Digital Egypt Pioneers Initiative-DEPI
AI & Data Science - Data Scientist-Round Two
Healthcare Predictive Analytics Project
ONL2_AIS4_S1 Data Scientist Group

Eng: Ahmed Azab

Identifying Malnutrition Risk Prediction

[Milestone One: Data Collection, Exploration, and Preprocessing]

Problem Statement: *Malnutrition Risk Prediction*:

The primary healthcare problem this project addresses is malnutrition among children, a pressing global health challenge. Malnutrition is not only a cause of mortality but also leads to long-term physical and cognitive impairments. Predicting malnutrition trends and identifying the contributing risk factors can support governments, NGOs, and healthcare providers in targeting interventions efficiently, improving policy-making, allocating resources more effectively, and enhancing child health outcomes.
This project uses machine learning techniques to build a predictive model that can classify whether a child is at risk of malnutrition, based on various health and demographic indicators.

Dataset Overview:

"Child Malnutrition UNICEF Dataset" was selected. It provides comprehensive global and regional data on key indicators of child malnutrition, including stunting, wasting, underweight, and overweight prevalence. The data is sourced from the UNICEF, WHO, and World Bank joint estimates, ensuring both reliability and relevance for healthcare modeling.

Digital Egypt Pioneers Initiative-DEPI
AI & Data Science - Data Scientist-Round Two
Healthcare Predictive Analytics Project
ONL2_AIS4_S1 Data Scientist Group

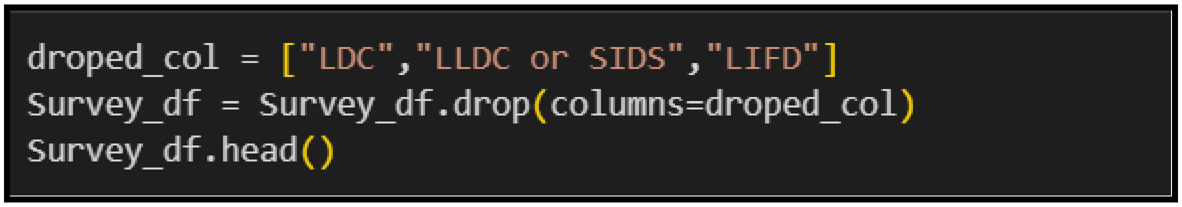
The dataset contains multiple Excel sheets; however, the first sheet was used in this project as it includes the core malnutrition metrics aggregated by country and year. These metrics represent a consolidated view of child health conditions and serve as strong predictive indicators for healthcare modeling tasks.

Code Structures:

The data preprocessing phase began by loading the **"Survey Estimates"** sheet from the UNICEF malnutrition dataset, which was then converted to a CSV format for ease of handling and analysis. An initial exploration was conducted to understand the structure and quality of the dataset. Summary statistics were generated to observe the distribution and range of numerical variables. A check for missing values revealed several null entries, highlighting the need for appropriate imputation or exclusion strategies in the cleaning phase. Data types for each column were inspected to ensure correct format handling during modeling, and a search for duplicate records was performed, revealing the presence of redundant rows. These preprocessing steps laid a solid foundation for the subsequent stages of analysis and model development by ensuring data consistency, reliability, and readiness for machine learning workflows.

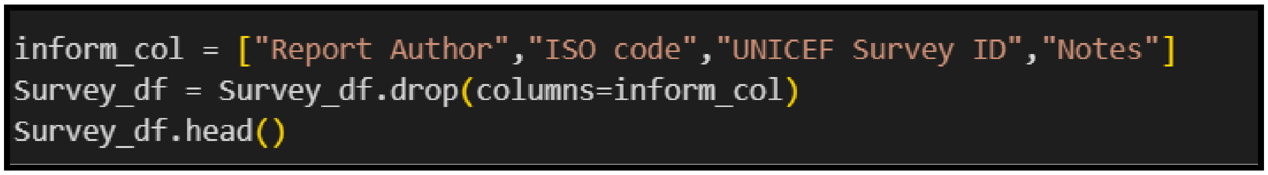
Drop Columns:

To ensure data quality and minimize noise in the model, columns containing a significant proportion of missing values—namely "LDC", "LLDC or SIDS", and "LIFD"—were removed from the dataset. These features lacked sufficient information for effective modeling and would likely have contributed more to noise than to a useful signal.



```
droped_col = ["LDC","LLDC or SIDS","LIFD"]
Survey_df = Survey_df.drop(columns=droped_col)
Survey_df.head()
```

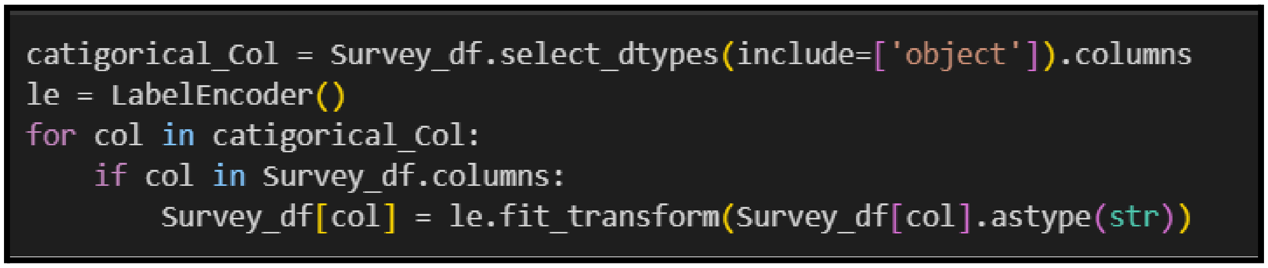
Several columns, such as "Report Author", "ISO code", "UNICEF Survey ID", and "Notes" were identified as non-informative metadata. These fields do not contribute to predictive modeling or inference and were removed to reduce dimensionality and improve model performance.



```
inform_col = ["Report Author","ISO code","UNICEF Survey ID","Notes"]
Survey_df = Survey_df.drop(columns=inform_col)
Survey_df.head()
```

Encoding Categorical Features

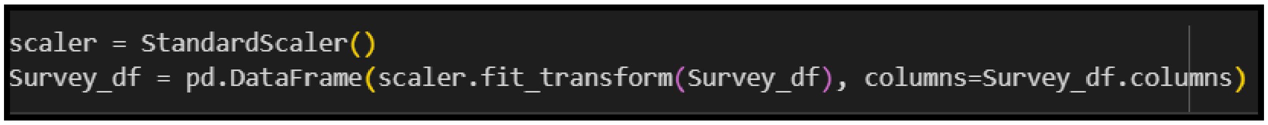
Categorical variables in the dataset were encoded using Label Encoding to convert textual data into a numerical format. This transformation is essential for enabling machine learning algorithms, which typically require numerical input. All columns with data type object were included in this transformation to standardize the dataset.



```
catigorical_Col = Survey_df.select_dtypes(include=['object']).columns
le = LabelEncoder()
for col in catigorical_Col:
    if col in Survey_df.columns:
        Survey_df[col] = le.fit_transform(Survey_df[col].astype(str))
```

Data Normalization

To ensure all features contribute equally to the model and to enhance convergence in algorithms sensitive to scale, the dataset was normalized using StandardScaler. This technique transforms the data to have a mean of 0 and a standard deviation of 1, which is particularly beneficial for distance-based models or gradient descent optimization.

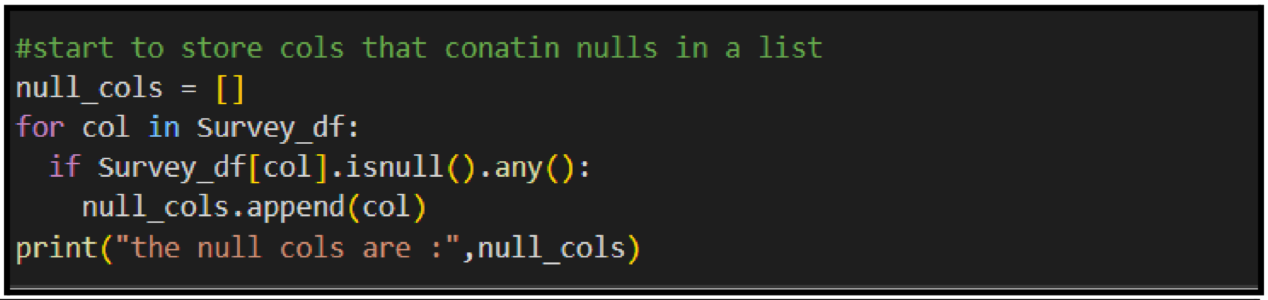


```
scaler = StandardScaler()
Survey_df = pd.DataFrame(scaler.fit_transform(Survey_df), columns=Survey_df.columns)
```

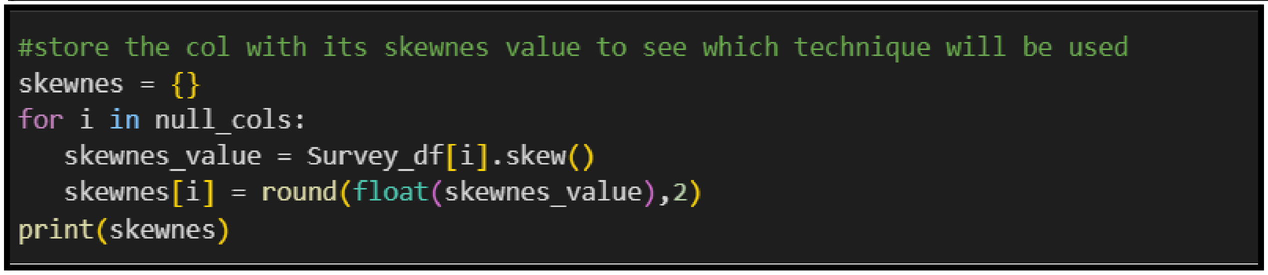
Digital Egypt Pioneers Initiative-DEPI
AI & Data Science - Data Scientist-Round Two
Healthcare Predictive Analytics Project
ONL2_AIS4_S1 Data Scientist Group

Handling Missing Values via Imputation

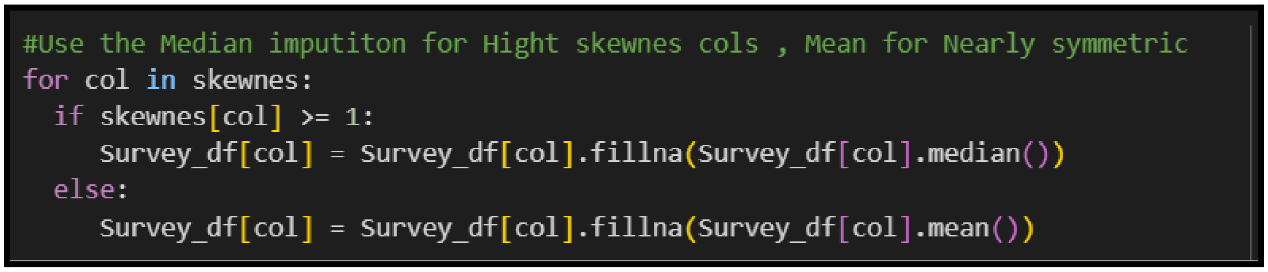
An imputation strategy was implemented for columns containing missing values. First, the skewness of each column was computed to determine the appropriate method. For highly skewed columns (skewness ⩾ 1), median imputation was applied to reduce the impact of outliers. For nearly symmetric distributions, mean imputation was used. This hybrid strategy helped maintain the statistical integrity of the data while filling in gaps.



```
#start to store cols that conatin nulls in a list
null_cols = []
for col in Survey_df:
  if Survey_df[col].isnull().any():
    null_cols.append(col)
print("the null cols are :",null_cols)
```



```
#store the col with its skewnes value to see which technique will be used
skewnes = {}
for i in null_cols:
   skewnes_value = Survey_df[i].skew()
   skewnes[i] = round(float(skewnes_value),2)
print(skewnes)
```



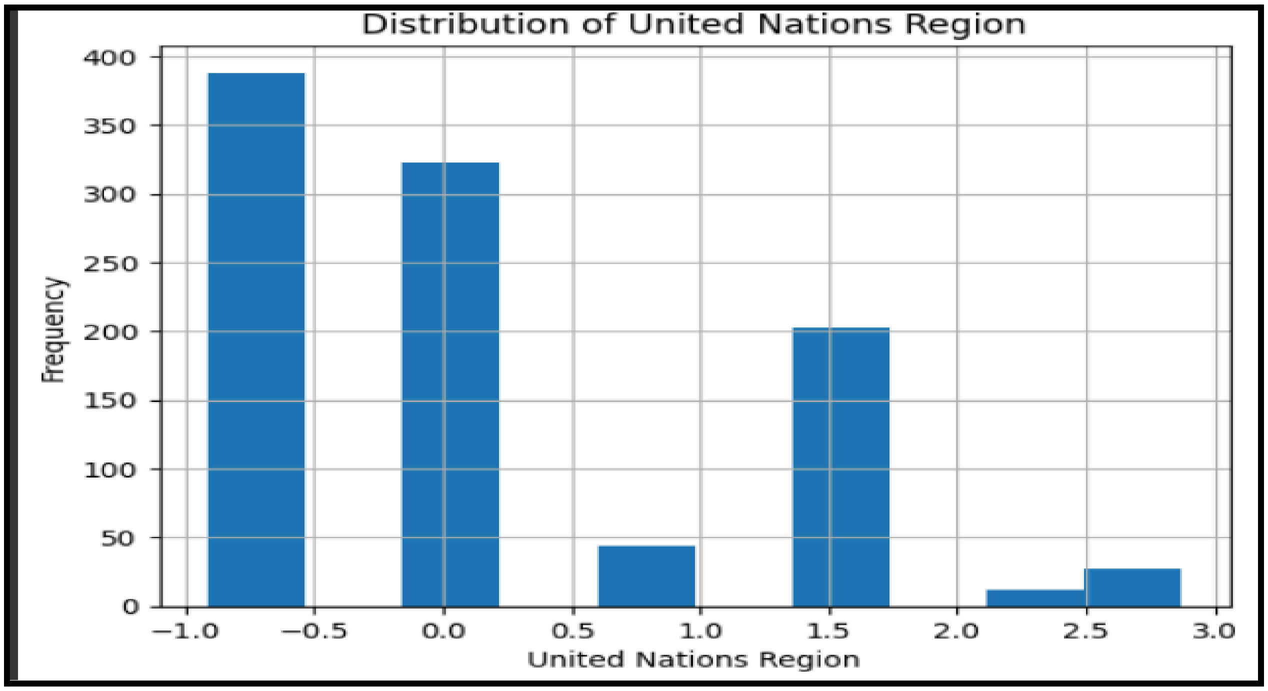
```
#Use the Median imputiton for Hight skewnes cols , Mean for Nearly symmetric
for col in skewnes:
  if skewnes[col] >= 1:
     Survey_df[col] = Survey_df[col].fillna(Survey_df[col].median())
  else:
     Survey_df[col] = Survey_df[col].fillna(Survey_df[col].mean())
```

Exploratory Data Analysis (EDA) Summary

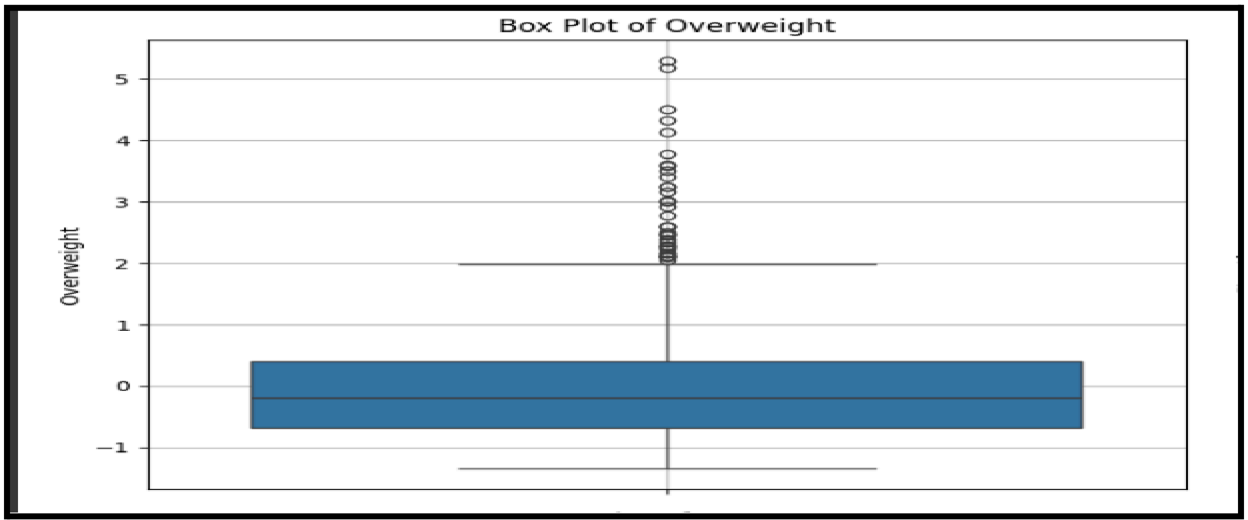
To understand the underlying structure of the dataset and detect anomalies, a comprehensive Exploratory Data Analysis (EDA) was performed.

Digital Egypt Pioneers Initiative-DEPI
AI & Data Science - Data Scientist-Round Two
Healthcare Predictive Analytics Project
ONL2_AIS4_S1 Data Scientist Group

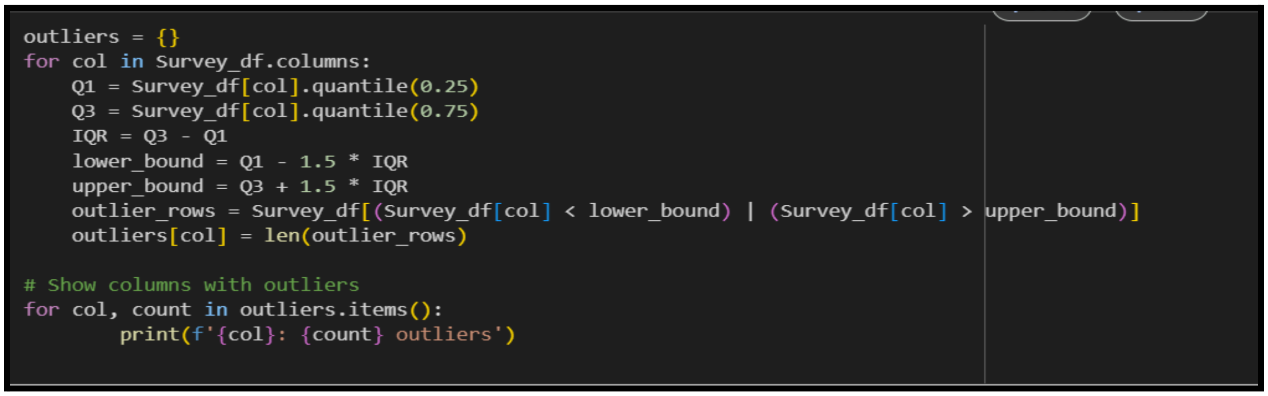
First, histograms were plotted for each feature to visualize their distributions and assess skewness or modality



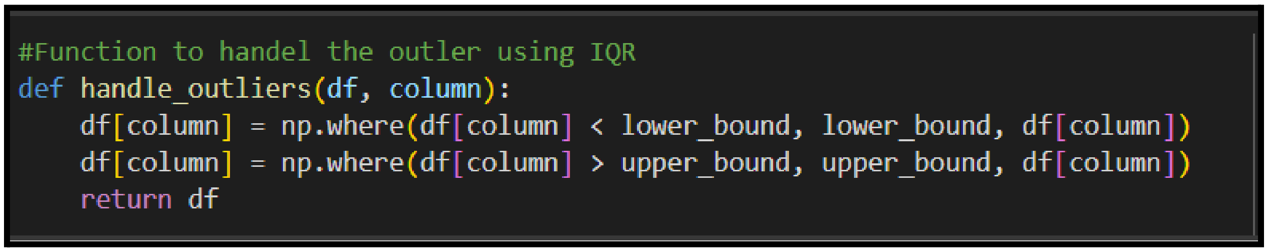
Box plots were then used to identify potential outliers and inspect data spread across all numerical variables. The **Interquartile Range (IQR)** method was applied to detect outliers in each column by calculating the 1.5*IQR bounds. A custom function was created to cap these outliers to the lower or upper IQR thresholds, thereby preserving the overall distribution while reducing the impact of extreme values. After outlier handling, updated box plots and histograms were generated to verify improvements in feature spread and distribution.



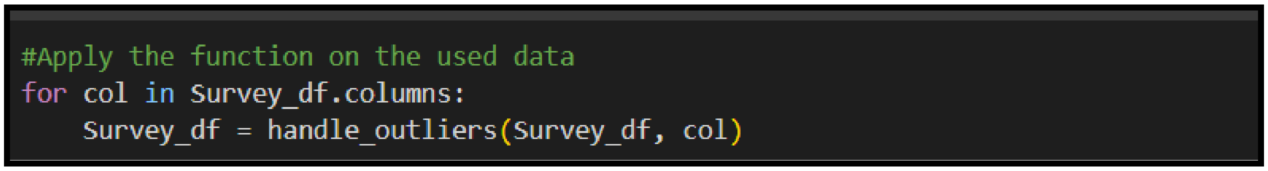
Digital Egypt Pioneers Initiative-DEPI
AI & Data Science - Data Scientist-Round Two
Healthcare Predictive Analytics Project
ONL2_AIS4_S1 Data Scientist Group



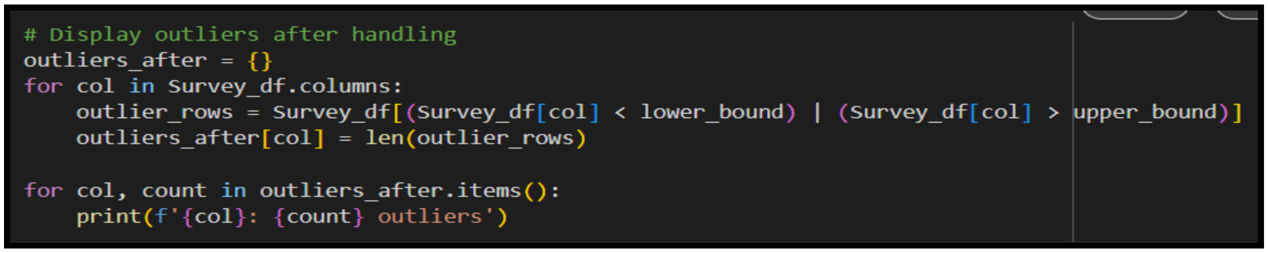
```
outliers = {}
for col in Survey_df.columns:
    Q1 = Survey_df[col].quantile(0.25)
    Q3 = Survey_df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outlier_rows = Survey_df[(Survey_df[col] < lower_bound) | (Survey_df[col] > upper_bound)]
    outliers[col] = len(outlier_rows)

# Show columns with outliers
for col, count in outliers.items():
        print(f'{col}: {count} outliers')
```



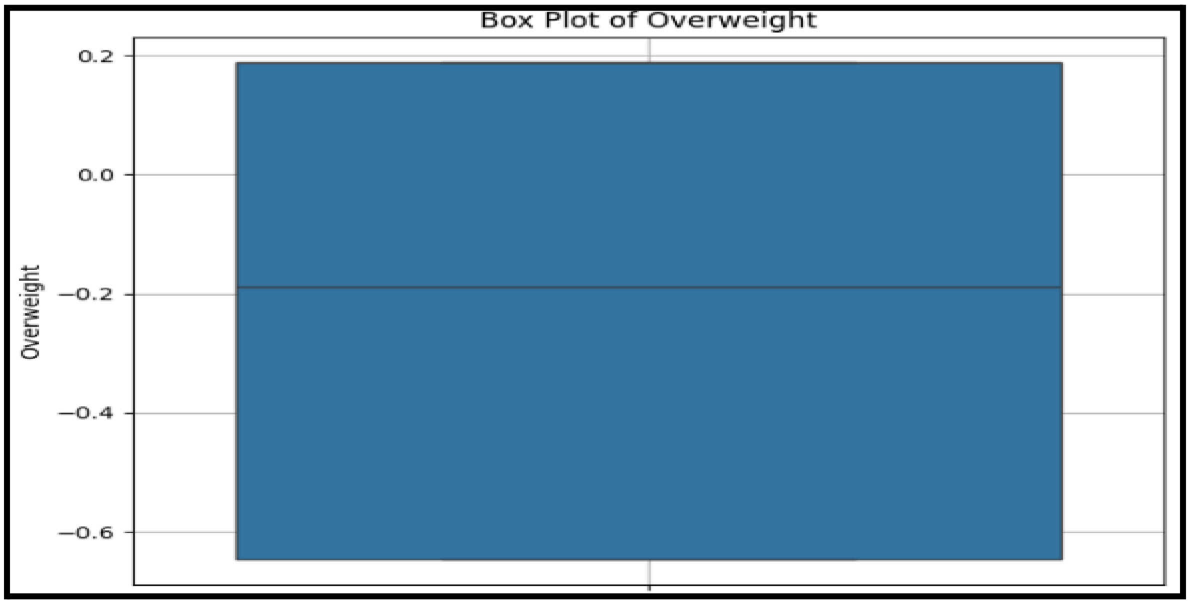
```
#Function to handel the outler using IQR
def handle_outliers(df, column):
    df[column] = np.where(df[column] < lower_bound, lower_bound, df[column])
    df[column] = np.where(df[column] > upper_bound, upper_bound, df[column])
    return df
```



```
#Apply the function on the used data
for col in Survey_df.columns:
    Survey_df = handle_outliers(Survey_df, col)
```

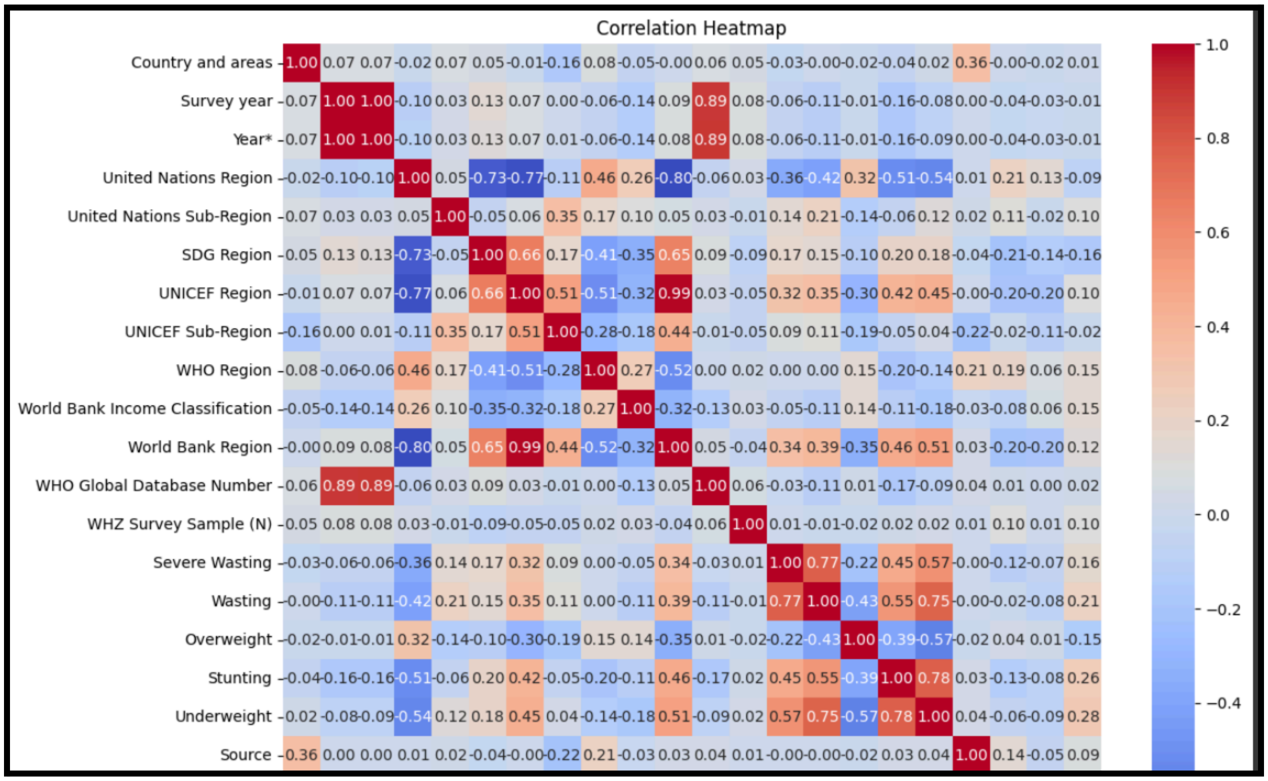


```
# Display outliers after handling
outliers_after = {}
for col in Survey_df.columns:
    outlier_rows = Survey_df[(Survey_df[col] < lower_bound) | (Survey_df[col] > upper_bound)]
    outliers_after[col] = len(outlier_rows)

for col, count in outliers_after.items():
    print(f'{col}: {count} outliers')
```

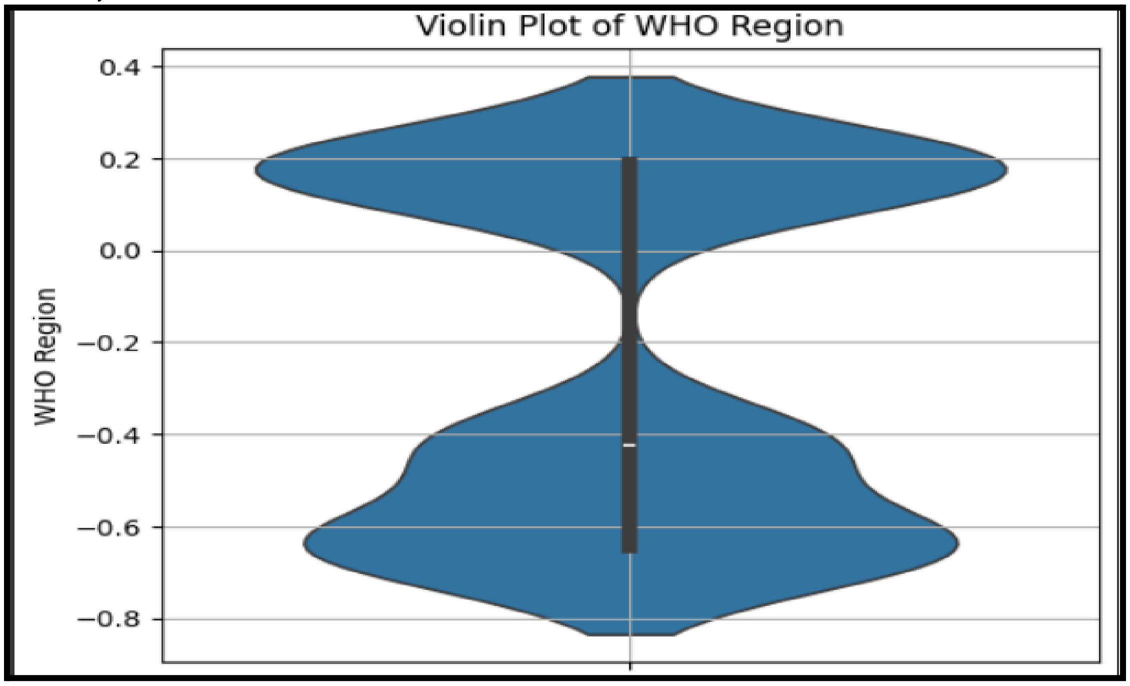


Digital Egypt Pioneers Initiative-DEPI
AI & Data Science - Data Scientist-Round Two
Healthcare Predictive Analytics Project
ONL2_AIS4_S1 Data Scientist Group

A correlation heatmap was constructed to explore relationships among numeric variables, revealing patterns and potential multicollinearity.



Finally, violin plots were employed for each feature to combine distributional insights with density visualization.


Violin Plot of WHO Region
WHO Region