Research about AI analysis in education for students with disabilities using data mining techniques for handling the missing data

---

Name: Aser Mohamed Ali Mahmoud

ID: 222102487

Link for the dataset:https://view.officeapps.live.com/op/view.aspx?src=https%3A%2F%2Fraw.githubusercontent.com%2FBOECHANGHORN%2FDataScience%2Frefs%2Fheads%2Fmain%2FDatabase-on-education-for-children-with-disabilities_2021.xlsx&wdOrigin=BROWSELINK

UNICEF Global database on education for children with disabilities

Last update: December 2021. Version 1.0

Three ways AI can help these students with disabilities:-

"**ACCESSIBILITY in TESTING**": The quality of synthetic speech is becoming more natural and improving rapidly. Educational Testing Service (ETS) improved the user experience for students with disabilities by reducing the turnaround time for producing alternate format materials and providing a more natural and clear text-to-speech voice for these students."

"**CONTENT DESCRIPTIONS**": Turnaround time is significant when producing things like text descriptions or a complex set of test questions for students who are legally blind or have low vision. AI techniques could be used to automatically describe images. AI-based systems could also be used to do a "first pass" at describing content. "

"**WEBPAGE INTERACTIONS**": AI-based tools can also be used to help with interactions by people who are unable to see content. Tools like provide ways of interacting with content through a spoken dialogue model"

Description of the dataset:-

"This database presents four indicators (described in the next section) for students with and without functional difficulty:

> **Foundational learning skills** (reading and numeracy for 7 to 14 year olds) :Foundational reading and numeracy skills are presented in separate sheets

---

The total indicator values as well as disaggregation by sex and urban location are also provided. This database is calculated using data from the five to seventeen questionnaire. It is important to note the value of the ""total"" presented here and the survey findings report may differ due to the different weighting scheme of the questionnaires estimated using the household questionnaire. However, the choice was made to make this information available despite the discrepancy to allow for comparison of the education for children with disabilities compared to those without disabilities and also against the population of all five to seventeen year olds.

Indicator definition:-

Foundational learning skills Foundational learning skills (SDG4.1.1.a) – Percentage of children achieving minimum proficiency in (i) reading and (ii) numeracy. If the child succeeds in 1) word recognition, 2) literal questions, and 3) inferential questions, s/he is considered to have foundational reading skills. If the child succeeds in 1) number reading, 2) number discrimination, 3) addition, and 4) pattern recognition, s/he is considered to have foundational numeracy skills.

Unicef Dataset about Education for students with disabilities 2021

```
#Importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import warnings
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import statistics
import scipy.stats as stats
from sklearn.metrics import mean_squared_error, mean_squared_log_error, mean_absolute_error, r2_score
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
import mlxtend
from mlxtend.frequent_patterns import apriori, association_rules, fpgrowth
from mlxtend.preprocessing import TransactionEncoder
from sklearn.cluster import DBSCAN
from sklearn.cluster import KMeans, AgglomerativeClustering, AffinityPropagation
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage
import skfuzzy as fuzz
from sklearn.mixture import GaussianMixture
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen d
  and should_run_async(code)
```

```
# Loading the csv file
file_path = '/content/Unicef-dataset-on-education-for-students-with-disabilities_2021.csv'
data = pd.read_csv(file_path)
```

```
data
```

	Countries and areas	ISO Code	Region	Sub-region	Development regions	Indicator	Category	Total	Unnamed: 8	Unnamed: 9	Children without functional difficulties	Unnamed: 11	Unnamed: 12	Children with functional difficulties	Unnamed: 14	Unnamed: 15	Data source	Time period
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Point estimate	Upper limit	Lower limit	Point estimate	Upper limit	Lower limit	Point estimate	Upper limit	Lower limit	NaN	NaN
1	Bangladesh	BGD	SA	SA	Least Developed	Foundational reading skill	Total	48.8	49.5	48.2	50.0	50.7	49.3	35.7	38.1	33.4	MICS6	2019
2	Bangladesh	BGD	SA	SA	Least Developed	Foundational reading skill	Male	45.1	46.0	44.1	46.4	47.4	45.4	31.3	34.4	28.3	MICS6	2019
3	Bangladesh	BGD	SA	SA	Least Developed	Foundational reading skill	Female	52.4	53.3	51.5	53.3	54.3	52.4	40.7	44.2	37.2	MICS6	2019
4	Bangladesh	BGD	SA	SA	Least Developed	Foundational reading skill	Urban	55.8	57.3	54.3	56.7	58.2	55.2	42.6	48.6	36.7	MICS6	2019
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
159	Zimbabwe	ZWE	SSA	ESA	Less Developed	Foundational reading skill	Urban	67.6	70.2	65.0	68.3	71.0	65.6	60.5	70.5	50.4	MICS6	2019
160	Zimbabwe	ZWE	SSA	ESA	Less Developed	Foundational reading skill	Rural	36.8	38.5	35.0	38.3	40.2	36.5	23.5	28.3	18.7	MICS6	2019
161	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
162	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
163	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

164 rows × 18 columns

```
print(data['Children with functional difficulties'])
```

```
0       NaN
1      35.7
2      31.3
3      40.7
4      42.6
       ...
159    60.5
160    23.5
161     NaN
162     NaN
163     NaN
Name: Children with functional difficulties, Length: 164, dtype: float64
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen d
```

```
  and should_run_async(code)
```

```
print(data['Children without functional difficulties'])
```

```
0       NaN
1      50.0
2      46.4
3      53.3
4      56.7
       ...
159    68.3
160    38.3
161     NaN
162     NaN
163     NaN
Name: Children without functional difficulties, Length: 164, dtype: float64
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen d
  and should_run_async(code)
```

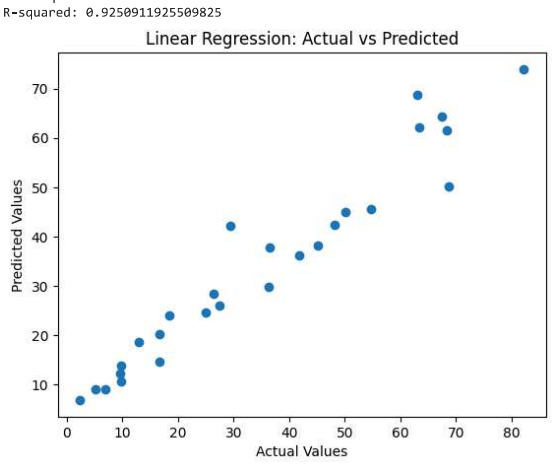
Converting the selected columns to numeric (float) data

```
data['Children with functional difficulties'] = pd.to_numeric(
    data['Children with functional difficulties'], errors='coerce'
)

data['Children without functional difficulties'] = pd.to_numeric(
    data['Children without functional difficulties'], errors='coerce'
)
```

Calculating the Mean Squared Error (MSE) & R-squared and plotting the Linear Regression plot

```
selected_column = ['Children with functional difficulties']
target_column = ['Children without functional difficulties']

X = df[selected_column]
y = df[target_column]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the Linear Regression model
lin_reg = LinearRegression()

# Fit the model on the training data
lin_reg.fit(X_train, y_train)

# Make predictions on the test set
y_pred = lin_reg.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")

# Optional: Plot the predicted vs actual values for a visual evaluation
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Linear Regression: Actual vs Predicted')
plt.show()
```

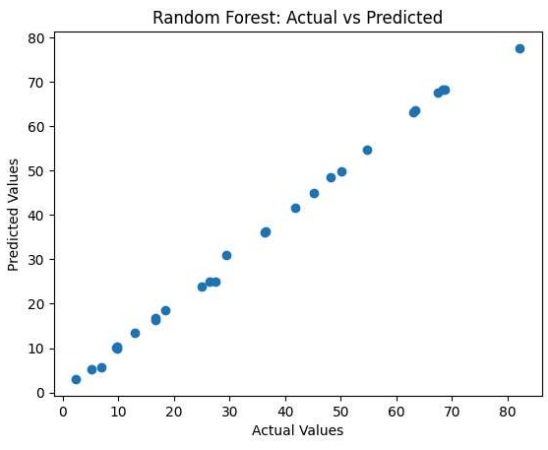
```
Mean Squared Error: 40.027797672854426
R-squared: 0.9250911925509825
```



Calculating the Random Forest Mean Squared Error and plotting the Random Forest plot

```
selected_column = ['Children with functional difficulties']
target_column = ['Children without functional difficulties']

X = df[selected_columns]
y = df[target_column]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the Random Forest Regressor model
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)

# Fit the model on the training data
rf_regressor.fit(X_train, y_train)

# Make predictions on the test set
y_pred_rf = rf_regressor.predict(X_test)

# Evaluate the model
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

print(f"Random Forest Mean Squared Error: {mse_rf}")
print(f"Random Forest R-squared: {r2_rf}")

# Optional: Plot the predicted vs actual values for a visual evaluation
plt.scatter(y_test, y_pred_rf)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Random Forest: Actual vs Predicted')
plt.show()
```

```
Random Forest Mean Squared Error: 1.3909606666666792
Random Forest R-squared: 0.9973969288642839
```



Calculating the Mean, Median, Mode, Standard deviation, Quartiles and IQR before handling the missing data

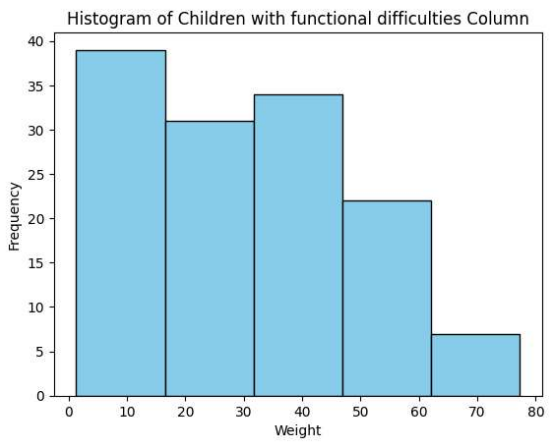
```
mean=data['Children with functional difficulties'].mean()
print('Mean= ',mean)


from scipy.stats import trim_mean
trimmed_mean = trim_mean(data['Children with functional difficulties'], 0.1)
print('Trimmed Mean= ',trimmed_mean)



median=data['Children with functional difficulties'].median()
print('Median', median)



mode=statistics.mode(data['Children with functional difficulties'])
print('Mode=', mode)



std=data['Children with functional difficulties'].std()
print('Standard Deviation= ',std)



Q1 = np.percentile(data['Children with functional difficulties'], 25)
Q2 = np.median(data['Children with functional difficulties'])
Q3 = np.percentile(data['Children with functional difficulties'], 75)
IQR = Q3 - Q1
print('Q1= ',Q1)
print('Q2 (Median)= ',Q2)
print('Q3= ',Q3)
print('IQR= ',IQR)
```

```
Mean=  30.56842105263158
Trimmed Mean=  nan
Median 30.7
Mode= 7.8
Standard Deviation=  18.631788445285025
Q1=  nan
Q2 (Median)=  nan
Q3=  nan
IQR=  nan
```

Plotting the Histogram

```
plt.hist(data['Children with functional difficulties'], bins=5, color='skyblue', edgecolor='black')
plt.xlabel('Weight')
plt.ylabel('Frequency')
plt.title('Histogram of Children with functional difficulties Column')
plt.show()
```

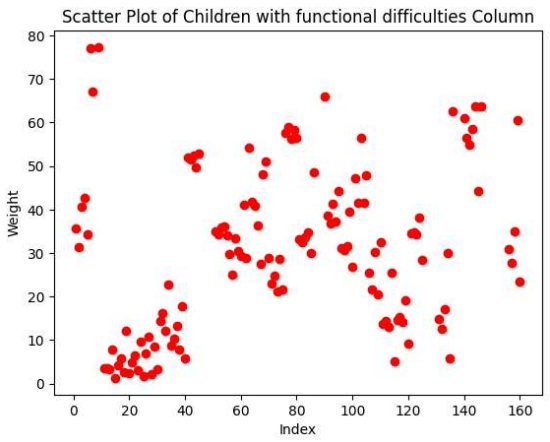
```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen d
  and should_run_async(code)
```



Plotting the Scatter Plot

```
plt.scatter(data.index, data['Children with functional difficulties'], color='red', marker='o')
plt.xlabel('Index')
plt.ylabel('Weight')
plt.title('Scatter Plot of Children with functional difficulties Column')
plt.show()
```

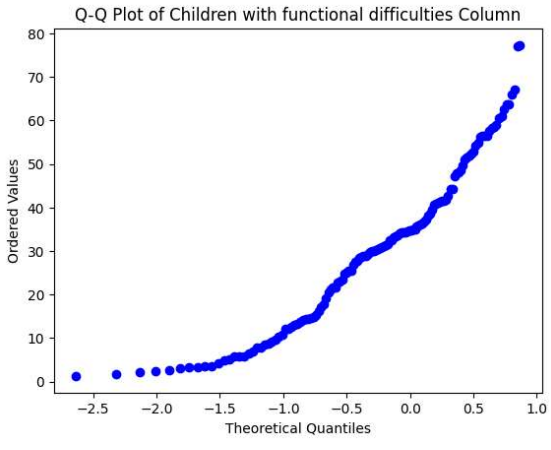
```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen d
  and should_run_async(code)
```



Plotting the Q-Q Plot

```
stats.probplot(data['Children with functional difficulties'], dist="norm", plot=plt)
plt.title('Q-Q Plot of Children with functional difficulties Column')
plt.ylabel('Ordered Values')
plt.xlabel('Theoretical Quantiles')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen d
  and should_run_async(code)
```



Calculating the Cosine Similarity before handling the missing data

```
from numpy.linalg import norm
A= data['Children with functional difficulties'].values
B= data['Children without functional difficulties'].values
cosine = np.dot(A, B) / (norm(A) * norm(B))
print("Cosine Similarity:", cosine)
```

```
Cosine Similarity: nan
```

Showing the missing values

```
print("Missing values in each column:")
print(data.isnull().sum())
```

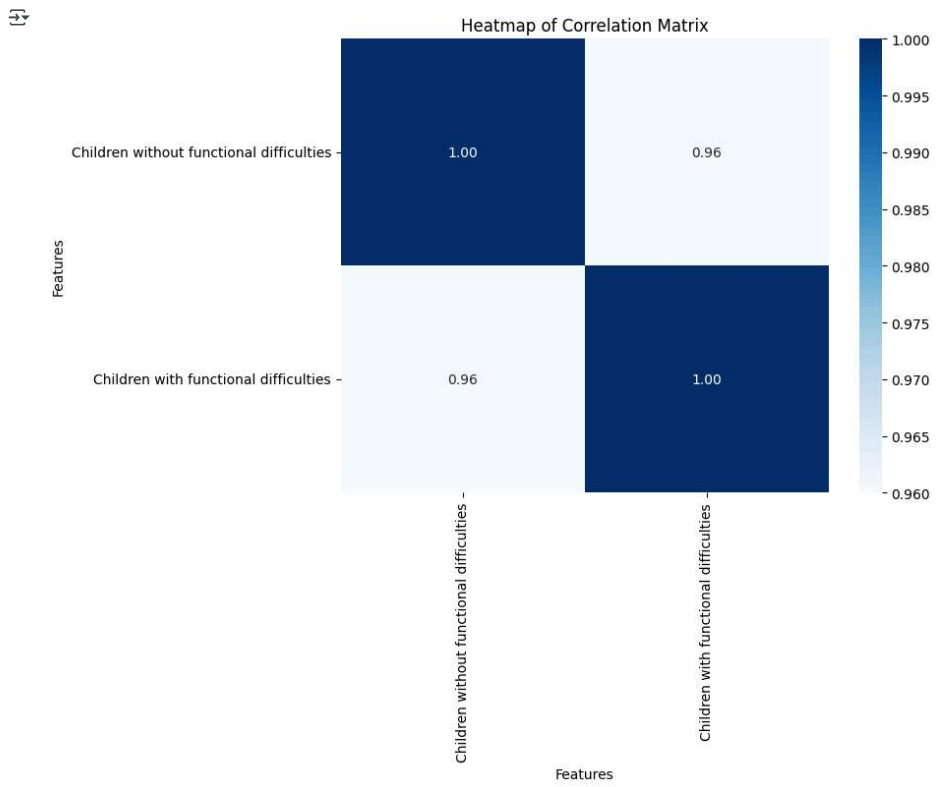
```
Missing values in each column:
Countries and areas                          4
ISO Code                                    19
Region                                       4
Sub-region                                   4
Development regions                          9
Indicator                                    4
Category                                     4
Total                                        8
Unnamed: 8                                   8
Unnamed: 9                                   8
Children without functional difficulties     9
Unnamed: 11                                  8
Unnamed: 12                                  8
Children with functional difficulties       31
Unnamed: 14                                 30
Unnamed: 15                                 30
Data source                                  4
Time period                                  4
dtype: int64
```

```
numeric_data = data.select_dtypes(include=[np.number])
correlation_matrix = numeric_data.corr()
print(correlation_matrix)
```

```
                                    Children without functional difficulties  \
Children without functional difficulties                                1.000000
Children with functional difficulties                                   0.959998

                                    Children with functional difficulties
Children without functional difficulties                             0.959998
Children with functional difficulties                                1.000000
```

Plotting the Heatmap of Correlation Matrix

```
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='Blues', fmt='.2f')
plt.xlabel('Features')
plt.ylabel('Features')
plt.title('Heatmap of Correlation Matrix')
plt.show()
```



Applying the Linear interpolation method to handle the missing values

```
data_interpolated = data.interpolate (method='linear')
print("Data after linear interpolation:")
print(data_interpolated)
```

```
Data after linear interpolation:
    Countries and areas ISO Code Region Sub-region Development regions  \
0                   NaN      NaN    NaN        NaN                 NaN
1            Bangladesh      BGD     SA         SA     Least Developed
2            Bangladesh      BGD     SA         SA     Least Developed
3            Bangladesh      BGD     SA         SA     Least Developed
4            Bangladesh      BGD     SA         SA     Least Developed
..                  ...      ...    ...        ...                 ...
159            Zimbabwe      ZWE    SSA        ESA      Less Developed
160            Zimbabwe      ZWE    SSA        ESA      Less Developed
161                 NaN      NaN    NaN        NaN                 NaN
162                 NaN      NaN    NaN        NaN                 NaN
163                 NaN      NaN    NaN        NaN                 NaN

                      Indicator Category          Total   Unnamed: 8  \
0                           NaN      NaN  Point estimate  Upper limit
1    Foundational reading skill    Total            48.8        49.5
2    Foundational reading skill     Male            45.1        46.0
3    Foundational reading skill   Female            52.4        53.3
4    Foundational reading skill    Urban            55.8        57.3
..                          ...      ...             ...         ...
159  Foundational reading skill    Urban            67.6        70.2
160  Foundational reading skill    Rural            36.8        38.5
161                         NaN      NaN             NaN         NaN
162                         NaN      NaN             NaN         NaN
163                         NaN      NaN             NaN         NaN

      Unnamed: 9  Children without functional difficulties  Unnamed: 11  \
0    Lower limit                                       NaN  Upper limit
1           48.2                                      50.0        50.7
2           44.1                                      46.4        47.4
3           51.5                                      53.3        54.3
4           54.3                                      56.7        58.2
..           ...                                       ...         ...
159         65.0                                      68.3        71.0
160         35.0                                      38.3        40.2
161          NaN                                      38.3         NaN
162          NaN                                      38.3         NaN
163          NaN                                      38.3         NaN

     Unnamed: 12  Children with functional difficulties  Unnamed: 14  \
0    Lower limit                                    NaN  Upper limit
1           49.3                                   35.7        38.1
2           45.4                                   31.3        34.4
3           52.4                                   40.7        44.2
4           55.2                                   42.6        48.6
..           ...                                    ...         ...
159         65.6                                   60.5        70.5
160         36.5                                   23.5        28.3
161          NaN                                   23.5         NaN
162          NaN                                   23.5         NaN
163          NaN                                   23.5         NaN

     Unnamed: 15 Data source Time period
0    Lower limit         NaN         NaN
1           33.4       MICS6        2019
2           28.3       MICS6        2019
```

```
3          37.2       MICS6       2019
```

Dropping rows with missing values

```
data_drop_na = data.dropna()
print("Data after dropping rows with missing values:")
print(data_drop_na)
```

```
Data after dropping rows with missing values:
     Countries and areas ISO Code Region Sub-region Development regions  \
1             Bangladesh      BGD     SA         SA     Least Developed
2             Bangladesh      BGD     SA         SA     Least Developed
3             Bangladesh      BGD     SA         SA     Least Developed
4             Bangladesh      BGD     SA         SA     Least Developed
5             Bangladesh      BGD     SA         SA     Least Developed
..                   ...      ...    ...        ...                 ...
156             Zimbabwe      ZWE    SSA        ESA      Less Developed
157             Zimbabwe      ZWE    SSA        ESA      Less Developed
158             Zimbabwe      ZWE    SSA        ESA      Less Developed
159             Zimbabwe      ZWE    SSA        ESA      Less Developed
160             Zimbabwe      ZWE    SSA        ESA      Less Developed

                        Indicator Category Total Unnamed: 8 Unnamed: 9  \
1    Foundational reading skill    Total  48.8       49.5       48.2
2    Foundational reading skill     Male  45.1       46.0       44.1
3    Foundational reading skill   Female  52.4       53.3       51.5
4    Foundational reading skill    Urban  55.8       57.3       54.3
5    Foundational reading skill    Rural  47.0       47.7       46.3
..                          ...      ...   ...        ...        ...
156  Foundational reading skill    Total  44.4       45.9       42.9
157  Foundational reading skill     Male  40.5       42.6       38.4
158  Foundational reading skill   Female  48.4       50.5       46.2
159  Foundational reading skill    Urban  67.6       70.2       65.0
160  Foundational reading skill    Rural  36.8       38.5       35.0

     Children without functional difficulties Unnamed: 11 Unnamed: 12  \
1                                        50.0        50.7        49.3
2                                        46.4        47.4        45.4
3                                        53.3        54.3        52.4
4                                        56.7        58.2        55.2
5                                        48.2        48.9        47.4
..                                        ...         ...         ...
156                                      45.9        47.4        44.3
157                                      42.1        44.3        39.9
158                                      49.7        52.0        47.5
159                                      68.3        71.0        65.6
160                                      38.3        40.2        36.5

     Children with functional difficulties Unnamed: 14 Unnamed: 15  \
1                                     35.7        38.1        33.4
2                                     31.3        34.4        28.3
3                                     40.7        44.2        37.2
4                                     42.6        48.6        36.7
5                                     34.4        36.9        31.9
..                                     ...         ...         ...
156                                   31.0        35.6        26.4
157                                   27.8        33.7        21.9
158                                   34.9        42.1        27.7
159                                   60.5        70.5        50.4
160                                   23.5        28.3        18.7

    Data source Time period
1         MICS6        2019
2         MICS6        2019
3         MICS6        2019
4         MICS6        2019
```

Filling missing values with mean

```
numeric_data = data.select_dtypes (include=[np.number])
data_fill_mean = data.copy()
data_fill_mean[numeric_data.columns] = numeric_data.fillna(numeric_data.mean())
print("Data after filling missing values with mean:")
print(data_fill_mean)
```

```
Data after filling missing values with mean:
     Countries and areas ISO Code Region Sub-region Development regions  \
0                    NaN      NaN    NaN        NaN                 NaN
1             Bangladesh      BGD     SA         SA     Least Developed
2             Bangladesh      BGD     SA         SA     Least Developed
3             Bangladesh      BGD     SA         SA     Least Developed
4             Bangladesh      BGD     SA         SA     Least Developed
..                   ...      ...    ...        ...                 ...
159             Zimbabwe      ZWE    SSA        ESA      Less Developed
160             Zimbabwe      ZWE    SSA        ESA      Less Developed
161                  NaN      NaN    NaN        NaN                 NaN
162                  NaN      NaN    NaN        NaN                 NaN
163                  NaN      NaN    NaN        NaN                 NaN

                        Indicator Category           Total   Unnamed: 8  \
0                              NaN      NaN  Point estimate  Upper limit
1    Foundational reading skill    Total            48.8         49.5
2    Foundational reading skill     Male            45.1         46.0
3    Foundational reading skill   Female            52.4         53.3
4    Foundational reading skill    Urban            55.8         57.3
..                          ...      ...             ...          ...
159  Foundational reading skill    Urban            67.6         70.2
160  Foundational reading skill    Rural            36.8         38.5
161                          NaN      NaN             NaN          NaN
162                          NaN      NaN             NaN          NaN
163                          NaN      NaN             NaN          NaN

      Unnamed: 9  Children without functional difficulties  Unnamed: 11  \
0    Lower limit                                43.008387  Upper limit
1           48.2                                50.000000         50.7
2           44.1                                46.400000         47.4
3           51.5                                53.300000         54.3
4           54.3                                56.700000         58.2
..           ...                                      ...          ...
159         65.0                                68.300000         71.0
160         35.0                                38.300000         40.2
161          NaN                                43.008387          NaN
162          NaN                                43.008387          NaN
163          NaN                                43.008387          NaN

     Unnamed: 12  Children with functional difficulties  Unnamed: 14  \
0    Lower limit                             30.568421  Upper limit
1           49.3                             35.700000         38.1
2           45.4                             31.300000         34.4
3           52.4                             40.700000         44.2
4           55.2                             42.600000         48.6
..           ...                                   ...          ...
159         65.6                             60.500000         70.5
160         36.5                             23.500000         28.3
161          NaN                             30.568421          NaN
162          NaN                             30.568421          NaN
163          NaN                             30.568421          NaN

     Unnamed: 15 Data source Time period
0    Lower limit         NaN         NaN
1           33.4       MICS6        2019
2           28.3       MICS6        2019
3           37.2       MICS6        2019
```

```
data
```

	Countries and areas	ISO Code	Region	Sub-region	Development regions	Indicator	Category	Total	Unnamed: 8	Unnamed: 9	Children without functional difficulties	Unnamed: 11	Unnamed: 12	Children with functional difficulties	Unnamed: 14	Unnamed: 15	Data source	Time period
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Point estimate	Upper limit	Lower limit	NaN	Upper limit	Lower limit	NaN	Upper limit	Lower limit	NaN	NaN
1	Bangladesh	BGD	SA	SA	Least Developed	Foundational reading skill	Total	48.8	49.5	48.2	50.0	50.7	49.3	35.7	38.1	33.4	MICS6	2019
2	Bangladesh	BGD	SA	SA	Least Developed	Foundational reading skill	Male	45.1	46.0	44.1	46.4	47.4	45.4	31.3	34.4	28.3	MICS6	2019
3	Bangladesh	BGD	SA	SA	Least Developed	Foundational reading skill	Female	52.4	53.3	51.5	53.3	54.3	52.4	40.7	44.2	37.2	MICS6	2019
4	Bangladesh	BGD	SA	SA	Least Developed	Foundational reading skill	Urban	55.8	57.3	54.3	56.7	58.2	55.2	42.6	48.6	36.7	MICS6	2019
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
159	Zimbabwe	ZWE	SSA	ESA	Less Developed	Foundational reading skill	Urban	67.6	70.2	65.0	68.3	71.0	65.6	60.5	70.5	50.4	MICS6	2019
160	Zimbabwe	ZWE	SSA	ESA	Less Developed	Foundational reading skill	Rural	36.8	38.5	35.0	38.3	40.2	36.5	23.5	28.3	18.7	MICS6	2019
161	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
162	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
163	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

164 rows × 18 columns

Calculating the Quartiles (Q1, Q2 median, Q3) and IQR after handling the missing data

```
print(  data.describe())
print(  data.info())
print('Q1= ', Q1)
print('Q2 (Median)= ',Q2)
print('Q3= ',Q3)
print('IQR= ',IQR)
```

```
       Children without functional difficulties  \
count                                155.000000
mean                                  43.008387
std                                   23.047235
min                                    2.400000
25%                                   21.150000
50%                                   44.400000
75%                                   60.450000
max                                   84.700000

       Children with functional difficulties
count                             133.000000
mean                               30.568421
std                                18.631788
min                                 1.300000
25%                                14.300000
50%                                30.700000
75%                                41.600000
max                                77.300000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 164 entries, 0 to 163
Data columns (total 18 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Countries and areas                   160 non-null    object
 1   ISO Code                              145 non-null    object
 2   Region                                160 non-null    object
 3   Sub-region                            160 non-null    object
 4   Development regions                   155 non-null    object
 5   Indicator                             160 non-null    object
 6   Category                              160 non-null    object
 7   Total                                 156 non-null    object
 8   Unnamed: 8                            156 non-null    object
 9   Unnamed: 9                            156 non-null    object
 10  Children without functional difficulties  155 non-null    float64
 11  Unnamed: 11                           156 non-null    object
 12  Unnamed: 12                           156 non-null    object
 13  Children with functional difficulties     133 non-null    float64
 14  Unnamed: 14                           134 non-null    object
 15  Unnamed: 15                           134 non-null    object
 16  Data source                           160 non-null    object
 17  Time period                           160 non-null    object
dtypes: float64(2), object(16)
memory usage: 23.2+ KB
None
Q1=  Children with functional difficulties       14.3
Children without functional difficulties    18.4
Name: 0.25, dtype: float64
Q2 (Median)=  nan
Q3=  Children with functional difficulties       41.6
Children without functional difficulties    55.0
Name: 0.75, dtype: float64
IQR=  Children with functional difficulties       27.3
Children without functional difficulties    36.6
dtype: float64
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen
  and should_run_async(code)
```

Calculating the Cosine Similarity after handling the missing data

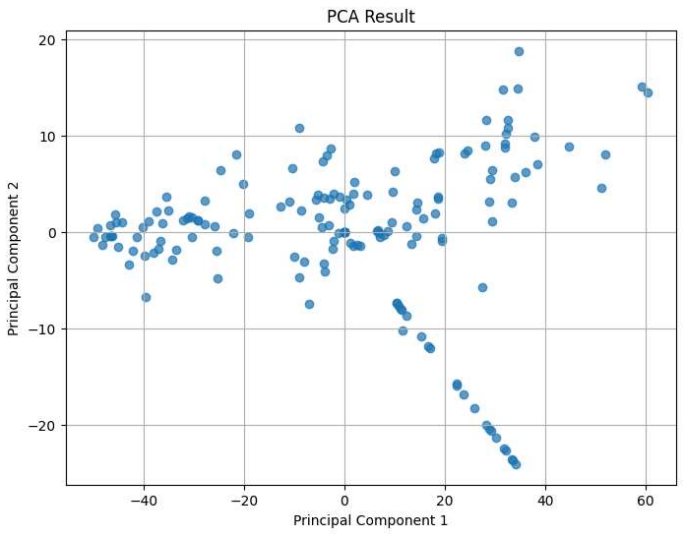
```
from numpy.linalg import norm
data_fill_mean.replace(0, np.nan, inplace=True)
numeric_columns= data_fill_mean.select_dtypes(include=[np.number]).columns
data_fill_mean[numeric_columns] = data_fill_mean[numeric_columns].fillna(data_fill_mean[numeric_columns].mean())

A =data_fill_mean['Children with functional difficulties'].values
B =data_fill_mean['Children without functional difficulties'].values
cosine = np.dot(A, B) / (norm(A) * norm(B))
print("Cosine Similarity:", cosine)
```

```
Cosine Similarity: 0.9607105985343067
```

Principle Component Analysis (PCA)

```
from sklearn.decomposition import PCA
selected_columns= data_fill_mean[['Children with functional difficulties', 'Children without functional difficulties']]
pca = PCA(n_components=2)
principal_components = pca.fit_transform(selected_columns)
principal_data = pd.DataFrame(data=principal_components, columns=['Principal Component 1', 'Principal Component 2'])
print("Principal Components:")
print(principal_data)
plt.figure(figsize=(8, 6))
plt.scatter(principal_data[ 'Principal Component 1'], principal_data['Principal Component 2'], alpha=0.7)
plt.title('PCA Result')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.grid()
plt.show()
```

```
Principal Components:
     Principal Component 1  Principal Component 2
0                 0.000000           5.967449e-15
1                 8.671112           1.660580e-01
2                 3.193615          -1.356122e+00
3                14.249094           2.351490e+00
4                18.122749           1.945538e+00
..                     ...                    ...
159              37.917015           9.892679e+00
160              -7.920839          -3.064604e+00
161               0.000000           5.967449e-15
162               0.000000           5.967449e-15
163               0.000000           5.967449e-15

[164 rows x 2 columns]
```




```
pip install mlxtend
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen d
  and should_run_async(code)
Requirement already satisfied: mlxtend in /usr/local/lib/python3.10/dist-packages (0.23.3)
Requirement already satisfied: scipy>=1.2.1 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (1.13.1)
Requirement already satisfied: numpy>=1.16.2 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (1.26.4)
Requirement already satisfied: pandas>=0.24.2 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (2.2.2)
Requirement already satisfied: scikit-learn>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (1.6.0)
Requirement already satisfied: matplotlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (3.10.0)
Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (1.4.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (4.55.3)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.2->mlxtend) (2024.2)
```

```
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.2->mlxtend) (2024.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.3.1->mlxtend) (3.5.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=3.0.0->mlxtend) (1.17.0)
```

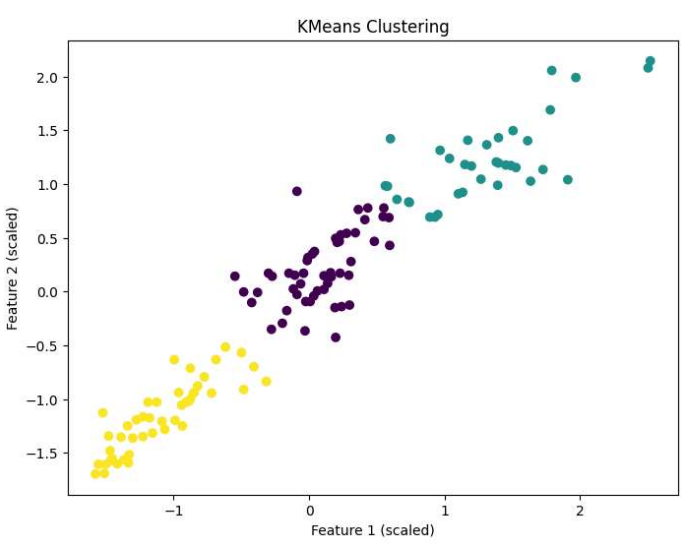
Apriori & FP-Growth

```
selected_columns = ['Children with functional difficulties', 'Children without functional difficulties']
df = df[selected_columns]

# Handle non-numeric data (convert to numeric and handle NaN values)
df = df.apply(pd.to_numeric, errors='coerce').dropna()

# Scale the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)

# Threshold the scaled data to convert to binary (0/1)
df_bin = pd.DataFrame(scaled_data, columns=df.columns)
df_bin[df_bin > 0] = 1  # Convert to binary (1 if value > 0, else 0)
df_bin[df_bin <= 0] = 0

# Apply Apriori to find frequent itemsets with minimum support of 0.6
frequent_itemsets_apriori = apriori(df_bin, min_support=0.6, use_colnames=True)

# Display the frequent itemsets from Apriori
print("Frequent Itemsets from Apriori:")
print(frequent_itemsets_apriori)

# Apply FP-Growth to find frequent itemsets with minimum support of 0.6
frequent_itemsets_fpgrowth = fpgrowth(df_bin, min_support=0.6, use_colnames=True)

# Display the frequent itemsets from FP-Growth
print("\nFrequent Itemsets from FP-Growth:")
print(frequent_itemsets_fpgrowth)
```

```
Frequent Itemsets from Apriori:
Empty DataFrame
Columns: [support, itemsets]
Index: []

Frequent Itemsets from FP-Growth:
Empty DataFrame
Columns: [support, itemsets]
Index: []
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen d
  and should_run_async(code)
/usr/local/lib/python3.10/dist-packages/mlxtend/frequent_patterns/fpcommon.py:161: DeprecationWarning: DataFrames with non-bool types result in worse computationalperformance and their support might be discontinued in the future.Please use a DataFrame
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/mlxtend/frequent_patterns/fpcommon.py:161: DeprecationWarning: DataFrames with non-bool types result in worse computationalperformance and their support might be discontinued in the future.Please use a DataFrame
  warnings.warn(
```

KMeans Clustering

```
# Select relevant features for clustering (replace 'feature1', 'feature2' with your column names)
selected_columns = ['Children with functional difficulties', 'Children without functional difficulties']
df = data[selected_columns]

# Handle non-numeric data (convert to numeric and handle NaN values)
df = df.apply(pd.to_numeric, errors='coerce').dropna()

# Scale the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)

# Apply KMeans clustering
num_clusters = 3  # Choose the number of clusters
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
kmeans.fit(scaled_data)

# Add cluster labels to the original DataFrame
df['Cluster'] = kmeans.labels_

# Visualize the clusters (for 2D data)
plt.figure(figsize=(8, 6))
plt.scatter(scaled_data[:, 0], scaled_data[:, 1], c=kmeans.labels_, cmap='viridis')
plt.title('KMeans Clustering')
plt.xlabel('Feature 1 (scaled)')
plt.ylabel('Feature 2 (scaled)')
plt.show()

# Output the clustered data
print(df.head())
```




```
   Children with functional difficulties  \
1                                   35.7
2                                   31.3
3                                   40.7
4                                   42.6
5                                   34.4

   Children without functional difficulties  Cluster
1                                      50.0        0
2                                      46.4        0
3                                      53.3        0
4                                      56.7        1
5                                      48.2        0
```

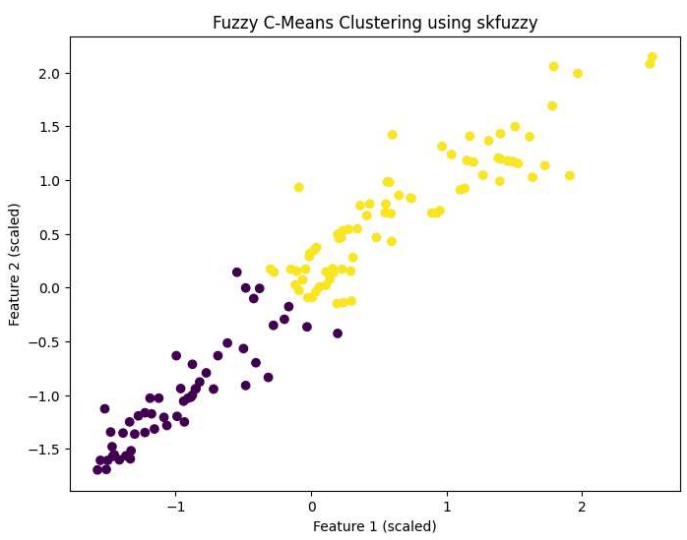
```
pip install scikit-fuzzy matplotlip
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen d
  and should_run_async(code)
Requirement already satisfied: scikit-fuzzy in /usr/local/lib/python3.10/dist-packages (0.5.0)
Requirement already satisfied: matplotlip in /usr/local/lib/python3.10/dist-packages (0.2)
```

Fuzzy C-Means Clustering

```
# Select relevant features for clustering
selected_columns = ['Children with functional difficulties', 'Children without functional difficulties']
df = df[selected_columns]

# Handle non-numeric data (convert to numeric and handle NaN values)
df = df.apply(pd.to_numeric, errors='coerce').dropna()

# Scale the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)

# Apply Fuzzy C-Means clustering using skfuzzy
num_clusters = 2  # We are assuming 2 clusters: one for children with difficulties, one for children without
cntr, u, u0, d, jm, p, fpc = fuzz.cmeans(scaled_data.T, num_clusters, 2, error=0.005, maxiter=1000, init=None)

# The output u is the fuzzy membership matrix
# u will be a 2D matrix with rows as data points and columns as clusters
# The membership value indicates the degree of belonging to each cluster

# Predict the cluster (most likely cluster)
cluster_predictions = np.argmax(u, axis=0)

# Assign cluster labels based on fuzzy membership (adjust as needed)
```

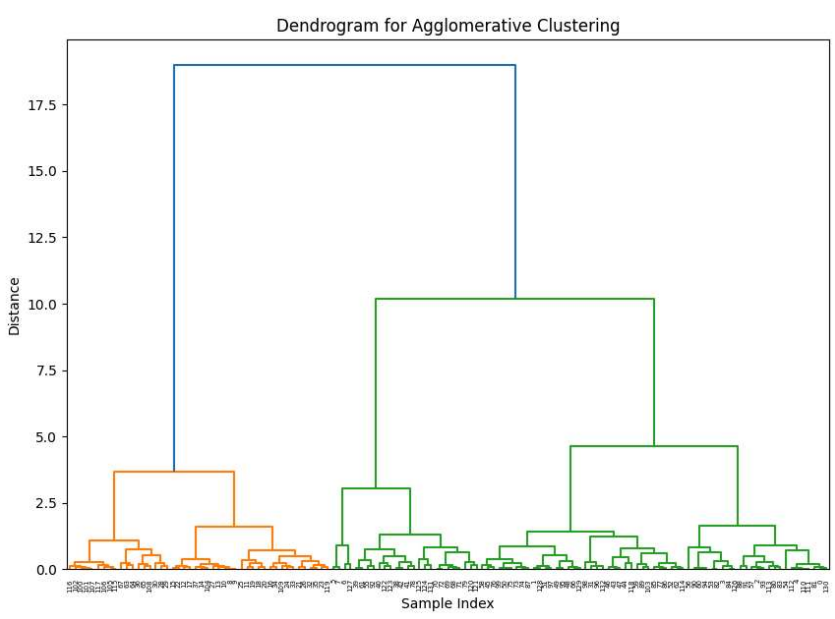
```
# Cluster 0: Children with functional difficulties, Cluster 1: Children without functional difficulties
cluster_0_mean_membership = np.mean(u[0, :])
cluster_1_mean_membership = np.mean(u[1, :])

if cluster_0_mean_membership > cluster_1_mean_membership:
    df['Cluster'] = ['Children with functional difficulties' if pred == 0 else 'Children without functional difficulties' for pred in cluster_predictions]
else:
    df['Cluster'] = ['Children with functional difficulties' if pred == 1 else 'Children without functional difficulties' for pred in cluster_predictions]

# Visualize the clusters (for 2D data)
plt.figure(figsize=(8, 6))
plt.scatter(scaled_data[:, 0], scaled_data[:, 1], c=cluster_predictions, cmap='viridis')
plt.title('Fuzzy C-Means Clustering using skfuzzy')
plt.xlabel('Feature 1 (scaled)')
plt.ylabel('Feature 2 (scaled)')
plt.show()

# Output the clustered data
print(df.head())
```




```
   Children with functional difficulties  \
1                                   35.7
2                                   31.3
3                                   40.7
4                                   42.6
5                                   34.4

   Children without functional difficulties  \
1                                      50.0
2                                      46.4
3                                      53.3
4                                      56.7
5                                      48.2

                               Cluster
1  Children with functional difficulties
2  Children with functional difficulties
3  Children with functional difficulties
4  Children with functional difficulties
5  Children with functional difficulties
```

Agglomerative Clustering & Plotting the Dendrogram

```
# Select relevant features for clustering
selected_columns = ['Children with functional difficulties', 'Children without functional difficulties']
df = df[selected_columns]

# Handle non-numeric data (convert to numeric and handle NaN values)
df = df.apply(pd.to_numeric, errors='coerce').dropna()

# Scale the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)

# Perform Agglomerative Clustering using Scipy
Z = linkage(scaled_data, method='ward')  # 'ward' minimizes the variance of the clusters being merged

# Plot the dendrogram
plt.figure(figsize=(10, 7))
dendrogram(Z)
plt.title('Dendrogram for Agglomerative Clustering')
plt.xlabel('Sample Index')
plt.ylabel('Distance')
plt.show()

# Apply AgglomerativeClustering (with 2 clusters, for example)
agg_clust = AgglomerativeClustering(n_clusters=2, linkage='ward')
df['Cluster'] = agg_clust.fit_predict(scaled_data)

# Output the clustered data
print(df.head())
```



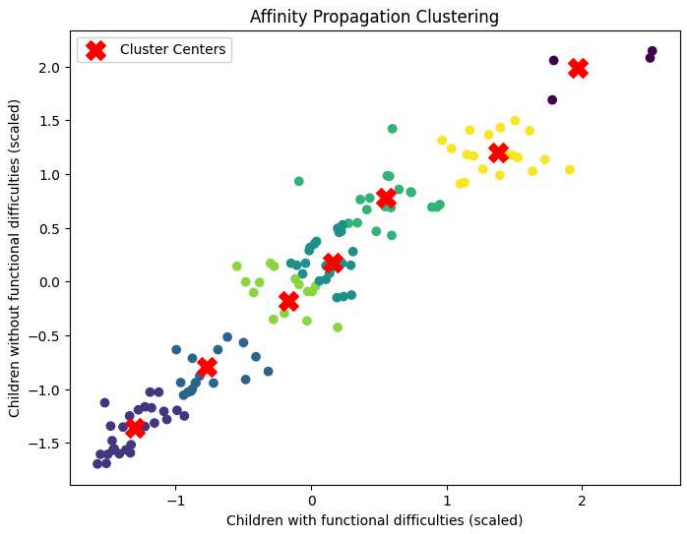

```
   Children with functional difficulties  \
1                                   35.7
2                                   31.3
3                                   40.7
4                                   42.6
5                                   34.4

   Children without functional difficulties  Cluster
1                                      50.0        0
2                                      46.4        0
3                                      53.3        0
4                                      56.7        0
5                                      48.2        0
```

Affinity Propagation Clustering

```
selected_columns = ['Children with functional difficulties', 'Children without functional difficulties']
df = df[selected_columns]

# Handle non-numeric data (convert to numeric and handle NaN values)
df = df.apply(pd.to_numeric, errors='coerce').dropna()

# Scale the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)

# Apply Affinity Propagation
affinity_model = AffinityPropagation(random_state=42)
affinity_model.fit(scaled_data)
```

```
# Get the cluster centers and labels
cluster_centers_indices = affinity_model.cluster_centers_indices_
labels = affinity_model.labels_

# Add cluster labels to the original DataFrame
df['Cluster'] = labels

# Print out the number of clusters
print(f"Number of clusters: {len(np.unique(labels))}")

# Visualize the clusters (if the data is 2D)
plt.figure(figsize=(8, 6))
plt.scatter(scaled_data[:, 0], scaled_data[:, 1], c=labels, cmap='viridis')
plt.scatter(scaled_data[cluster_centers_indices, 0], scaled_data[cluster_centers_indices, 1],
            s=200, marker='X', c='red', label='Cluster Centers')
plt.title('Affinity Propagation Clustering')
plt.xlabel('Children with functional difficulties (scaled)')
plt.ylabel('Children without functional difficulties (scaled)')
plt.legend()
plt.show()

# Output the clustered data
print(df.head())
```

```
Number of clusters: 7
```

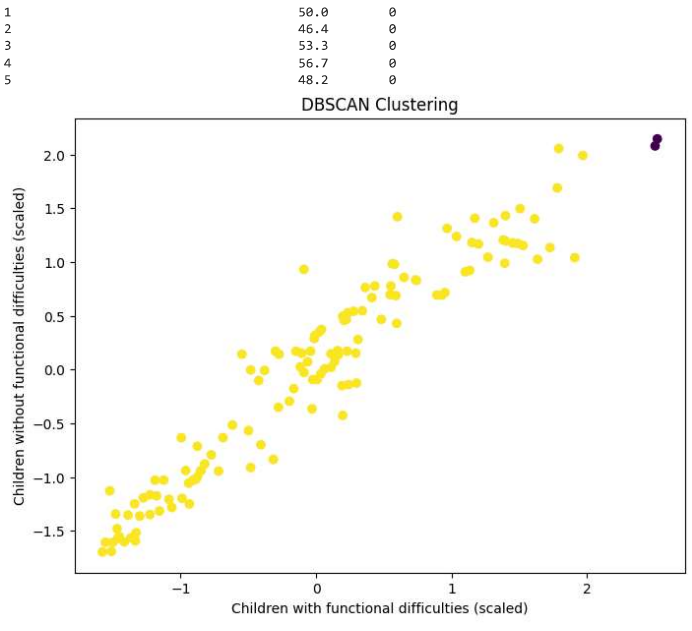



```
   Children with functional difficulties  \
1                                   35.7
2                                   31.3
3                                   40.7
4                                   42.6
5                                   34.4

   Children without functional difficulties  Cluster
1                                      50.0        4
2                                      46.4        3
3                                      53.3        4
4                                      56.7        4
5                                      48.2        3
```

Density Based Clustering (DBSCAN)

```
selected_columns = ['Children with functional difficulties', 'Children without functional difficulties']
df = df[selected_columns]

# Handle non-numeric data (convert to numeric and handle NaN values)
df = df.apply(pd.to_numeric, errors='coerce').dropna()

# Scale the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)

# Apply DBSCAN clustering
dbscan = DBSCAN(eps=0.5, min_samples=5)  # You can adjust eps and min_samples as needed
dbscan_labels = dbscan.fit_predict(scaled_data)

# Add the DBSCAN cluster labels to the DataFrame
df['Cluster'] = dbscan_labels

# Print the resulting clusters
print(df.head())

# Visualize the DBSCAN clustering (only if the data is 2D)
plt.figure(figsize=(8, 6))
plt.scatter(scaled_data[:, 0], scaled_data[:, 1], c=dbscan_labels, cmap='viridis')
plt.title('DBSCAN Clustering')
plt.xlabel('Children with functional difficulties (scaled)')
plt.ylabel('Children without functional difficulties (scaled)')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen d
  and should_run_async(code)
   Children with functional difficulties  \
1                                   35.7
2                                   31.3
3                                   40.7
4                                   42.6
5                                   34.4

   Children without functional difficulties  Cluster
1                                      50.0        0
2                                      46.4        0
3                                      53.3        0
4                                      56.7        0
5                                      48.2        0
```

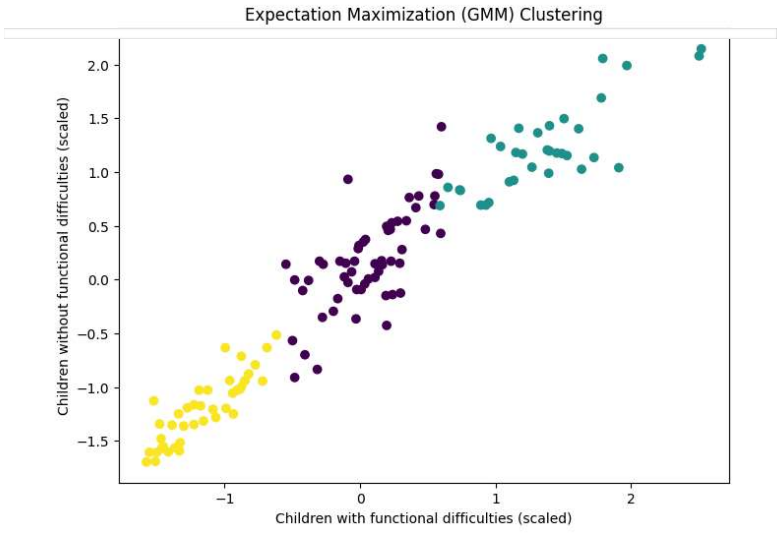


Expectation Maximization using Gaussian Mixture (GMM)

```
selected_columns = ['Children with functional difficulties', 'Children without functional difficulties'
df = df[selected_columns]

# Handle non-numeric data (convert to numeric and handle NaN values)
df = df.apply(pd.to_numeric, errors='coerce').dropna()

# Scale the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)

# Apply Gaussian Mixture Model (GMM) for clustering (Expectation Maximization)
gmm = GaussianMixture(n_components=3, random_state=42)  # You can change the number of components
gmm_labels = gmm.fit_predict(scaled_data)

# Add the GMM cluster labels to the DataFrame
df['Cluster'] = gmm_labels
```

```
# Print the resulting clusters
print(df.head())

# Visualize the GMM clustering (only if the data is 2D)
plt.figure(figsize=(8, 6))
plt.scatter(scaled_data[:, 0], scaled_data[:, 1], c=gmm_labels, cmap='viridis')
plt.title('Expectation Maximization (GMM) Clustering')
plt.xlabel('Children with functional difficulties (scaled)')
plt.ylabel('Children without functional difficulties (scaled)')
plt.show()
```

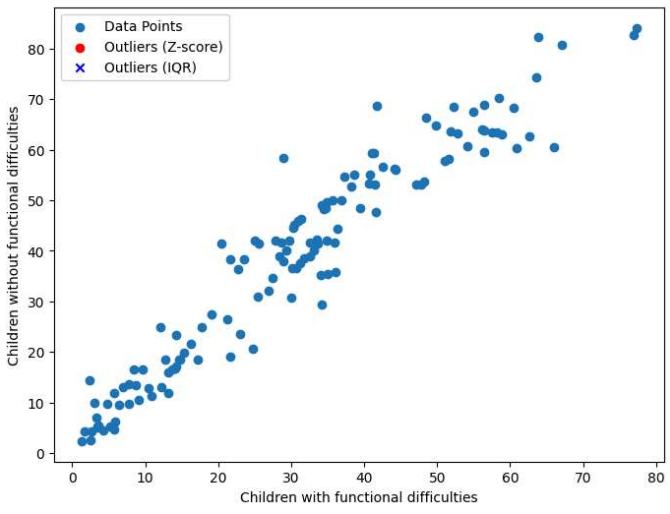
```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen d
  and should_run_async(code)
   Children with functional difficulties  \
1                                   35.7
2                                   31.3
3                                   40.7
4                                   42.6
5                                   34.4

   Children without functional difficulties  Cluster
1                                      50.0        0
2                                      46.4        0
3                                      53.3        0
4                                      56.7        1
5                                      48.2        0
```



Showing the Outliers

```
selected_columns = ['Children with functional difficulties', 'Children without functional difficulties']
df = df[selected_columns]

# Handle non-numeric data (convert to numeric and handle NaN values)
df = df.apply(pd.to_numeric, errors='coerce').dropna()

# Z-score Method to identify outliers
z_scores = np.abs(stats.zscore(df))
outliers_z = (z_scores > 3).all(axis=1)  # Outliers where z-score > 3

# IQR Method to identify outliers
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
outliers_iqr = ((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).any(axis=1)

# Combine both outlier detection results
df['Outlier_Z'] = outliers_z
df['Outlier_IQR'] = outliers_iqr

# Print the rows with outliers (Z-score and IQR)
outliers_df = df[(outliers_z | outliers_iqr)]
print("Outliers detected (Z-score or IQR method):")
print(outliers_df)

# Visualize the outliers (using a scatter plot for 2D data)
plt.figure(figsize=(8, 6))
plt.scatter(df['Children with functional difficulties'], df['Children without functional difficulties'], label='Data Points')

# Mark the outliers
plt.scatter(df[outliers_z]['Children with functional difficulties'],
            df[outliers_z]['Children without functional difficulties'],
            color='red', label='Outliers (Z-score)')

plt.scatter(df[outliers_iqr]['Children with functional difficulties'],
            df[outliers_iqr]['Children without functional difficulties'],
            color='blue', label='Outliers (IQR)', marker='x')

plt.title('Outlier Detection')
plt.xlabel('Children with functional difficulties')
plt.ylabel('Children without functional difficulties')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen d
  and should_run_async(code)
Outliers detected (Z-score or IQR method):
Empty DataFrame
Columns: [Children with functional difficulties, Children without functional difficulties, Outlier_Z, Outlier_IQR]
Index: []
```



Thank You

With the supervission of Dr. Mohamed Abdelaziz