FTL Global ML bootcamp 3 | Capstone project proposal | Group 13

Malnutrition Risk Prediction ML Model

Data Preparation, Feature Engineering and Model Exploration

Group members:

- Lina Ahmed
- Linda Adil
- Maha Abdalfedil
- Nada Ali

1. Overview

Data preparation and feature engineering phase involves cleaning the data, handling missing values, and transforming raw data into meaningful features that can improve model efficiency.

In our project, we began by addressing the issue of missing values and zeros in the dataset, utilizing techniques such as K-nearest neighbors (KNN) imputation to replace missing values with more informative estimates based on the similarity between data points. Feature selection followed, where we identified and retained the most relevant features, such as **malnutrition indicators** (stunted, wasted, underweight_bmi) and **socio-economic factors** (poorest).

Normalization using StandardScaler was applied to ensure that all features contribute equally to the model. These steps are significant because they help in enhancing the model's ability to learn from the data effectively, reduce noise, and avoid potential biases, ultimately leading to more robust and accurate predictions.

2. Data Collection

The dataset used in this project was sourced from Kaggle, a well-known platform for data science competitions and datasets. This dataset focuses on various socio-economic and health indicators relevant to predicting malnutrition and poverty in Least Developed Countries (LDCs)

3. Data Cleaning

Handling Missing Values:

Columns with significant missing values were dropped after verifying they were not essential for the model

```
[321] df.iloc[:,9:105].head()
```

	marketm0	marketm1	marketm10	marketm11	marketm12	marketm13	marketm14	marketm15	marketm16	marketm17	...	markets43	markets44	markets45	mar
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	

5 rows × 96 columns

And we end up with:

```
[322] df.drop(df.columns[9:105], axis=1, inplace=True)
```

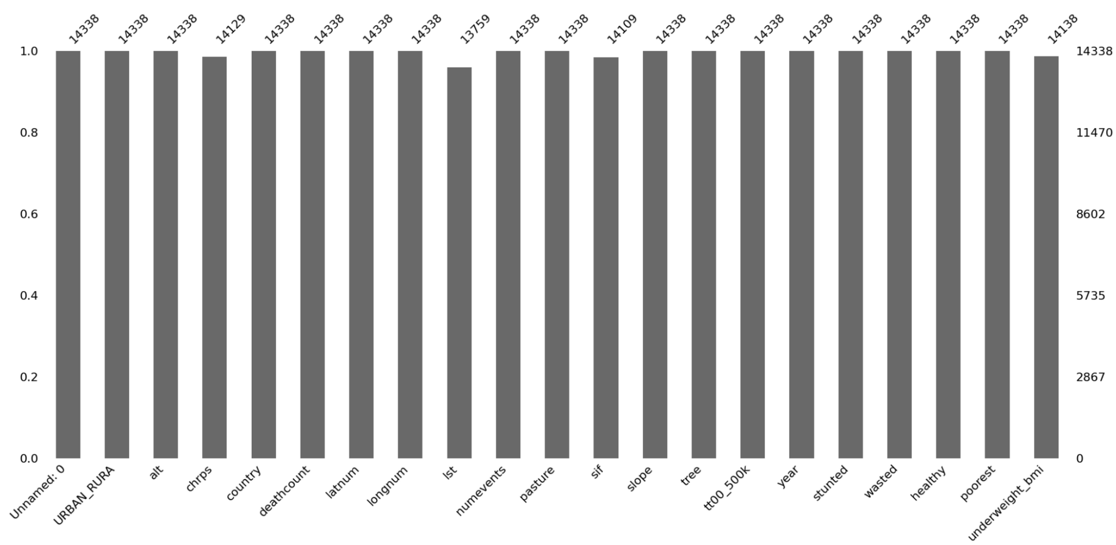
```
[323] df.head()
```

	Unnamed: 0	URBAN_RURA	alt	chrps	country	deathcount	latnum	longnum	lst	numevents	...	sif	slope	tree	tt00_500k	year	stunted	wa
0	0	1	5.603	0.437	Bangladesh	0	22.982	90.156	-1.066	14	...	-0.745	0.022	16.344	365.678	2004	0.294	0
1	1	1	4.678	0.448	Bangladesh	0	22.444	90.329	-0.963	14	...	-0.841	0.007	0.000	397.886	2004	0.444	0
2	2	1	5.145	0.453	Bangladesh	0	22.487	90.206	-0.963	14	...	-0.842	0.005	0.000	344.366	2004	0.600	0
3	3	1	6.175	0.433	Bangladesh	0	23.016	90.193	-1.075	14	...	-0.745	0.028	16.033	369.385	2004	0.500	0
4	4	1	5.391	0.406	Bangladesh	0	22.952	90.454	-1.065	14	...	-0.748	0.025	18.765	273.924	2004	0.556	0

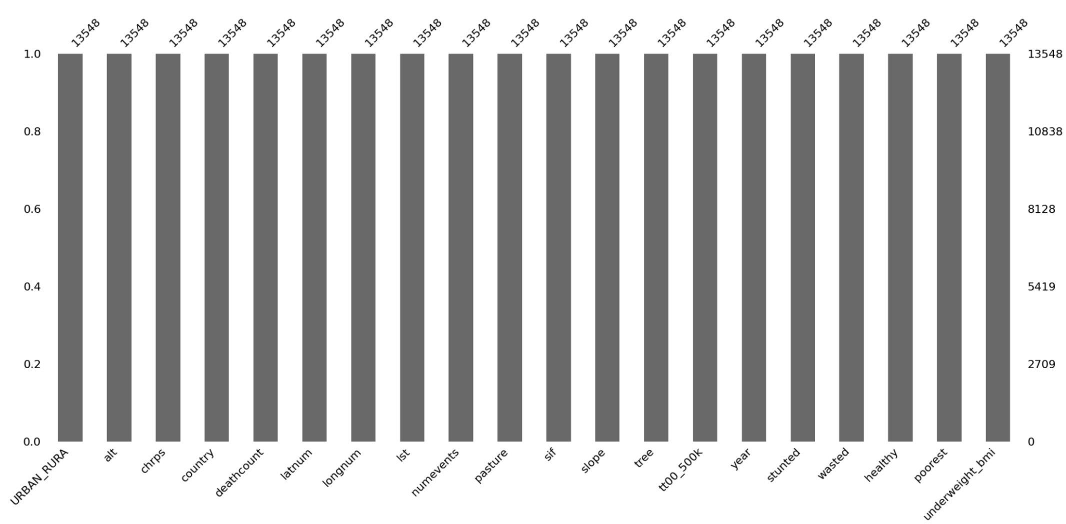
5 rows × 21 columns

Then we drop rows with null values, after finding that they were a small amount from the data:

So we went from this



To this:



Also, we noticed a lot of zeros in the dataset that was affecting the distribution, so we can't just ignore it nor drop it, so we solve this using KNN

```
from sklearn.impute import KNNImputer

# Get columns with zeros excluding 'URBAN_RURA'
zero_columns = (df == 0).mean() * 100
zero_columns = zero_columns[zero_columns > 0].index.tolist()
zero_columns.remove('URBAN_RURA')

# Replace zeros with NaN only in the columns from zero_columns
df[zero_columns] = df[zero_columns].replace(0, np.nan)

# Initialize KNNImputer
imputer = KNNImputer(n_neighbors=5)

# Apply the imputer only to the columns in zero_columns
df[zero_columns] = pd.DataFrame(imputer.fit_transform(df[zero_columns]),
                                columns=zero_columns,
                                index=df.index)
```

Here we can see the count of zero's before and after applying the KNN

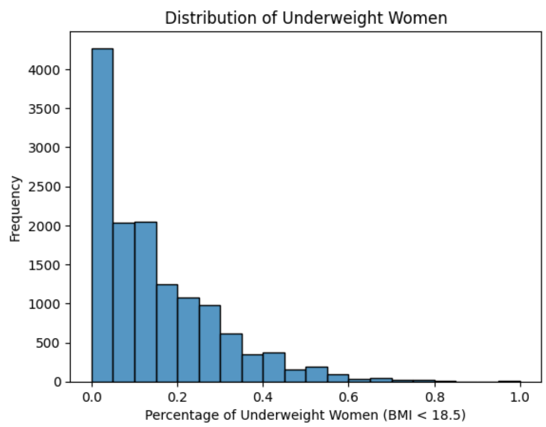
	0
URBAN_RURA	35.179
alt	0.686
chrps	0.000
country	0.000
deathcount	42.619
latnum	0.679
longnum	0.679
lst	0.000
numevents	16.357
pasture	13.013
sif	0.000
slope	0.775
tree	15.574
tt00_500k	0.701
year	0.000
stunted	14.467
wasted	49.099
healthy	0.340
poorest	60.592
underweight_bmi	24.483

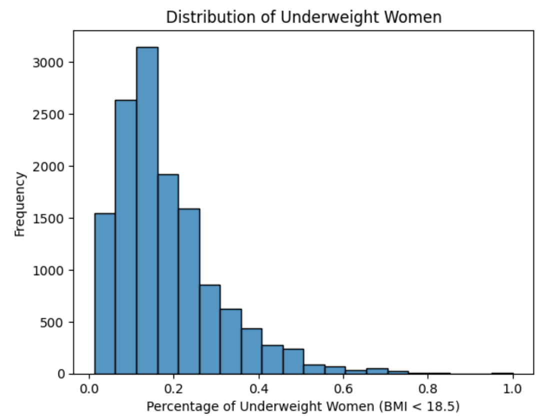
dtype: float64

	0
URBAN_RURA	35.179
alt	0.000
chrps	0.000
country	0.000
deathcount	0.000
latnum	0.000
longnum	0.000
lst	0.000
numevents	0.000
pasture	0.000
sif	0.000
slope	0.000
tree	0.000
tt00_500k	0.000
year	0.000
stunted	0.000
wasted	0.000
healthy	0.000
poorest	0.000
underweight_bmi	0.000

dtype: float64

And here we can see the effect of KNN of the distribution of the features:





4. Exploratory Data Analysis (EDA)

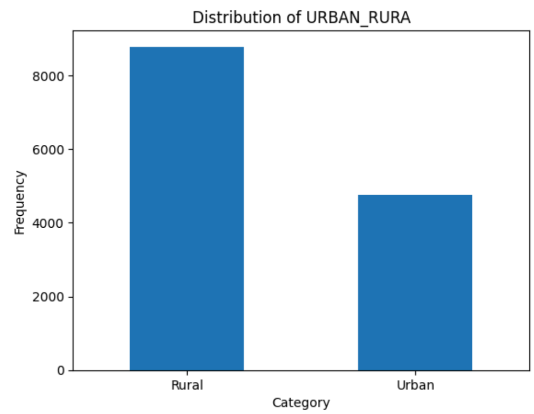
In this phase of the project, we aimed to understand the underlying patterns, distributions, and relationships within the dataset. This step guides us to feature engineering and model selection part

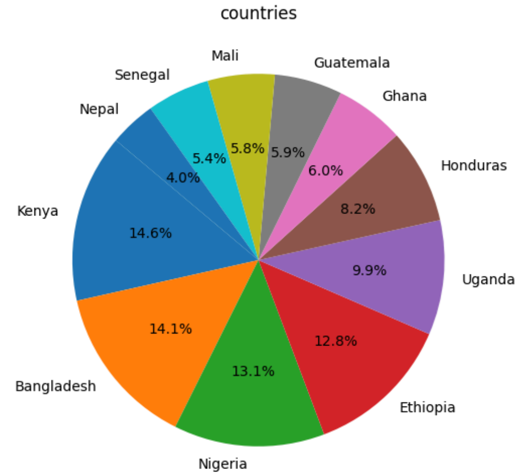
4.1. Distribution of Features:

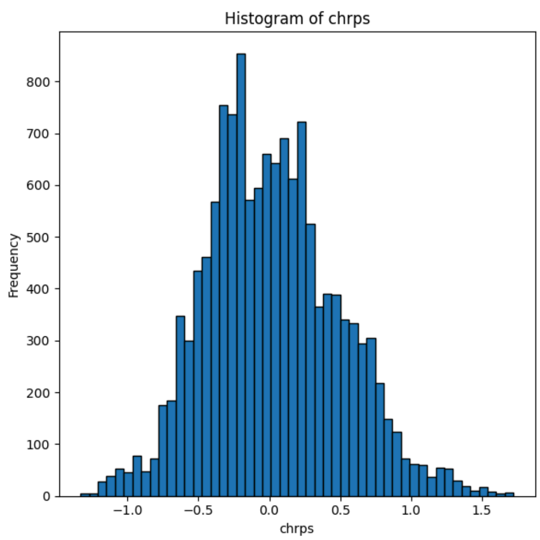
Features like altitude, pasture, and tree cover displayed right-skewed distributions, indicating that most data points are clustered around lower values.

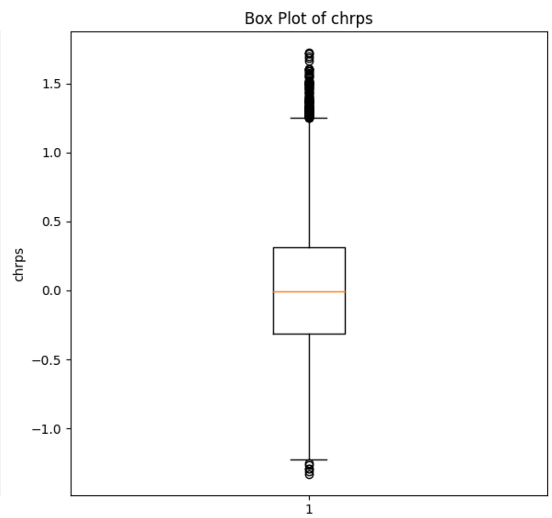
	count	mean	std	min	25%	50%	75%	max
URBAN_RURA	13548.000	0.648	0.478	0.000	0.000	1.000	1.000	1.000
alt	13548.000	753.504	762.200	0.000	49.334	404.679	1308.330	4832.420
chrps	13548.000	0.021	0.469	-1.331	-0.311	-0.009	0.314	1.724
deathcount	13548.000	78.867	148.552	0.000	0.000	3.000	71.000	659.000
latnum	13548.000	10.930	8.477	-4.661	3.944	10.910	15.023	29.966
longnum	13548.000	16.630	52.764	-92.176	-3.008	32.543	39.406	92.414
lst	13548.000	-0.037	0.566	-1.675	-0.408	-0.037	0.347	1.601
numevents	13548.000	64.493	100.531	0.000	2.000	21.000	77.000	560.000
pasture	13548.000	0.152	0.177	0.000	0.017	0.086	0.231	0.966
sif	13548.000	0.094	0.520	-1.402	-0.260	0.034	0.426	1.712
slope	13548.000	1.761	2.500	0.000	0.177	0.656	2.314	24.315
tree	13548.000	24.919	17.247	0.000	13.942	23.000	35.064	80.000
tt00_500k	13548.000	350.739	263.858	0.000	175.422	313.817	475.270	3337.030
year	13548.000	2010.943	3.453	2004.000	2008.000	2011.000	2014.000	2016.000
stunted	13548.000	0.325	0.226	0.000	0.154	0.312	0.500	1.000
wasted	13548.000	0.094	0.131	0.000	0.000	0.042	0.154	1.000
healthy	13548.000	0.863	0.152	0.000	0.795	0.889	1.000	1.000
poorest	13548.000	0.176	0.296	0.000	0.000	0.000	0.238	1.000
underweight_bmi	13548.000	0.144	0.144	0.000	0.023	0.111	0.217	1.000

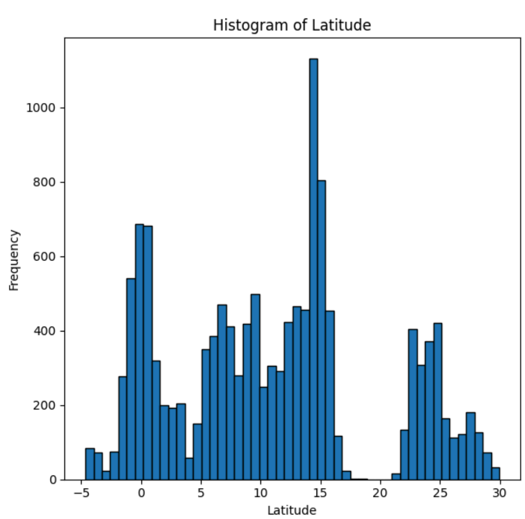
Count plots and bar plots were used to visualize the distribution of these indicators, here're some of them:

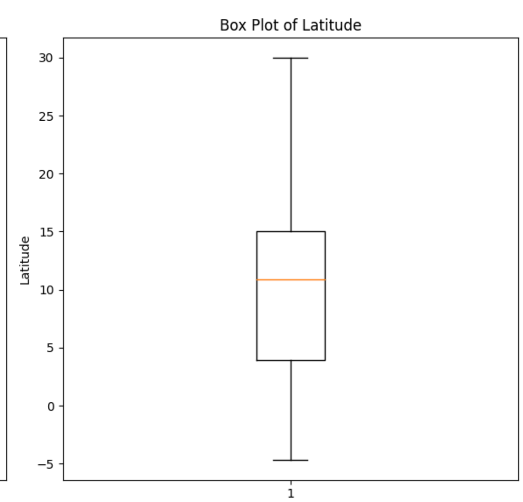


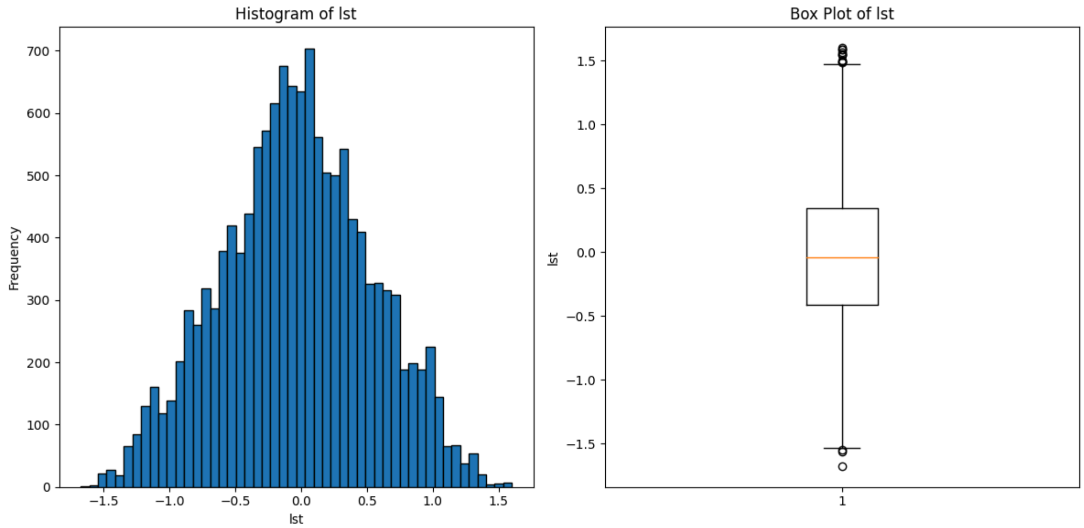


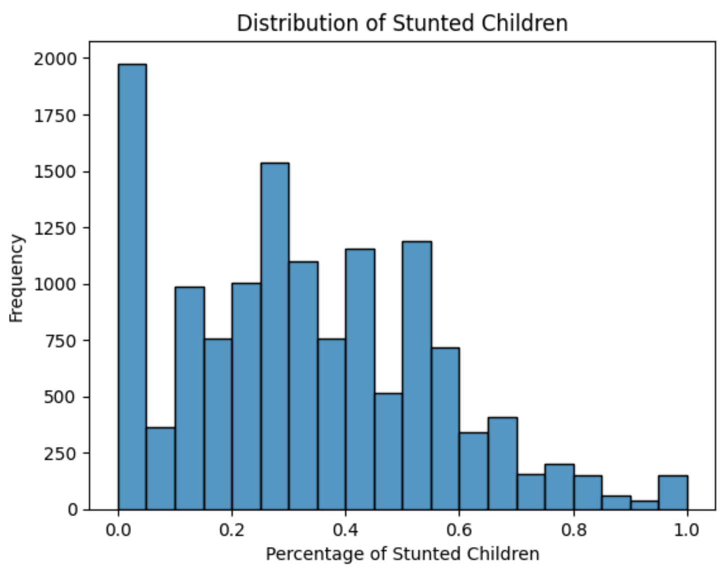


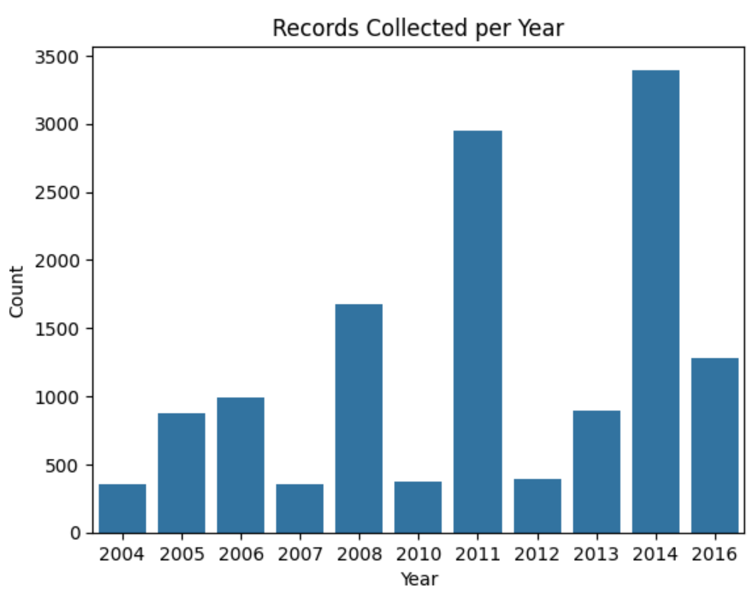






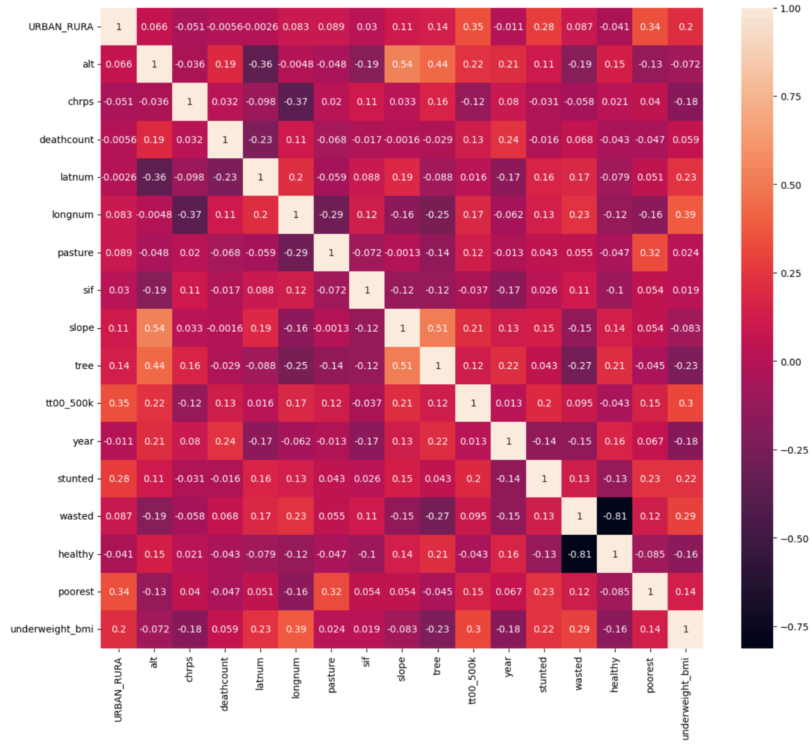

Histogram of lst
Frequency
lst
Box Plot of lst
lst
1


Distribution of Stunted Children
Frequency
Percentage of Stunted Children


Records Collected per Year
Count
2004
2005
2006
2007
2008
2010
2011
2012
2013
2014
2016
Year

4.2. Correlation Analysis:

We calculated pairwise correlations between numerical features and visualized them using a heatmap

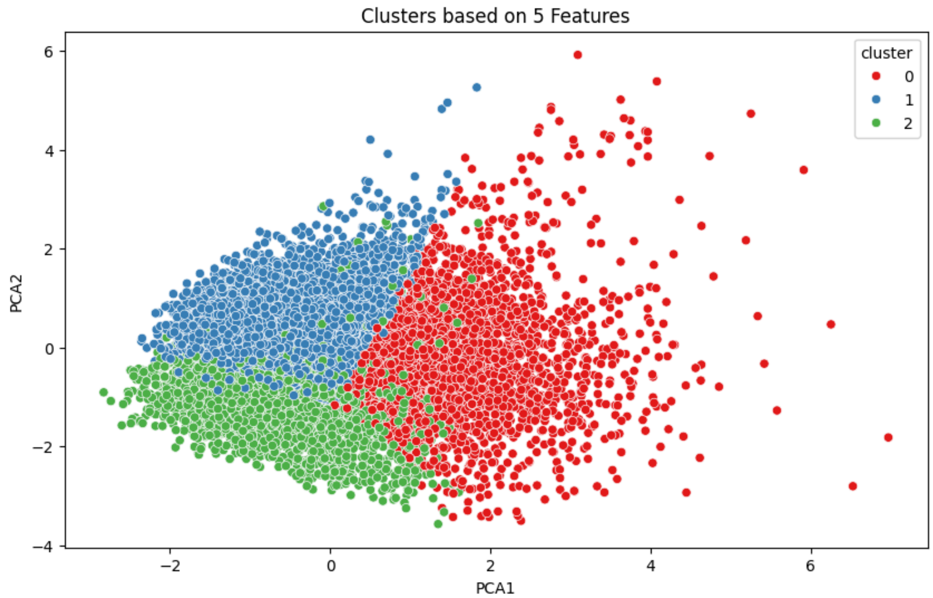


We have also calculated the corr. between the 5 key features and other features as follows:

```
[428] columns_to_check = ['stunted', 'deathcount', 'wasted', 'healthy', 'poorest', 'underweight_bmi']

      # Compute correlation with 'alt' for each column
      for column in columns_to_check:
          correlation = df['longnum'].corr(df[column])
          print(f"Correlation between longnum and {column}: {correlation}")

      Correlation between longnum and stunted: 0.1322009122347264
      Correlation between longnum and deathcount: 0.12547224882804003
      Correlation between longnum and wasted: 0.243407141411886
      Correlation between longnum and healthy: -0.13107793184967803
      Correlation between longnum and poorest: -0.15935640351369024
      Correlation between longnum and underweight_bmi: 0.3966362315664159
```

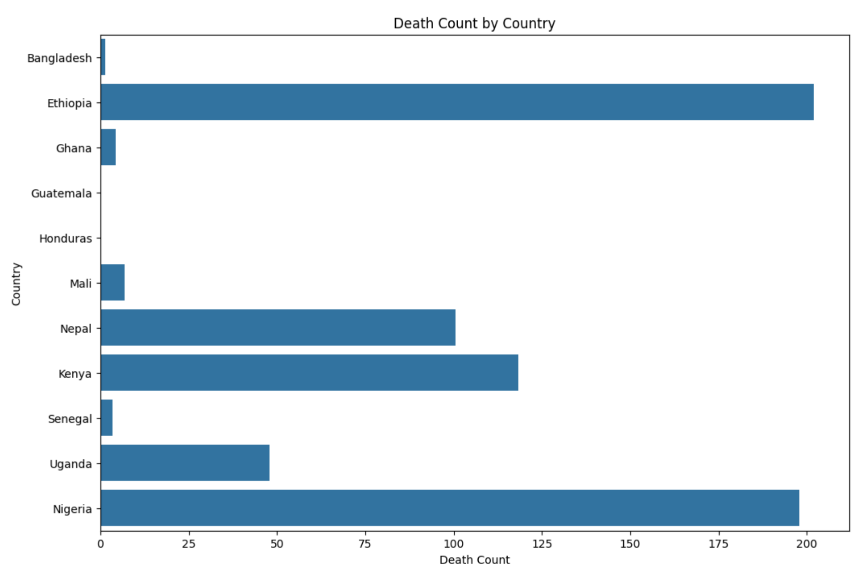
4.3. Clustering Analysis:

We performed K-means clustering to group the data into three clusters based on five features (stunted, poorest, underweight, wasted). The clusters were visualized to understand their distribution and characteristics

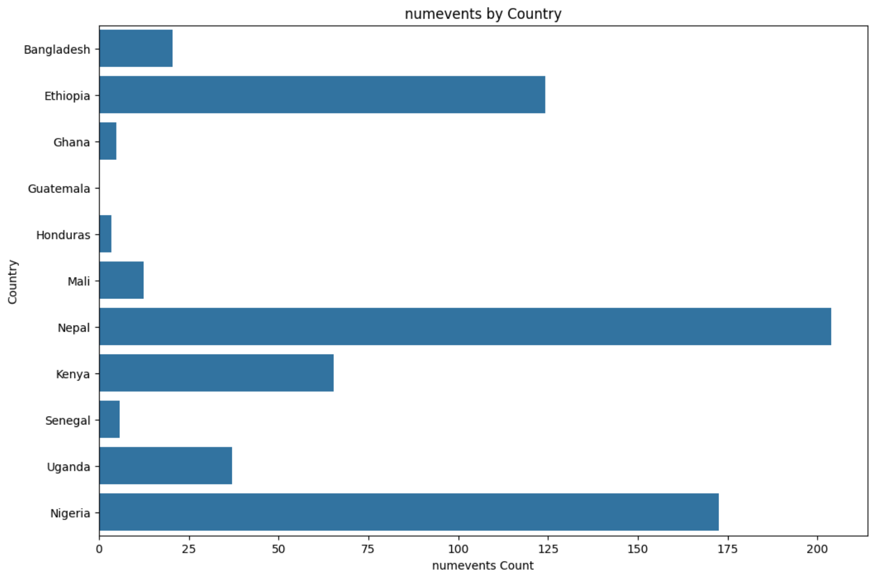


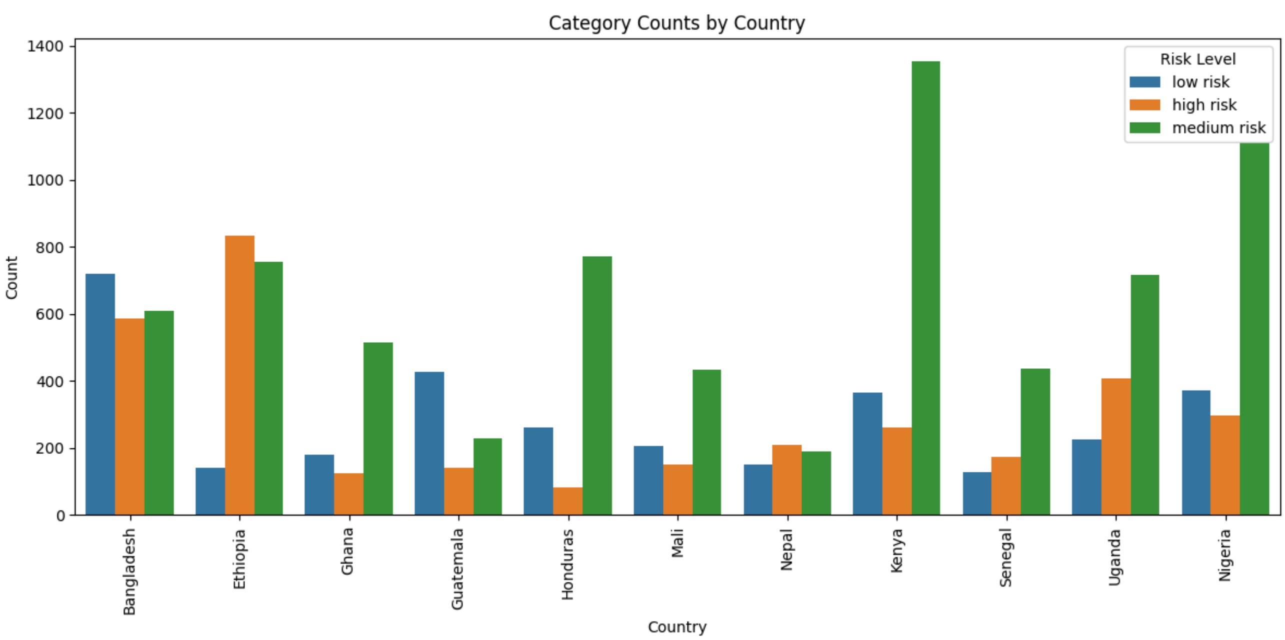
4.4. Distribution of Categories:

To understand how different categories are distributed across countries, we created grouped bar plots and stacked bar plots. This provided insights into the prevalence of different risk levels within each country.



```python
plt.figure(figsize=(12, 8))
sns.barplot(x='numevents', y='country', data=df, ci=None)
plt.title('numevents by Country')
plt.xlabel('numevents Count')
plt.ylabel('Country')
plt.show()
```





4.5. Key Insights from the EDA part:

Feature Distributions:
Features like altitude, pasture, and tree cover displayed ***right-skewed*** distributions, indicating that most data points are *clustered around lower values*.

Correlation:
There were moderate correlations between some features, such as altitude and tree cover, which can inform feature selection and model interpretation.

Clustering:
The K-means clustering results provided *a clear separation into **three** risk levels*, which can be used for further analysis and model training.

Category Distribution:
Visualizations of category distributions by country revealed the geographic spread of risk levels, highlighting areas with higher or lower risk.

5. Feature Engineering:

In the feature engineering phase, we focused on transforming existing features and creating new ones to enhance the predictive power of our model. The objective was to make the features more informative and relevant to the problem of classifying risk levels.

5.1.Risk Levels Creation:

To assess and predict malnutrition and poverty, we defined three distinct risk levels based on those features:

Stunted: Reflects the proportion of children under five years old whose height-for-age z-score is below the standard threshold, indicating chronic malnutrition. This feature is crucial for identifying long-term nutritional deficiencies.

Wasted: Represents the proportion of children under five years old who are underweight for their height, signaling acute malnutrition. This indicator helps in understanding immediate health concerns and nutritional crises.

Underweight BMI: This feature captures the proportion of women aged 15 to 49 with a body mass index (BMI) below the healthy threshold, highlighting issues related to adult malnutrition.

Poorest: reflects the socio-economic dimension by identifying households within the lowest quintile of the wealth index, providing an economic context to the malnutrition indicators.

```
[461] # Selecting 4 features for clustering
      features = ['stunted', 'wasted', 'poorest', 'underweight_bmi']  # , 'healthy'
      df_selected = df[features]
```

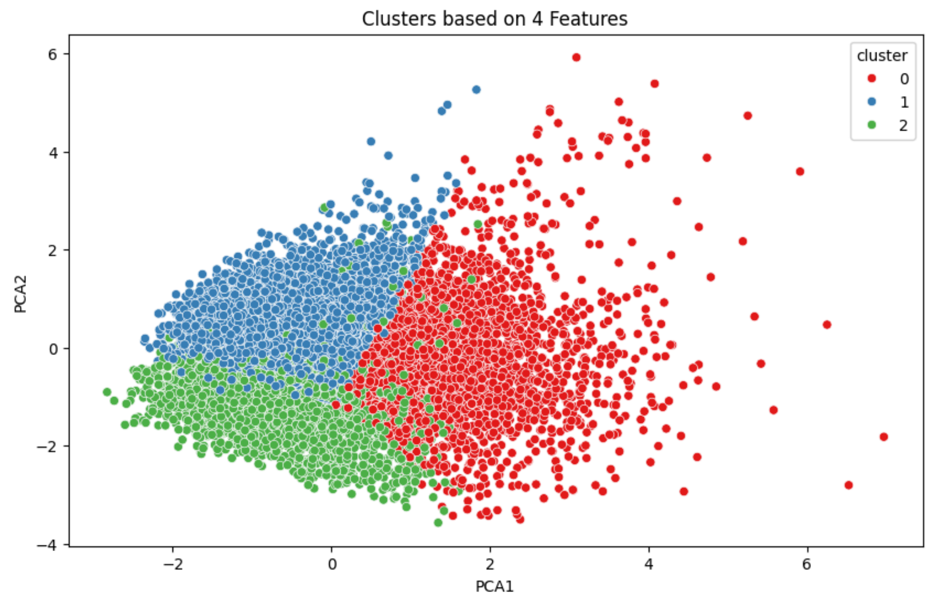
5.2. Feature Transformation and Creation:

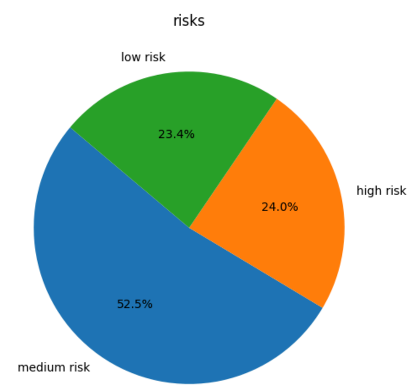
These features were selected because they directly relate to malnutrition and poverty conditions, which are key areas of focus in Least Developed Countries (LDCs). To enhance interpretability and model performance, we normalized the selected features using **StandardScaler** to ensure that each feature contributed equally to the model without being biased by differences in scale.

```
[462] scaler = StandardScaler()
      df_scaled = scaler.fit_transform(df_selected)
```

```
[463] kmeans = KMeans(n_clusters=3, random_state=42)
      df['cluster'] = kmeans.fit_predict(df_scaled)
```

Additionally, we clustered the data into three risk levels using K-means clustering, which grouped the data points based on similarities in malnutrition and poverty indicators. The rationale behind creating these clusters was to classify regions or populations into distinct risk levels:





- **High Risk**: Regions or populations with severe malnutrition and poverty conditions, characterized by higher stunted growth, wasted children, and a higher proportion of underweight BMI.
- **Moderate Risk**: Areas with moderate levels of malnutrition and poverty indicators, showing signs of improvement but still requiring significant interventions.
- **Low Risk**: Regions with relatively better health outcomes and economic conditions, where malnutrition and poverty risks are lower, but not completely absent.

```
[464] print(df['cluster'].value_counts())

cluster
1    7119
0    3256
2    3173
Name: count, dtype: int64
```

```
[470] cluster_summary = df.groupby('cluster').agg({
          'stunted': 'mean',        # Example of numeric column
          'wasted': 'mean',        # Example of numeric column
          'poorest': 'mean',       # Example of numeric column
          'numevents': 'mean',
          'deathcount': 'mean',
          'healthy': 'mean',       # Example of numeric column
          'underweight_bmi': 'mean',  # Example of numeric column
      })
      print(cluster_summary)
```

```
         stunted  wasted  poorest  numevents  deathcount  healthy  \
cluster
0          0.557   0.232    0.650     82.325     122.385    0.821
1          0.296   0.186    0.654     77.781     122.299    0.885
2          0.354   0.174    0.214     65.492     122.250    0.869

         underweight_bmi
cluster
0                  0.304
1                  0.135
2                  0.171
```

These clusters were then labeled as risk categories, allowing us to focus on making predictions based on the risk levels associated with each group.

```
[471] # Define a mapping from cluster numbers to risk labels
      cluster_labels = {0: 'high risk', 1: 'medium risk', 2: 'low risk'}

      # Apply the mapping to the cluster column
      df['risk'] = df['cluster'].map(cluster_labels)
```

And finally, here's the updated df:

```
df.head()
```

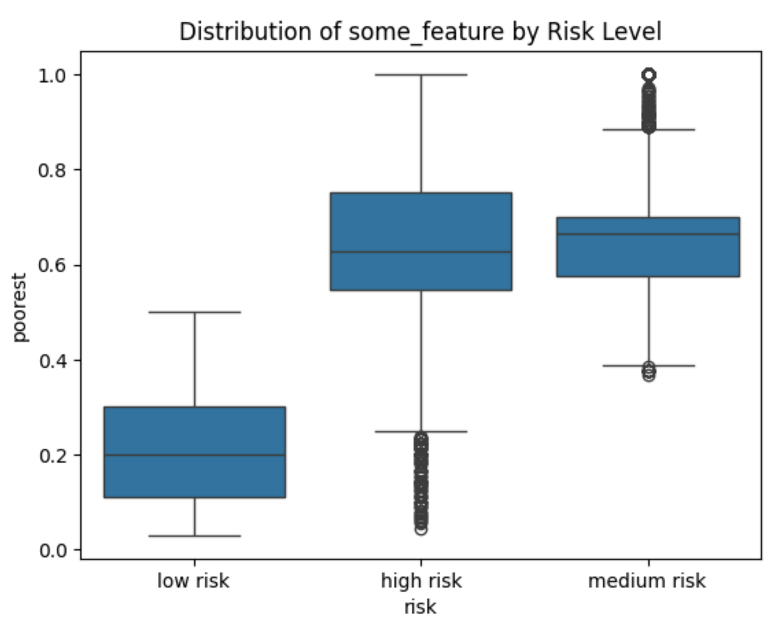
deathcount	latnum	longnum	lst	numevents	pasture	...	year	stunted	wasted	healthy	poorest	underweight_bmi	cluster	PCA1	PCA2	risk
6.400	22.982	90.156	-1.066	14.000	0.049	...	2004	0.294	0.059	0.941	0.071	0.273	2	-0.919	-2.240	low risk
7.800	22.444	90.329	-0.963	14.000	0.043	...	2004	0.444	0.203	1.000	0.062	0.400	2	0.771	-2.127	low risk
6.200	22.487	90.206	-0.963	14.000	0.030	...	2004	0.600	0.050	0.950	0.707	0.355	0	1.249	-1.191	high risk
6.400	23.016	90.193	-1.075	14.000	0.052	...	2004	0.500	0.062	0.938	0.062	0.324	2	0.018	-2.758	low risk
6.200	22.952	90.454	-1.065	14.000	0.042	...	2004	0.556	0.188	1.000	0.111	0.278	2	0.512	-1.919	low risk

6. Model Selection

For this project, several machine learning models were considered to predict malnutrition and poverty levels based on the identified risk clusters. First thing, we tried **Random Forest** as the primary model for classification. The rationale behind choosing Random Forest lies in its versatility and robustness when dealing with complex, high-dimensional datasets like ours, which contains both socio-economic and health-related features

```
[468] from sklearn.ensemble import RandomForestClassifier
      from sklearn.model_selection import train_test_split
      from sklearn.metrics import classification_report

      # Prepare the data
      X = df[['stunted', 'wasted', 'poorest', 'underweight_bmi']]  # Use selected features
      y = df['cluster']  # Cluster labels as target

      # Split the data
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

      # Train a classifier
      model = RandomForestClassifier(random_state=42)
      model.fit(X_train, y_train)
      y_pred = model.predict(X_test)

      # Evaluate the model
      print(classification_report(y_test, y_pred))
```

The results shows high accuracy so we consider having **overfitting**, and this box plots can show this:



We've also checked the stability of our model using cross validation and it shows those results

```
[469] from sklearn.model_selection import cross_val_score

      # Cross-validation with the existing model
      cv_scores = cross_val_score(model, X, y, cv=5, scoring='f1_weighted')
      print(f'Cross-Validation Scores: {cv_scores}')
      print(f'Mean Cross-Validation Score: {cv_scores.mean()}')
```

```
Cross-Validation Scores: [0.96392957 0.97791815 0.97290555 0.97933667 0.98445791]
Mean Cross-Validation Score: 0.9757095710150983
```

7. Conclusion

In this project, we explored the prediction of malnutrition and poverty risk levels in Least Developed Countries (LDCs) using a machine learning approach. Starting with extensive data preparation and feature engineering, we transformed critical socio-economic and health indicators into meaningful features.

After applying clustering techniques to categorize regions or populations into three risk levels, we selected the Random Forest model for classification due to its robustness and ability to handle complex, high-dimensional data.

While the model initially produced high performance metrics, including accuracy and recall, we noticed signs of overfitting, particularly when evaluating cross-validation results.

Overfitting suggests that the model may be too closely fitted to the training data, leading to potential issues when generalizing to new data.

In response to this, ***we are now reevaluating*** our data handling processes and considering different machine learning models to improve generalization and avoid overfitting. Models such as Support Vector Machines (SVM), Gradient Boosting, and more advanced regularization techniques are being tested to achieve better long-term performance and stability.

As we continue refining our model, the focus remains on improving prediction accuracy while ensuring that the model remains generalizable across different populations, ensuring its practical application for NGOs and policymakers in addressing malnutrition and poverty.