 non-null   float64
 29  FGC-FGC_SRR                             2307 non-null   float64
 30  FGC-FGC_SRR_Zone                        2269 non-null   float64
 31  FGC-FGC_TL                              2324 non-null   float64
 32  FGC-FGC_TL_Zone                         2285 non-null   float64
 33  BIA-Season                              2145 non-null   object
 34  BIA-BIA_Activity_Level_num              1991 non-null   float64
 35  BIA-BIA_BMC                             1991 non-null   float64
 36  BIA-BIA_BMI                             1991 non-null   float64
 37  BIA-BIA_BMR                             1991 non-null   float64
 38  BIA-BIA_DEE                             1991 non-null   float64
 39  BIA-BIA_ECW                             1991 non-null   float64
 40  BIA-BIA_FFM                             1991 non-null   float64
 41  BIA-BIA_FFMI                            1991 non-null   float64
 42  BIA-BIA_FMI                             1991 non-null   float64
 43  BIA-BIA_Fat                             1991 non-null   float64
 44  BIA-BIA_Frame_num                       1991 non-null   float64
 45  BIA-BIA_ICW                             1991 non-null   float64
 46  BIA-BIA_LDM                             1991 non-null   float64
 47  BIA-BIA_LST                             1991 non-null   float64
 48  BIA-BIA_SMM                             1991 non-null   float64
 49  BIA-BIA_TBW                             1991 non-null   float64
 50  PAQ_A-Season                            475 non-null    object
 51  PAQ_A-PAQ_A_Total                       475 non-null    float64
 52  PAQ_C-Season                            1721 non-null   object
 53  PAQ_C-PAQ_C_Total                       1721 non-null   float64
 54  PCIAT-Season                            2736 non-null   object
 55  PCIAT-PCIAT_01                          2733 non-null   float64
 56  PCIAT-PCIAT_02                          2734 non-null   float64
 57  PCIAT-PCIAT_03                          2731 non-null   float64
 58  PCIAT-PCIAT_04                          2731 non-null   float64
 59  PCIAT-PCIAT_05                          2729 non-null   float64
 60  PCIAT-PCIAT_06                          2732 non-null   float64
 61  PCIAT-PCIAT_07                          2729 non-null   float64
 62  PCIAT-PCIAT_08                          2730 non-null   float64
 63  PCIAT-PCIAT_09                          2730 non-null   float64
 64  PCIAT-PCIAT_10                          2733 non-null   float64
 65  PCIAT-PCIAT_11                          2734 non-null   float64
 66  PCIAT-PCIAT_12                          2731 non-null   float64
 67  PCIAT-PCIAT_13                          2729 non-null   float64
 68  PCIAT-PCIAT_14                          2732 non-null   float64
 69  PCIAT-PCIAT_15                          2730 non-null   float64
```

Figure 4: about the training dataset

```
 70  PCIAT-PCIAT_16                          2728 non-null   float64
 71  PCIAT-PCIAT_17                          2725 non-null   float64
 72  PCIAT-PCIAT_18                          2728 non-null   float64
 73  PCIAT-PCIAT_19                          2730 non-null   float64
 74  PCIAT-PCIAT_20                          2733 non-null   float64
 75  PCIAT-PCIAT_Total                       2736 non-null   float64
 76  SDS-Season                              2618 non-null   object
 77  SDS-SDS_Total_Raw                       2609 non-null   float64
 78  SDS-SDS_Total_T                         2606 non-null   float64
 79  PreInt_EduHx-Season                     3540 non-null   object
 80  PreInt_EduHx-computerinternet_hoursday  3301 non-null   float64
 81  sii                                     2736 non-null   float64
dtypes: float64(68), int64(2), object(12)
memory usage: 2.5+ MB
```

```
[13]: test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 59 columns):
 #   Column                                  Non-Null Count  Dtype  
---  ------                                  --------------  -----  
 0   id                                      20 non-null     object 
 1   Basic_Demos-Enroll_Season               20 non-null     object 
 2   Basic_Demos-Age                         20 non-null     int64  
 3   Basic_Demos-Sex                         20 non-null     int64  
 4   CGAS-Season                             10 non-null     object 
 5   CGAS-CGAS_Score                         8 non-null      float64
 6   Physical-Season                         14 non-null     object 
 7   Physical-BMI                            13 non-null     float64
 8   Physical-Height                         13 non-null     float64
 9   Physical-Weight                         13 non-null     float64
 10  Physical-Waist_Circumference            5 non-null      float64
 11  Physical-Diastolic_BP                   11 non-null     float64
 12  Physical-HeartRate                      12 non-null     float64
 13  Physical-Systolic_BP                    11 non-null     float64
 14  Fitness_Endurance-Season                4 non-null      object 
 15  Fitness_Endurance-Max_Stage             3 non-null      float64
 16  Fitness_Endurance-Time_Mins             3 non-null      float64
 17  Fitness_Endurance-Time_Sec              3 non-null      float64
 18  FGC-Season                              17 non-null     object 
 19  FGC-FGC_CU                              13 non-null     float64
 20  FGC-FGC_CU_Zone                         13 non-null     float64
 21  FGC-FGC_GSND                            5 non-null      float64
 22  FGC-FGC_GSND_Zone                       5 non-null      float64
 23  FGC-FGC_GSD                             5 non-null      float64
 24  FGC-FGC_GSD_Zone                        5 non-null      float64
 25  FGC-FGC_PU                              13 non-null     float64
 26  FGC-FGC_PU_Zone                         13 non-null     float64
 27  FGC-FGC_SRL                             13 non-null     float64
 28  FGC-FGC_SRL_Zone                        13 non-null     float64
 29  FGC-FGC_SRR                             13 non-null     float64
 30  FGC-FGC_SRR_Zone                        13 non-null     float64
 31  FGC-FGC_TL                              13 non-null     float64
 32  FGC-FGC_TL_Zone                         13 non-null     float64
 33  BIA-Season                              8 non-null      object 
 34  BIA-BIA_Activity_Level_num              8 non-null      float64
 35  BIA-BIA_BMC                             8 non-null      float64
 36  BIA-BIA_BMI                             8 non-null      float64
 37  BIA-BIA_BMR                             8 non-null      float64
 38  BIA-BIA_DEE                             8 non-null      float64
 39  BIA-BIA_ECW                             8 non-null      float64
 40  BIA-BIA_FFM                             8 non-null      float64
 41  BIA-BIA_FFMI                            8 non-null      float64
 42  BIA-BIA_FMI                             8 non-null      float64
 43  BIA-BIA_Fat                             8 non-null      float64
 44  BIA-BIA_Frame_num                       8 non-null      float64
 45  BIA-BIA_ICW                             8 non-null      float64
 46  BIA-BIA_LDM                             8 non-null      float64
 47  BIA-BIA_LST                             8 non-null      float64
 48  BIA-BIA_SMM                             8 non-null      float64
 49  BIA-BIA_TBW                             8 non-null      float64
 50  PAQ_A-Season                            1 non-null      object 
 51  PAQ_A-PAQ_A_Total                       1 non-null      float64
 52  PAQ_C-Season                            9 non-null      object 
 53  PAQ_C-PAQ_C_Total                       9 non-null      float64
 54  SDS-Season                              10 non-null     object 
 55  SDS-SDS_Total_Raw                       10 non-null     float64
 56  SDS-SDS_Total_T                         10 non-null     float64
 57  PreInt_EduHx-Season                     18 non-null     object 
 58  PreInt_EduHx-computerinternet_hoursday  16 non-null     float64
dtypes: float64(46), int64(2), object(11)
memory usage: 9.3+ KB
```

Figure 5: about the test dataset

```
[15]: train_df.shape
[15]: (3960, 82)
[17]: test_df.shape
[17]: (20, 59)
```

Figure 6: shape of df

```
[19]: # get column names
print(train_df.columns.values)
```

```
['id' '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' 'PCIAT-Season' 'PCIAT-PCIAT_01'
 'PCIAT-PCIAT_02' 'PCIAT-PCIAT_03' 'PCIAT-PCIAT_04' 'PCIAT-PCIAT_05'
 'PCIAT-PCIAT_06' 'PCIAT-PCIAT_07' 'PCIAT-PCIAT_08' 'PCIAT-PCIAT_09'
 'PCIAT-PCIAT_10' 'PCIAT-PCIAT_11' 'PCIAT-PCIAT_12' 'PCIAT-PCIAT_13'
 'PCIAT-PCIAT_14' 'PCIAT-PCIAT_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']
```

Figure 7: get column names

```
[21]: # preview first 5 rows
train_df.head()
```

[21]:

	id	Basic_Demos-Enroll_Season	Basic_Demos-Age	Basic_Demos-Sex	CGAS-Season	CGAS-CGAS_Score	Physical-Season	Physical-BMI	Physical-Height	Physical-Weight	...	PCIAT-PCIAT_18	PCIAT-PCIAT_19	PCIAT-PCIAT_20
0	00008ff9	Fall	5	0	Winter	51.0	Fall	16.877316	46.0	50.8	...	4.0	2.0	4.0
1	000fd460	Summer	9	0	NaN	NaN	Fall	14.035590	48.0	46.0	...	0.0	0.0	0.0
2	00105258	Summer	10	1	Fall	71.0	Fall	16.648696	56.5	75.6	...	2.0	1.0	1.0
3	00115b9f	Winter	9	0	Fall	71.0	Summer	18.292347	56.0	81.6	...	3.0	4.0	1.0
4	0016bb22	Spring	18	1	Summer	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN

5 rows × 82 columns

Figure 8: first 5 rows of train_df

```
[23]: # preview last 10 rows
train_df.tail(10)
```

[23]:

	id	Basic_Demos-Enroll_Season	Basic_Demos-Age	Basic_Demos-Sex	CGAS-Season	CGAS-CGAS_Score	Physical-Season	Physical-BMI	Physical-Height	Physical-Weight	...	PCIAT-PCIAT_18	PCIAT-PCIAT_19	PCI PCIAT_
3950	ff0ab367	Spring	9	0	NaN	NaN	Spring	20.200490	52.5	79.2	...	NaN	NaN	N
3951	ff18b749	Spring	7	0	NaN	NaN	Summer	14.768842	47.5	47.4	...	0.0	0.0	
3952	ff60112d	Summer	15	0	Spring	40.0	Winter	26.364710	70.5	186.4	...	1.0	1.0	
3953	ff6c2bb8	Fall	8	0	NaN	NaN	Fall	17.139810	52.5	67.2	...	2.0	2.0	
3954	ff759544	Summer	7	1	NaN	NaN	Summer	13.927006	48.5	46.6	...	3.0	3.0	
3955	ff8a2de4	Fall	13	0	Spring	60.0	Fall	16.362460	59.5	82.4	...	1.0	1.0	
3956	ffa9794a	Winter	10	0	NaN	NaN	Spring	18.764678	53.5	76.4	...	NaN	NaN	N
3957	ffcd4dbd	Fall	11	0	Spring	68.0	Winter	21.441500	60.0	109.8	...	1.0	0.0	
3958	ffed1dd5	Spring	13	0	Spring	70.0	Winter	12.235895	70.7	87.0	...	1.0	1.0	
3959	ffef538e	Spring	11	0	NaN	NaN	Winter	NaN	NaN	NaN	...	NaN	NaN	N

10 rows × 82 columns

Figure 9: last 10 rows of train_df

[25]:
```python
train_df.describe().transpose()
```

[25]:

	count	mean	std	min	25%	50%	75%	max
Basic_Demos-Age	3960.0	10.433586	3.574648	5.0	8.00000	10.000000	13.000000	22.000000
Basic_Demos-Sex	3960.0	0.372727	0.483591	0.0	0.00000	0.000000	1.000000	1.000000
CGAS-CGAS_Score	2421.0	65.454771	22.341862	25.0	59.00000	65.000000	75.000000	999.000000
Physical-BMI	3022.0	19.331929	5.113934	0.0	15.86935	17.937682	21.571244	59.132048
Physical-Height	3027.0	55.946713	7.473764	33.0	50.00000	55.000000	62.000000	78.500000
...	...	...	...	...	...	...	...	...
PCIAT-PCIAT_Total	2736.0	27.896199	20.338853	0.0	12.00000	26.000000	41.000000	93.000000
SDS-SDS_Total_Raw	2609.0	41.088923	10.427433	17.0	33.00000	39.000000	46.000000	96.000000
SDS-SDS_Total_T	2606.0	57.763622	13.196091	38.0	47.00000	55.000000	64.000000	100.000000
PreInt_EduHx-computerinternet_hoursday	3301.0	1.060588	1.094875	0.0	0.00000	1.000000	2.000000	3.000000
sii	2736.0	0.580409	0.771122	0.0	0.00000	0.000000	1.000000	3.000000

70 rows × 8 columns

Figure 10: summary statistics of training dataset

[27]:
```python
# data preprocessing
```

[29]:
```python
# analyse missing data
null_data = train_df.isna().sum().sort_values(ascending = False).head(46)
null_data = pd.DataFrame(null_data)
null_data = null_data.rename(columns={0:'Missing'})
null_data.style.background_gradient(cmap='YlOrRd')
```

Figure 11: analyse missing data

[29]:

	Missing
PAQ_A-PAQ_A_Total	3485
PAQ_A-Season	3485
Fitness_Endurance-Time_Sec	3220
Fitness_Endurance-Time_Mins	3220
Fitness_Endurance-Max_Stage	3217
Physical-Waist_Circumference	3062
FGC-FGC_GSND_Zone	2898
FGC-FGC_GSD_Zone	2897
FGC-FGC_GSD	2886
FGC-FGC_GSND	2886
Fitness_Endurance-Season	2652
PAQ_C-Season	2239
PAQ_C-PAQ_C_Total	2239
BIA-BIA_DEE	1969
BIA-BIA_FFM	1969
BIA-BIA_BMR	1969
BIA-BIA_BMI	1969
BIA-BIA_ECW	1969
BIA-BIA_FFMI	1969
BIA-BIA_FMI	1969
BIA-BIA_Activity_Level_num	1969
BIA-BIA_Fat	1969
BIA-BIA_Frame_num	1969
BIA-BIA_ICW	1969
BIA-BIA_LDM	1969
BIA-BIA_LST	1969
BIA-BIA_SMM	1969
BIA-BIA_TBW	1969
BIA-BIA_BMC	1969
BIA-Season	1815
FGC-FGC_SRL_Zone	1693
FGC-FGC_SRR_Zone	1691
FGC-FGC_PU_Zone	1689
FGC-FGC_CU_Zone	1678
FGC-FGC_TL_Zone	1675
FGC-FGC_SRL	1655
FGC-FGC_SRR	1653
FGC-FGC_PU	1650
FGC-FGC_CU	1638
FGC-FGC_TL	1636
CGAS-CGAS_Score	1539
CGAS-Season	1405
SDS-SDS_Total_T	1354
SDS-SDS_Total_Raw	1351
SDS-Season	1342
PCIAT-PCIAT_17	1235

Figure 12: missing data in train_df

[31]:
```python
# drop id column
train_df = train_df.drop(columns='id')
test_df = test_df.drop(columns='id')
```

[33]:
```python
# check for presence of null values
print(train_df.isnull().values.any())
print(test_df.isnull().values.any())
```

```
True
True
```

[35]:
```python
# remove data in train_df with null sii
train_df = train_df.dropna(subset=['sii'])
```

[37]:
```python
# return rows where 'sii' is NaN
train_df[train_df['sii'].isnull()]
```

[37]:

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	...	PCIAT-PCIAT_18	PCIAT-PCIAT_19	F

0 rows × 81 columns

Figure 13: data cleaning

[47]:
```python
# convert seasons into numeric encoding
# handle missing seasons data

train_df_cols = train_df.select_dtypes(exclude = 'number').columns
for season in train_df_cols:
    train_df[season] = train_df[season].fillna(0)
    train_df[season] = train_df[season].replace({'Spring': 1, 'Summer': 2, 'Fall': 3, 'Winter': 4})

test_df_cols = test_df.select_dtypes(exclude = 'number').columns
for season in test_df_cols:
    test_df[season] = test_df[season].fillna(0)
    test_df[season] = test_df[season].replace({'Spring': 1, 'Summer': 2, 'Fall': 3, 'Winter': 4})
```

[49]:
```python
train_df
```

[49]:

	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	...	PCIAT-PCIAT_18	PCI PCIAT
0	3	5	0	4	51.0	3	16.877316	46.0	50.8	NaN	...	4.0	
1	2	9	0	0	NaN	3	14.035590	48.0	46.0	22.0	...	0.0	
2	2	10	1	3	71.0	3	16.648696	56.5	75.6	NaN	...	2.0	
3	4	9	0	3	71.0	2	18.292347	56.0	81.6	NaN	...	3.0	
5	1	13	1	4	50.0	2	22.279952	59.5	112.2	NaN	...	1.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
3953	3	8	0	0	NaN	3	17.139810	52.5	67.2	25.0	...	2.0	
3954	2	7	1	0	NaN	2	13.927006	48.5	46.6	23.0	...	3.0	
3955	3	13	0	1	60.0	3	16.362460	59.5	82.4	NaN	...	1.0	
3957	3	11	0	1	68.0	4	21.441500	60.0	109.8	NaN	...	1.0	
3958	1	13	0	1	70.0	4	12.235895	70.7	87.0	NaN	...	1.0	

2736 rows × 81 columns

Figure 14: column label encoding

[51]:
```python
# pick a season column
# check for presnece of null
train_df[train_df['CGAS-Season'].isnull()]
```

[51]:

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	...	PCIAT-PCIAT_18	PCIAT-PCIAT_19	F

0 rows × 81 columns

Figure 15: check for absence of null values

```
[53]: # mark 50% as the threshold for columns with >50% non null values
# fill in missing values

# training dataset
threshold_train = 0.5 * len(train_df)
columns_with_data_train = train_df.columns[train_df.isnull().sum() < threshold_train]
train_df = train_df[columns_with_data_train]
# replace missing values with 0
train_df = train_df.fillna(0)

# testing dataset
threshold_test = 0.5 * len(test_df)
columns_with_data_test = test_df.columns[test_df.isnull().sum() < threshold_test]
test_df = test_df[columns_with_data_test]
# replace missing values with 0
test_df = test_df.fillna(0)
```

```
[55]: train_df
```

[55]:

	Basic_Demos-Enroll_Season	Basic_Demos-Age	Basic_Demos-Sex	CGAS-Season	CGAS-CGAS_Score	Physical-Season	Physical-BMI	Physical-Height	Physical-Weight	Physical-Diastolic_BP	...	PCIAT-PCIAT_18	PCIAT-PCIAT_19	PCI
0	3	5	0	4	51.0	3	16.877316	46.0	50.8	0.0	...	4.0	2.0	
1	2	9	0	0	0.0	3	14.035590	48.0	46.0	75.0	...	0.0	0.0	
2	2	10	1	3	71.0	3	16.648696	56.5	75.6	65.0	...	2.0	1.0	
3	4	9	0	3	71.0	2	18.292347	56.0	81.6	60.0	...	3.0	4.0	
5	1	13	1	4	50.0	2	22.279952	59.5	112.2	60.0	...	1.0	2.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3953	3	8	0	0	0.0	3	17.139810	52.5	67.2	60.0	...	2.0	2.0	
3954	2	7	1	0	0.0	2	13.927006	48.5	46.6	65.0	...	3.0	3.0	
3955	3	13	0	1	60.0	3	16.362460	59.5	82.4	71.0	...	1.0	1.0	
3957	3	11	0	1	68.0	4	21.441500	60.0	109.8	79.0	...	1.0	0.0	
3958	1	13	0	1	70.0	4	12.235895	70.7	87.0	59.0	...	1.0	1.0	

2736 rows × 72 columns

Figure 16: missing values imputation for train_df

```
[57]: test_df
```

[57]:

	Basic_Demos-Enroll_Season	Basic_Demos-Age	Basic_Demos-Sex	CGAS-Season	Physical-Season	Physical-BMI	Physical-Height	Physical-Weight	Physical-Diastolic_BP	Physical-HeartRate	...	FGC-FGC_SRR	FGC-FGC_SRR_Zone	FC
0	3	5	0	4	3	16.877316	46.00	50.8	0.0	0.0	...	6.0	0.0	
1	2	9	0	0	3	14.035590	48.00	46.0	75.0	70.0	...	11.0	1.0	
2	2	10	1	3	3	16.648696	56.50	75.6	65.0	94.0	...	10.0	1.0	
3	4	9	0	3	2	18.292347	56.00	81.6	60.0	97.0	...	7.0	0.0	
4	1	18	1	2	0	0.000000	0.00	0.0	0.0	0.0	...	0.0	0.0	
5	1	13	1	4	2	22.279952	59.50	112.2	60.0	73.0	...	11.0	1.0	
6	3	10	0	0	3	19.660760	55.00	84.6	123.0	83.0	...	11.0	1.0	
7	3	10	1	0	3	16.861286	59.25	84.2	71.0	90.0	...	0.0	0.0	
8	2	15	0	0	1	0.000000	0.00	0.0	0.0	0.0	...	0.0	0.0	
9	2	19	1	2	0	0.000000	0.00	0.0	0.0	0.0	...	0.0	0.0	
10	1	11	1	0	0	0.000000	0.00	0.0	0.0	0.0	...	0.0	0.0	
11	3	11	0	2	0	0.000000	0.00	0.0	0.0	0.0	...	0.0	0.0	
12	3	13	0	0	4	21.079065	57.75	100.0	63.0	79.0	...	9.5	1.0	
13	1	12	0	0	1	15.544111	60.00	79.6	57.0	71.0	...	9.0	1.0	
14	1	12	0	0	0	0.000000	0.00	0.0	0.0	0.0	...	0.0	0.0	
15	4	7	0	2	1	29.315775	54.00	121.6	80.0	75.0	...	15.0	1.0	
16	1	5	1	2	1	17.284504	44.00	47.6	61.0	76.0	...	10.0	1.0	
17	3	10	1	0	3	19.893157	55.00	85.6	0.0	81.0	...	0.0	0.0	
18	4	6	0	1	4	30.094649	37.50	60.2	61.0	91.0	...	4.0	0.0	
19	4	10	0	0	0	0.000000	0.00	0.0	0.0	0.0	...	0.0	0.0	

20 rows × 29 columns

Figure 17: missing values imputation for test_df

```
[59]: # check for null values
      train_df.isnull().any()
```

```
[59]: Basic_Demos-Enroll_Season                    False
      Basic_Demos-Age                              False
      Basic_Demos-Sex                              False
      CGAS-Season                                  False
      CGAS-CGAS_Score                              False
                                                   ...
      SDS-SDS_Total_Raw                            False
      SDS-SDS_Total_T                              False
      PreInt_EduHx-Season                          False
      PreInt_EduHx-computerinternet_hoursday       False
      sii                                          False
      Length: 72, dtype: bool
```

Figure 18: check for null values in train_df

```
[61]: # check for null values
      test_df.isnull().any()
```

```
[61]: Basic_Demos-Enroll_Season                    False
      Basic_Demos-Age                              False
      Basic_Demos-Sex                              False
      CGAS-Season                                  False
      Physical-Season                              False
      Physical-BMI                                 False
      Physical-Height                              False
      Physical-Weight                              False
      Physical-Diastolic_BP                        False
      Physical-HeartRate                           False
      Physical-Systolic_BP                         False
      Fitness_Endurance-Season                     False
      FGC-Season                                   False
      FGC-FGC_CU                                   False
      FGC-FGC_CU_Zone                              False
      FGC-FGC_PU                                   False
      FGC-FGC_PU_Zone                              False
      FGC-FGC_SRL                                  False
      FGC-FGC_SRL_Zone                             False
      FGC-FGC_SRR                                  False
      FGC-FGC_SRR_Zone                             False
      FGC-FGC_TL                                   False
      FGC-FGC_TL_Zone                              False
      BIA-Season                                   False
      PAQ_A-Season                                 False
      PAQ_C-Season                                 False
      SDS-Season                                   False
      PreInt_EduHx-Season                          False
      PreInt_EduHx-computerinternet_hoursday       False
      dtype: bool
```

Figure 19: check for null values in test_df

```
[63]: # check for presence of null values
      print(train_df.isnull().values.any())
      print(test_df.isnull().values.any())

      False
      False
```

Figure 20: verify absence of null values for train & test datasets

Predictor variables/features and response/target variable

There are a total of 82 predictor variables that comprises of data on demographics, internet use, children's global assessment scale, physical measures, fitnessgram vitals and treadmill, fitnessgram child, bio-electric impedance analysis, physical activity questionnaire, sleep disturbance scale, actigraphy and parent-child internet addiction test. They include all the variables listed below except 'sii'.

```
[16]: # get column names
      print(train_df.columns.values)

['id' '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' 'PCIAT-Season' 'PCIAT-PCIAT_01'
 'PCIAT-PCIAT_02' 'PCIAT-PCIAT_03' 'PCIAT-PCIAT_04' 'PCIAT-PCIAT_05'
 'PCIAT-PCIAT_06' 'PCIAT-PCIAT_07' 'PCIAT-PCIAT_08' 'PCIAT-PCIAT_09'
 'PCIAT-PCIAT_10' 'PCIAT-PCIAT_11' 'PCIAT-PCIAT_12' 'PCIAT-PCIAT_13'
 'PCIAT-PCIAT_14' 'PCIAT-PCIAT_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']
```

The response variable would be 'sii', which is derived from 'PCIAT-PCIAT_Total'. It is categorised as follows: 0: None, 1: Mild, 2: Moderate, 3: Severe.

Actual implemented code for predictor variables/features and the response/target variable and output generated

```
[39]: train_df['sii'].value_counts()

[39]: 0.0    1594
      1.0     730
      2.0     378
      3.0      34
      Name: sii, dtype: int64

[41]: train_df
```

[41]:

l- nt	Physical- Waist_Circumference	...	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
8	NaN	...	4.0	2.0	4.0	55.0	NaN	NaN	NaN	Fall	3.0	2.0
0	22.0	...	0.0	0.0	0.0	0.0	Fall	46.0	64.0	Summer	0.0	0.0
6	NaN	...	2.0	1.0	1.0	28.0	Fall	38.0	54.0	Summer	2.0	0.0
6	NaN	...	3.0	4.0	1.0	44.0	Summer	31.0	45.0	Winter	0.0	1.0
2	NaN	...	1.0	2.0	1.0	34.0	Summer	40.0	56.0	Spring	0.0	1.0
..	...	...	...	...	...	...	...	...	...	...	...	...
2	25.0	...	2.0	2.0	1.0	22.0	Fall	41.0	58.0	Fall	2.0	0.0
6	23.0	...	3.0	3.0	0.0	33.0	Summer	48.0	67.0	Summer	0.0	1.0
4	NaN	...	1.0	1.0	0.0	32.0	Winter	35.0	50.0	Fall	1.0	1.0
8	NaN	...	1.0	0.0	1.0	31.0	Winter	56.0	77.0	Fall	0.0	1.0
0	NaN	...	1.0	1.0	1.0	19.0	Spring	33.0	47.0	Spring	1.0	0.0

Figure 21: response variable – sii

```
[43]: train_df.shape

[43]: (2736, 81)

[45]: # response variable
      target = train_df['sii']
```

Figure 22: mark sii as target

```
[65]: # EDA - exploratory data analysis
      # data visualizations
```

```
[67]: # correlation matrix
      correlation_matrix = train_df.corr()

      # plot heatmap
      plt.figure(figsize=(30,30))
      sns.heatmap(correlation_matrix, annot=True, fmt='.1f', cmap='coolwarm', square=True)
      plt.title('Correlation Heatmap')
      plt.show()
```

Figure 23: view correlation between predictor & response variables

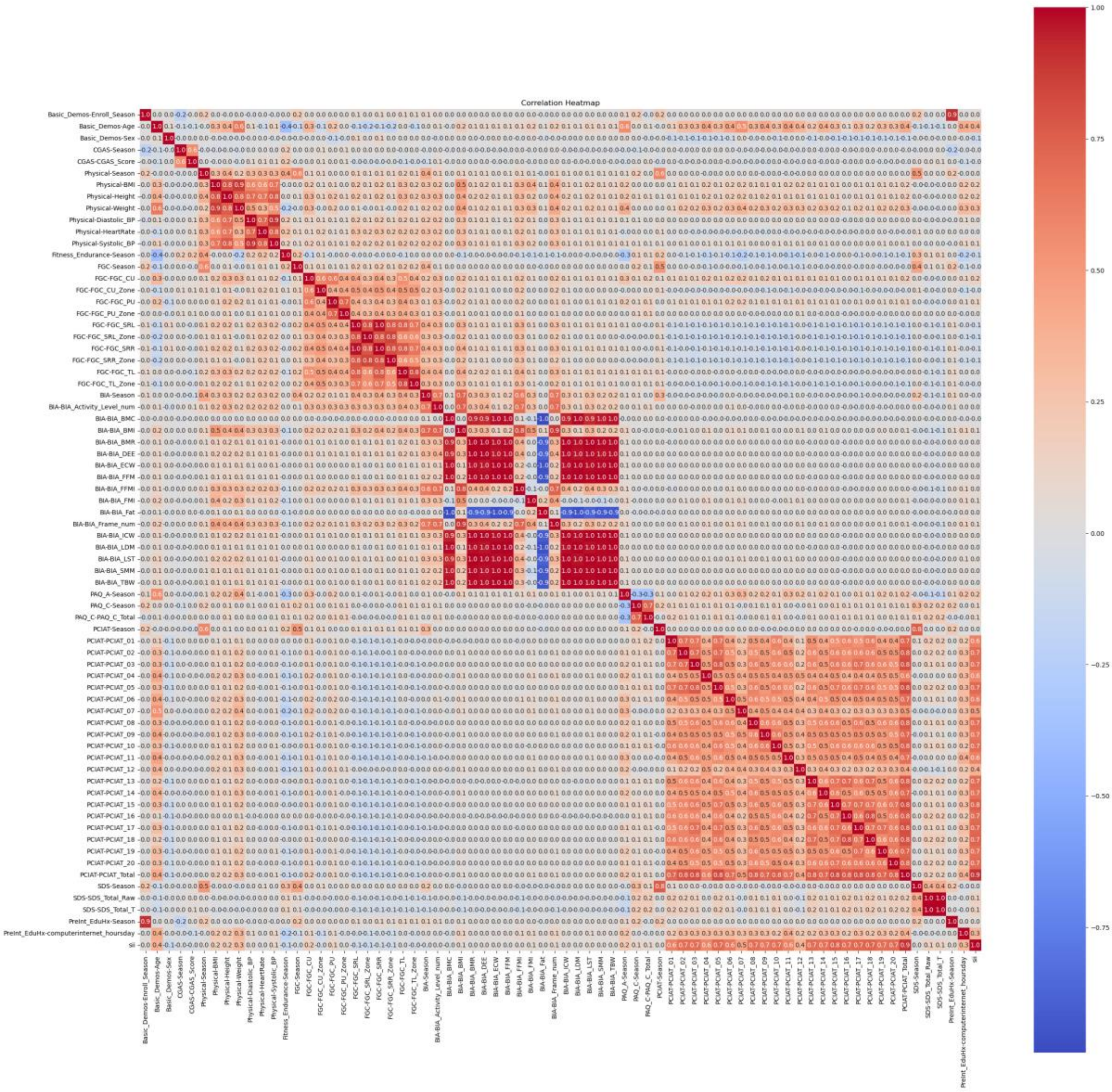

Figure 24: correlation matrix

Training and testing datasets

The training dataset comprises of 82 features and 3960 data objects, while the testing dataset comprises of 59 features and 20 data objects. We could further split our training dataset into 80:20 train-test validation sets to run our models so that we could train the validation set on the 82 features, before using it to test on the testing dataset that only has 59 features.

Actual implemented code relating to the training & testing data processing

```
[68]: # define X and y
X = train_df.iloc[:, :-1]
y = target

print(X.columns)
```

```
Index(['Basic_Demos-Enroll_Season', 'Basic_Demos-Age', 'Basic_Demos-Sex',
       'CGAS-Season', 'CGAS-CGAS_Score', 'Physical-Season', 'Physical-BMI',
       'Physical-Height', 'Physical-Weight', 'Physical-Diastolic_BP',
       'Physical-HeartRate', 'Physical-Systolic_BP',
       'Fitness_Endurance-Season', 'FGC-Season', 'FGC-FGC_CU',
       'FGC-FGC_CU_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_C-Season', 'PAQ_C-PAQ_C_Total', 'PCIAT-Season', 'PCIAT-PCIAT_01',
       'PCIAT-PCIAT_02', 'PCIAT-PCIAT_03', 'PCIAT-PCIAT_04', 'PCIAT-PCIAT_05',
       'PCIAT-PCIAT_06', 'PCIAT-PCIAT_07', 'PCIAT-PCIAT_08', 'PCIAT-PCIAT_09',
       'PCIAT-PCIAT_10', 'PCIAT-PCIAT_11', 'PCIAT-PCIAT_12', 'PCIAT-PCIAT_13',
       'PCIAT-PCIAT_14', 'PCIAT-PCIAT_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'],
      dtype='object')
```

Figure 25: define X and y for training dataset

```
[69]: # split data into training and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figure 26: further split training dataset into train & test sets for validation

ML techniques

To make a prediction on the level (target = [‘None’, ‘Mild’, ‘Moderate’, ‘Severe’]) of problematic internet use exhibited by children by analysing their physical activity through fitness data, we could consider the following ML techniques:

ML techniques	Description
Decision Tree	To fit a categorical variable decision tree such that we make categorical classifications.
Random Forest	To fit a random forest model for classification to predict the class label for the given input.
Gradient Boosting	To fit a gradient boosting model for classification, whereby an ensemble of decision trees is used to iteratively improve predictions for class labels.
K-Nearest Neighbours	To fit a KNN model for classification, producing outputs comprising of class label memberships.

Step-by-step code for each ML techniques

Decision Tree model

```
[83]: # decision tree model
dt_model = DecisionTreeClassifier(random_state=42)

# perform 10-fold cross validation
kfold = KFold(n_splits=10, shuffle=True, random_state=42)
scores = cross_val_score(dt_model, X, y, cv=kfold, scoring='accuracy')

print("Accuracy scores for each fold:", scores)
print("Mean accuracy score:", scores.mean())

# fit model
dt_model.fit(X_train, y_train)

Accuracy scores for each fold: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Mean accuracy score: 1.0

[83]: DecisionTreeClassifier(random_state=42)
```

Figure 27: fit decision tree model

```
[85]: y_pred_dt = dt_model.predict(X_test)
print(classification_report(y_test, y_pred_dt))

              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       336
         1.0       1.00      1.00      1.00       131
         2.0       1.00      1.00      1.00        72
         3.0       1.00      1.00      1.00         9

    accuracy                           1.00       548
   macro avg       1.00      1.00      1.00       548
weighted avg       1.00      1.00      1.00       548
```

Figure 28: predict based on decision tree model

```
[87]: print('Prediction Accuracy: ', accuracy_score(y_test, y_pred_dt))

Prediction Accuracy:  1.0

[89]: # precision
dt_precision = precision_score(y_test, y_pred_dt, average='macro')
print(f"Precision: {dt_precision}")

# recall
dt_recall = recall_score(y_test, y_pred_dt, average='macro')
print(f"Recall: {dt_recall}")

# f1 score
dt_f1 = f1_score(y_test, y_pred_dt, average='macro')
print(f"F1 Score: {dt_f1}")

# f2 score
dt_f2 = fbeta_score(y_test, y_pred_dt, beta=2, average='macro')
print(f"F2 Score: {dt_f2}")

Precision: 1.0
Recall: 1.0
F1 Score: 1.0
F2 Score: 1.0
```

Figure 29: generate performance metrics for decision tree model

```
[91]: # confusion matrix
dt_cm = confusion_matrix(y_test, y_pred_dt)

plt.figure(figsize=(8, 6))
sns.heatmap(dt_cm, annot=True, fmt="d", cmap="Blues", xticklabels=dt_model.classes_, yticklabels=dt_model.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Decision Tree Confusion Matrix')
plt.show()
```

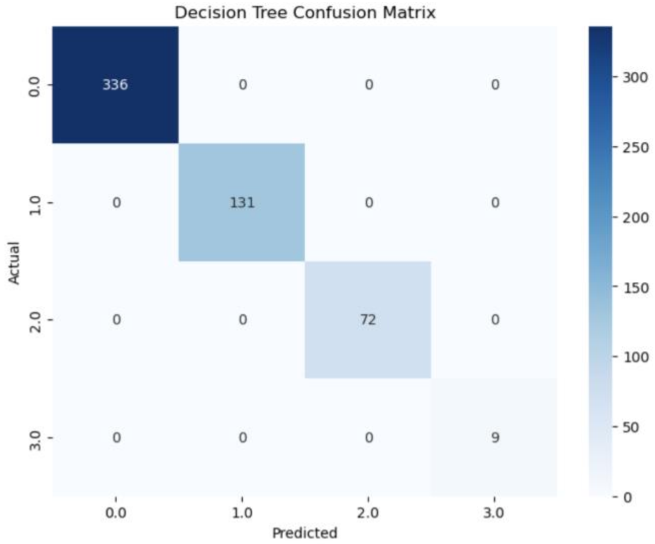

Figure 30: confusion matrix for decision tree

Random Forest model

```
[70]: # random forest model
rf_model = RandomForestClassifier(random_state=80)

# 10-fold cross validation
cv = KFold(n_splits=10, shuffle=True, random_state=42)
scores = cross_val_score(rf_model, X, y, cv=cv, scoring='accuracy')

print("Cross-validation scores:", scores)
print("Mean accuracy:", scores.mean())

# fit model
rf_model.fit(X_train, y_train)
Cross-validation scores: [0.99635036 1.         1.         0.99270073 0.99270073 0.99270073
 0.98901099 1.         0.99267399 1.        ]
Mean accuracy: 0.9956137536429507
[70]: RandomForestClassifier(random_state=80)
```

Figure 31: fit random forest model

```
[75]: y_pred_rf = rf_model.predict(X_test)
print(classification_report(y_test, y_pred_rf))
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       336
         1.0       0.98      1.00      0.99       131
         2.0       1.00      0.97      0.99        72
         3.0       1.00      1.00      1.00         9

    accuracy                           1.00       548
   macro avg       1.00      0.99      0.99       548
weighted avg       1.00      1.00      1.00       548
```

Figure 32: predict based on random forest model

[79]:
```python
# precision
rf_precision = precision_score(y_test, y_pred_rf, average='macro')
print(f"Precision: {rf_precision}")

# recall
rf_recall = recall_score(y_test, y_pred_rf, average='macro')
print(f"Recall: {rf_recall}")

# f1 score
rf_f1 = f1_score(y_test, y_pred_rf, average='macro')
print(f"F1 Score: {rf_f1}")

# f2 score
rf_f2 = fbeta_score(y_test, y_pred_rf, beta=2, average='macro')
print(f"F2 Score: {rf_f2}")
```

```
Precision: 0.9962406015037594
Recall: 0.9930555555555556
F1 Score: 0.9945849338454972
F2 Score: 0.9936523728136187
```

Figure 33: generate performance metrics for random forest model

[81]:
```python
# confusion matrix
rf_cm = confusion_matrix(y_test, y_pred_rf)

plt.figure(figsize=(8, 6))
sns.heatmap(rf_cm, annot=True, fmt="d", cmap="Blues", xticklabels=rf_model.classes_, yticklabels=rf_model.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Random Forest Confusion Matrix')
plt.show()
```

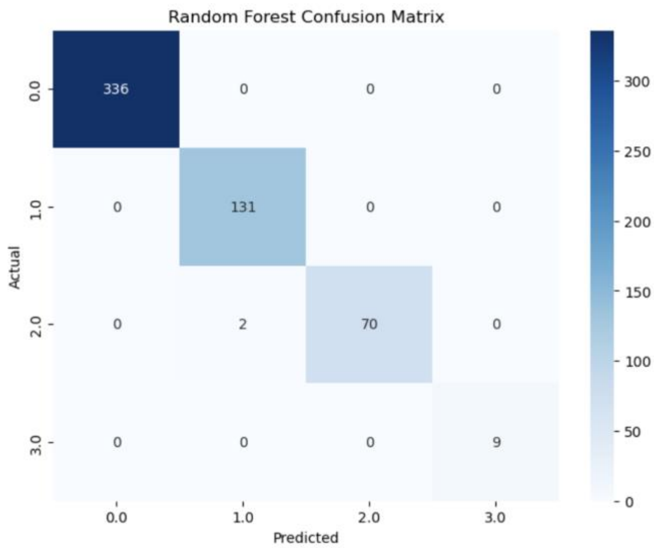

Figure 34: confusion matrix for random forest model

Gradient Boosting model

```
[93]: # gradient boosting model
gb_model = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)

# perform 10-fold cross validation
kf = KFold(n_splits=10, shuffle=True, random_state=42)
cv_scores = cross_val_score(gb_model, X, y, cv=kf, scoring='neg_mean_squared_error')

print("Cross-validation scores:", cv_scores)
print("Mean cross-validation score:", np.mean(cv_scores))
print("Standard deviation of cross-validation scores:", np.std(cv_scores))

# fit model
gb_model.fit(X_train, y_train)

Cross-validation scores: [-0. -0. -0. -0. -0. -0. -0. -0. -0. -0.]
Mean cross-validation score: 0.0
Standard deviation of cross-validation scores: 0.0

[93]: GradientBoostingClassifier(random_state=42)
```

Figure 35: fit gradient boosting model

```
[94]: y_pred_gb = dt_model.predict(X_test)
print(classification_report(y_test, y_pred_gb))

              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       336
         1.0       1.00      1.00      1.00       131
         2.0       1.00      1.00      1.00        72
         3.0       1.00      1.00      1.00         9

    accuracy                           1.00       548
   macro avg       1.00      1.00      1.00       548
weighted avg       1.00      1.00      1.00       548
```

Figure 36: predict based on gradient boosting model

```
[95]: print('Prediction Accuracy: ', accuracy_score(y_test, y_pred_gb))

Prediction Accuracy:  1.0

[99]: # precision
gb_precision = precision_score(y_test, y_pred_gb, average='macro')
print(f"Precision: {gb_precision}")

# recall
gb_recall = recall_score(y_test, y_pred_gb, average='macro')
print(f"Recall: {gb_recall}")

# f1 score
gb_f1 = f1_score(y_test, y_pred_gb, average='macro')
print(f"F1 Score: {gb_f1}")

# f2 score
gb_f2 = fbeta_score(y_test, y_pred_gb, beta=2, average='macro')
print(f"F2 Score: {gb_f2}")

Precision: 1.0
Recall: 1.0
F1 Score: 1.0
F2 Score: 1.0
```

Figure 37: performance metrics for gradient boosting model

```
[101]: # confusion matrix
gb_cm = confusion_matrix(y_test, y_pred_gb)

plt.figure(figsize=(8, 6))
sns.heatmap(gb_cm, annot=True, fmt="d", cmap="Blues", xticklabels=gb_model.classes_, yticklabels=gb_model.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Gradient Boosting Confusion Matrix')
plt.show()
```

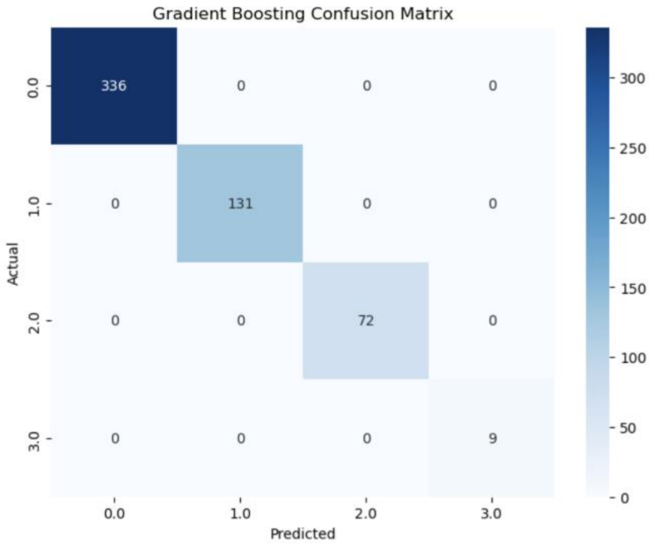

Figure 38: confusion matrix for gradient boosting model

K-Nearest Neighbours model

```
[103]: # k-nearest-neighbour model
knn_model = KNeighborsClassifier(n_neighbors=3)

# 10-fold cross validation
cv = KFold(n_splits=10, shuffle=True, random_state=42)
scores = cross_val_score(knn_model, X, y, cv=cv, scoring='accuracy')

print("Accuracy scores for each fold:", scores)
print("Mean accuracy:", np.mean(scores))

# fit model
knn_model.fit(X_train, y_train)

Accuracy scores for each fold: [0.67153285 0.71167883 0.71167883 0.72262774 0.63138686 0.64963504
 0.67765568 0.67399267 0.67032967 0.73260073]
Mean accuracy: 0.6853118900564157
[103]: KNeighborsClassifier(n_neighbors=3)
```

Figure 39: fit knn model

```
[105]: y_pred_knn = knn_model.predict(X_test)
print(classification_report(y_test, y_pred_knn))

              precision    recall  f1-score   support

         0.0       0.77      0.92      0.84       336
         1.0       0.50      0.40      0.45       131
         2.0       0.60      0.35      0.44        72
         3.0       0.00      0.00      0.00         9

    accuracy                           0.70       548
   macro avg       0.47      0.42      0.43       548
weighted avg       0.67      0.70      0.68       548
```

Figure 40: predict based on knn model

[107]:
```
print('Prediction Accuracy: ', accuracy_score(y_test, y_pred_knn))
```

```
Prediction Accuracy:  0.7043795620437956
```

[109]:
```
# precision
knn_precision = precision_score(y_test, y_pred_knn, average='macro')
print(f"Precision: {knn_precision}")

# recall
knn_recall = recall_score(y_test, y_pred_knn, average='macro')
print(f"Recall: {knn_recall}")

# f1 score
knn_f1 = f1_score(y_test, y_pred_knn, average='macro')
print(f"F1 Score: {knn_f1}")

# f2 score
knn_f2 = fbeta_score(y_test, y_pred_knn, beta=2, average='macro')
print(f"F2 Score: {knn_f2}")
```

```
Precision: 0.46562375565080927
Recall: 0.41711726039016117
F1 Score: 0.4305174701459531
F2 Score: 0.42057257990324637
```

Figure 41: performance metrics for knn model

[111]:
```
# confusion matrix
knn_cm = confusion_matrix(y_test, y_pred_knn)

plt.figure(figsize=(8, 6))
sns.heatmap(knn_cm, annot=True, fmt="d", cmap="Blues", xticklabels=knn_model.classes_, yticklabels=knn_model.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('K-Nearest Neighbour Confusion Matrix')
plt.show()
```

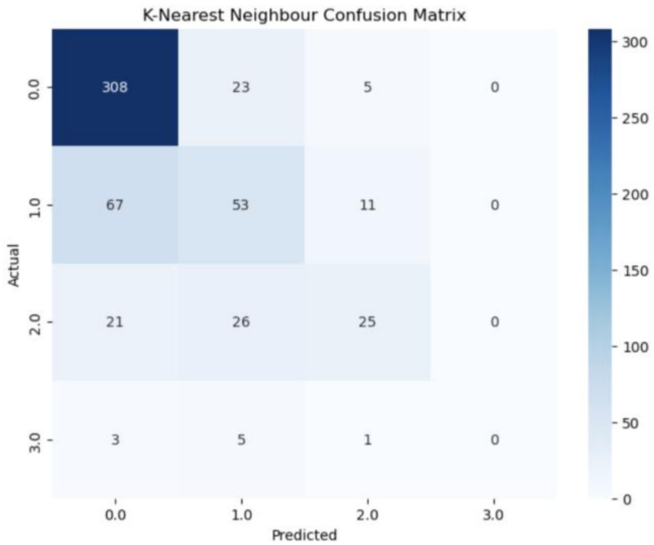

Figure 42: confusion matrix for knn model

Summary of work

Model Summary

In conclusion, Decision Trees and Gradient Boosting gives the best model performance based on the performance metrics for accuracy, precision, recall, f1-score and f2-score. However, they might risk overfitting. On the other hand, KNN has the worst model performance, with lowest scores for accuracy, precision, recall, f1-score and f2-score.

Therefore, the best model would be Random Forest with its high accuracy score of 0.99635, precision of 0.99635, recall of 0.99395, f1-score of 0.99458, f2-score of 0.99365, and low probability of overfitting.

Comparison across ML models

Model	Accuracy	Precision	Recall	F1	F2
RF	0.99635	0.99624	0.99305	0.99458	0.99365
DT	1.0	1.0	1.0	1.0	1.0
GB	1.0	1.0	1.0	1.0	1.0
KNN	0.70437	0.46562	0.41711	0.43051	0.42057