EC 9560 – DATA MINING

LAB 03

DARMILA.T

2020/E/027

SEMESTER 07

30th OCTOBER 2024

Data Preprocessing

```python
print(train_data[train_data['PCIAT-PCIAT_Total']<=30].sii.value_counts())
print(train_data[(train_data['PCIAT-PCIAT_Total']>30)
    & (train_data['PCIAT-PCIAT_Total']<50)].sii.value_counts())
print(train_data[(train_data['PCIAT-PCIAT_Total']>=50)
    & (train_data['PCIAT-PCIAT_Total']<80)].sii.value_counts())
print(train_data[train_data['PCIAT-PCIAT_Total']>=80].sii.value_counts())

train_data.sii.value_counts()
```

```
sii
0.0    1594
Name: count, dtype: int64
sii
1.0    730
Name: count, dtype: int64
sii
2.0    378
Name: count, dtype: int64
sii
3.0    34
Name: count, dtype: int64
```

sii	count
0.0	1594
1.0	730
2.0	378
3.0	34

dtype: int64

This sii values are correctly distributed within the PCIAT-PCIAT_Total score intervals.

```python
# Check if sii values are consistent within each PCIAT_Total score range
is_consistent = (
    (train_data[train_data['PCIAT-PCIAT_Total'] <= 30]['sii'] == 0).all() and
    (train_data[(train_data['PCIAT-PCIAT_Total'] > 30) & (train_data['PCIAT-PCIAT_Total'] < 50)]['sii'] == 1).all() and
    (train_data[(train_data['PCIAT-PCIAT_Total'] >= 50) & (train_data['PCIAT-PCIAT_Total'] < 80)]['sii'] == 2).all() and
    (train_data[train_data['PCIAT-PCIAT_Total'] >= 80]['sii'] == 3).all()
)

print("SII distribution is consistent with PCIAT_Total score ranges:", is_consistent)
```

```
SII distribution is consistent with PCIAT_Total score ranges: True
```

To ensure the `sii` target variable distribution aligns with the expected categories of `PCIAT-PCIAT_Total` score ranges, verify that each range of scores corresponds correctly to its designated `sii` value.

Define the Expected Ranges: Based on the original criteria:

- `PCIAT-PCIAT_Total` between 0–30 should have `sii = 0`.
- `PCIAT-PCIAT_Total` between 31–49 should have `sii = 1`.
- `PCIAT-PCIAT_Total` between 50–79 should have `sii = 2`.
- `PCIAT-PCIAT_Total` between 80–100 should have `sii = 3`.

Removes any rows in the train DataFrame where the sii column has missing (NaN) values.

```
[20]  train_data = train_data.dropna(subset='sii')
```

Here, I clean the data by ensuring all rows in train have valid values in the sii column, which is likely important if sii is used as a target variable or feature in further analysis.

Correlations

Given the large number of available features, I decided to perform feature selection to assess its impact on the model. Here, I selected features with the strongest correlation to the PCIAT total score, discarding those with weaker correlations.

```
# Selecting only numeric columns for correlation
numeric_data = train_data.select_dtypes(include=['float64', 'int64'])
# Calculate correlation with PCIAT-PCIAT_Total
corr = numeric_data.corr()['PCIAT-PCIAT_Total'].sort_values(ascending=False)
corr = pd.DataFrame(corr)
corr.style.background_gradient(cmap='YlOrRd')
```

	PCIAT-PCIAT_Total
PCIAT-PCIAT_Total	1.000000
sii	0.899681
Physical-Height	0.420765
Basic_Demos-Age	0.409559
PreInt_EduHx-computerinternet_hoursday	0.374124
Physical-Weight	0.353048
Physical-Waist_Circumference	0.327013
FGC-FGC_CU	0.287494
BIA-BIA_BMI	0.248060
Physical-BMI	0.240858
SDS-SDS_Total_T	0.237718
SDS-SDS_Total_Raw	0.234432
PAQ_A-Season	0.219292
FGC-FGC_PU	0.196006
BIA-BIA_Frame_num	0.193631
FGC-FGC_GSD	0.160472
Physical-Systolic_BP	0.147081
FGC-FGC_GSND	0.146813
FGC-FGC_TL	0.136696
PAQ_C-Season	0.115316
BIA-BIA_FFMI	0.109694
BIA-BIA_FMI	0.085863
BIA-BIA_Activity_Level_num	0.084548
BIA-BIA_LST	0.075623
Physical-Diastolic_BP	0.069321
FGC-FGC_PU_Zone	0.056973
BIA-BIA_DEE	0.053094
BIA-BIA_SMM	0.052912
BIA-BIA_ICW	0.052593
BIA-BIA_TBW	0.043015
BIA-BIA_Fat	0.038548
BIA-BIA_BMR	0.037009
BIA-BIA_FFM	0.037009
BIA-BIA_ECW	0.035568
PreInt_EduHx-Season	0.033788
Basic_Demos-Enroll_Season	0.029793
BIA-BIA_LDM	0.025885
SDS-Season	0.025112
Physical-Season	0.021411
FGC-FGC_GSD_Zone	0.006861
Fitness_Endurance-Time_Sec	-0.000373
CGAS-Season	-0.003344
FGC-FGC_CU_Zone	-0.004454
BIA-BIA_BMC	-0.008870
FGC-FGC_GSND_Zone	-0.009525
PAQ_C-PAQ_C_Total	-0.021943
PAQ_A-PAQ_A_Total	-0.026854
FGC-Season	-0.030890
FGC-FGC_TL_Zone	-0.037214
Physical-HeartRate	-0.037594
BIA-Season	-0.040922
Fitness_Endurance-Max_Stage	-0.041720
Fitness_Endurance-Time_Mins	-0.052376
CGAS-CGAS_Score	-0.070542
FGC-FGC_SRR	-0.077836
FGC-FGC_SRL	-0.091221
Basic_Demos-Sex	-0.093648
Fitness_Endurance-Season	-0.097788
FGC-FGC_SRR_Zone	-0.109682
FGC-FGC_SRL_Zone	-0.148850

```python
selection = corr[(corr['PCIAT-PCIAT_Total']>.1) | (corr['PCIAT-PCIAT_Total']<-.1)]
selection = [val for val in selection.index]
selection.remove('PCIAT-PCIAT_Total')
selection.remove('sii')
selection.remove('Physical-BMI')
selection.remove('SDS-SDS_Total_Raw')
```

```python
[25] selection
```

```
['Physical-Height',
 'Basic_Demos-Age',
 'PreInt_EduHx-computerinternet_hoursday',
 'Physical-Weight',
 'Physical-Waist_Circumference',
 'FGC-FGC_CU',
 'BIA-BIA_BMI',
 'SDS-SDS_Total_T',
 'PAQ_A-Season',
 'FGC-FGC_PU',
 'BIA-BIA_Frame_num',
 'FGC-FGC_GSD',
 'Physical-Systolic_BP',
 'FGC-FGC_GSND',
 'FGC-FGC_TL',
 'PAQ_C-Season',
 'BIA-BIA_FFMI',
 'FGC-FGC_SRR_Zone',
 'FGC-FGC_SRL_Zone']
```

Here, I select a subset of features that have a moderate correlation (either positive or negative) with PCIAT-PCIAT_Total, while excluding certain specified columns. The goal is to narrow down the list of features to those most relevant for analysis or modeling, based on their correlation with PCIAT-PCIAT_Total, and to exclude target or potentially redundant features.

Find the missing values in the train_data

```python
null = train_data.isna().sum().sort_values(ascending = False).head(46)
null = pd.DataFrame(null)
null = null.rename(columns= {0:'Missing'})
null.style.background_gradient(cmap='YlOrRd')
```

	Missing
PAQ_A-PAQ_A_Total	2373
Physical-Waist_Circumference	2253
Fitness_Endurance-Time_Sec	2008
Fitness_Endurance-Time_Mins	2008
Fitness_Endurance-Max_Stage	2005
FGC-FGC_GSD_Zone	1872
FGC-FGC_GSND_Zone	1872
FGC-FGC_GSD	1865
FGC-FGC_GSND	1864
PAQ_C-PAQ_C_Total	1296
BIA-BIA_Activity_Level_num	923
BIA-BIA_BMC	923
BIA-BIA_BMI	923
BIA-BIA_TBW	923
BIA-BIA_DEE	923
BIA-BIA_ECW	923
BIA-BIA_FFM	923
BIA-BIA_FFMI	923
BIA-BIA_FMI	923
BIA-BIA_Fat	923
BIA-BIA_Frame_num	923
BIA-BIA_ICW	923
BIA-BIA_LDM	923
BIA-BIA_LST	923
BIA-BIA_SMM	923
BIA-BIA_BMR	923
FGC-FGC_PU_Zone	861
FGC-FGC_SRL_Zone	859
FGC-FGC_SRR_Zone	857
FGC-FGC_CU_Zone	852
FGC-FGC_TL_Zone	851
FGC-FGC_PU	827
FGC-FGC_SRL	825
FGC-FGC_SRR	823
FGC-FGC_CU	817
FGC-FGC_TL	817
CGAS-CGAS_Score	394
Physical-Systolic_BP	258
Physical-Diastolic_BP	258
Physical-HeartRate	250
SDS-SDS_Total_T	211
Physical-BMI	209
SDS-SDS_Total_Raw	209
Physical-Height	206
Physical-Weight	164
PreInt_EduHx-computerinternet_hoursday	82

```
[32]  half_missing = [val for val in train_data.columns[train_data.isnull().sum()>len(train_data)/2]]
      half_missing
```

```
['Physical-Waist_Circumference',
 'Fitness_Endurance-Max_Stage',
 'Fitness_Endurance-Time_Mins',
 'Fitness_Endurance-Time_Sec',
 'FGC-FGC_GSND',
 'FGC-FGC_GSND_Zone',
 'FGC-FGC_GSD',
 'FGC-FGC_GSD_Zone',
 'PAQ_A-PAQ_A_Total']
```

Here, Identifies the column that more than one value is missing in the data frame.

Then, I check is any selected features have more than half value is missing, If I Found remove that in from the selection list features.

```
selection = [i for i in selection if i not in half_missing]
selection
```

```
['Physical-Height',
 'Basic_Demos-Age',
 'PreInt_EduHx-computerinternet_hoursday',
 'Physical-Weight',
 'FGC-FGC_CU',
 'BIA-BIA_BMI',
 'SDS-SDS_Total_T',
 'PAQ_A-Season',
 'FGC-FGC_PU',
 'BIA-BIA_Frame_num',
 'Physical-Systolic_BP',
 'FGC-FGC_TL',
 'PAQ_C-Season',
 'BIA-BIA_FFMI',
 'FGC-FGC_SRR_Zone',
 'FGC-FGC_SRL_Zone']
```

I have now identified 16 selected features based on two criteria:

a) their correlation with the target variable and

b) the presence of relatively few missing values.

Here, I want to create a model that makes accurate predictions by using only the most useful information while avoiding distractions from irrelevant data.

In the following section, I display the minimum and maximum values of the selected features in the training data.

```
describe = train_data[selection].describe().T
describe = describe[['min','max']].sort_index()
describe.style.background_gradient(cmap='YlOrRd')
```

	min	max
BIA-BIA_BMI	0.048267	48.375400
BIA-BIA_FFMI	7.864850	217.771000
BIA-BIA_Frame_num	1.000000	3.000000
Basic_Demos-Age	5.000000	22.000000
FGC-FGC_CU	0.000000	115.000000
FGC-FGC_PU	0.000000	51.000000
FGC-FGC_SRL_Zone	0.000000	1.000000
FGC-FGC_SRR_Zone	0.000000	1.000000
FGC-FGC_TL	0.000000	21.000000
PAQ_A-Season	0.000000	4.000000
PAQ_C-Season	0.000000	4.000000
Physical-Height	36.000000	78.500000
Physical-Systolic_BP	49.000000	203.000000
Physical-Weight	0.000000	315.000000
PreInt_EduHx-computerinternet_hoursday	0.000000	3.000000
SDS-SDS_Total_T	38.000000	100.000000

Now, I scaled scale the selected features in your train_data DataFrame using MinMaxScaler.

```
scaler = MinMaxScaler()
train_data_scaled = pd.DataFrame(scaler.fit_transform(train_data[selection]),
                                 columns=selection,
                                 index=train_data.index)
```

```
# Get descriptive statistics for the scaled data
scaled_describe = train_data_scaled.describe().T[['count', 'min', 'max']].sort_index()

# Display the result
scaled_describe.style.background_gradient(cmap='YlOrRd')
```

	count	min	max
BIA-BIA_BMI	1813.000000	0.000000	1.000000
BIA-BIA_FFMI	1813.000000	0.000000	1.000000
BIA-BIA_Frame_num	1813.000000	0.000000	1.000000
Basic_Demos-Age	2736.000000	0.000000	1.000000
FGC-FGC_CU	1919.000000	0.000000	1.000000
FGC-FGC_PU	1909.000000	0.000000	1.000000
FGC-FGC_SRL_Zone	1877.000000	0.000000	1.000000
FGC-FGC_SRR_Zone	1879.000000	0.000000	1.000000
FGC-FGC_TL	1919.000000	0.000000	1.000000
PAQ_A-Season	2736.000000	0.000000	1.000000
PAQ_C-Season	2736.000000	0.000000	1.000000
Physical-Height	2530.000000	0.000000	1.000000
Physical-Systolic_BP	2478.000000	0.000000	1.000000
Physical-Weight	2572.000000	0.000000	1.000000
PreInt_EduHx-computerinternet_hoursday	2654.000000	0.000000	1.000000
SDS-SDS_Total_T	2525.000000	0.000000	1.000000

Split the dataset

```
[35]
X = train_data[selection]
test = test_data[selection]
y = train_data.sii
```