G H Patel College of Engineering & Technology
(A Constituent College of CVM University)

Global Disaster Analysis

A project report submitted in partial fulfilment of the requirements for the Degree of Bachelor of Engineering & Technology

in

COMPUTER ENGINEERING (CP)
SEM – V

by

Name: Param Dholakia, Meet Dadhaniya

Enrollment No.:12202040501049, 12202040501038

Under supervision of

Prof. Khyati Mehta

Academic Year 2024-25 (ODD)

Department of Computer Engineering
G H Patel College of Engineering &Technology
Bakrol Road, Vallabh Vidyanagar

Global Disaster Analysis and Prediction

1. Introduction

Natural disasters have a profound impact on human societies, economies, and environments. By analyzing past disaster data, we can better understand these events, helping governments and organizations implement more effective preparedness and response strategies.

This project aims to analyze a global dataset of disasters, identify significant trends, and predict future disaster occurrences. Key steps include cleaning the dataset, performing exploratory data analysis (EDA), and applying machine learning models for prediction.

Objective

The objective of this project is to:
1. Load and clean the global disaster dataset.
2. Perform exploratory data analysis to identify trends and patterns.
3. Implement machine learning algorithms to predict future disasters.
4. Visualize results and interpret the key findings.

2. Dataset Description

The dataset used in this analysis is a comprehensive collection of disaster events across the globe. It contains over 15,000 entries detailing various disasters such as floods, earthquakes, hurricanes, and wildfires. Key attributes include:
- Disaster Type: Type of disaster (e.g., flood, earthquake).
- Country and Region: The location where the disaster occurred.
- Start and End Dates: The timeframe of the disaster.
- Magnitude: The strength or severity of the disaster.
- Total Deaths, Injured, and Affected: Human impact metrics.
- Total Damage: The economic impact, expressed in dollars.

Data cleaning was essential as some fields, such as geographic coordinates and economic damage, had missing or incorrect values.

3. Data Pre-processing

To prepare the dataset for analysis, several preprocessing steps were taken:

3.1 Handling Missing Values

The dataset had several missing values, particularly in fields like Magnitude, Total Damage, and geographic coordinates. Missing values were handled by either filling them with median values or dropping rows if essential information (like disaster type) was missing.

[28]:
```
# Check for missing values in the dataset
df.isnull().sum()
```

[31]:
```
# Handle missing values
df.fillna(0, inplace=True)
df.isnull().sum()
```

3.2 Date-Time Processing

The Entry Date and Last Update columns were converted into datetime objects to allow easier manipulation and analysis of time-related trends.

[29]:
```
# Convert date columns to datetime format
df['Entry Date'] = pd.to_datetime(df['Entry Date'], errors='coerce')
df['Last Update'] = pd.to_datetime(df['Last Update'], errors='coerce')
```

4. Exploratory Data Analysis (EDA)

Exploratory data analysis was performed to gain insights into the dataset, including disaster frequency over time, the most affected countries, and correlations between disaster magnitude and damage.

4.1 Count-Wise Disaster Analysis

The top 10 most disaster-affected countries were identified, with the United States, India, and China being the most frequently affected regions. A bar plot was used to visualize the disaster count by country.

[31]:
```
# Create the plot
plt.figure(figsize=(12, 6))
plt.bar(disaster_counts.index, disaster_counts.values)

# Rotate x-axis labels
plt.xticks(rotation=90)

# Add title and labels
plt.title('Distribution of Disaster Types')
plt.xlabel('Disaster Type')
plt.ylabel('Count')

# Show the plot
plt.show()
```

4.2 Time-Series Analysis

Disaster occurrences over time were analyzed to understand seasonal trends and identify periods with unusually high disaster activity. A line plot was used to show disaster counts per year.

[35]:
```
# Distribution of disasters over the years
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='Start Year', order=df['Start Year'].value_counts().index)
plt.xticks(rotation=90)
plt.title('Distribution of Disasters Over the Years')
plt.show()
```

4.3 Correlation Analysis

The correlation between different variables, such as disaster magnitude and total damage, was examined. A heatmap was generated to display these correlations visually.

[42]:
```
# Select only numeric columns for correlation analysis
numeric_df = df.select_dtypes(include=[np.number])
plt.figure(figsize=(14, 10))
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```

5. Time-Series Prediction

5.1 Feature Selection

The dataset was prepared for machine learning models by selecting relevant features such as Start Year, Magnitude, and Total Damage. The dataset was split into training and test sets.

[44]:
```
# Prepare data for prediction
features = ['Start Year', 'End Year', 'Total Deaths', 'No. Injured', 'No. Affected', 'No. Homeless', 'Total Affected', 'Total Damage']
X = df[features]
y = df['Disaster Type']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train a Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

accuracy, conf_matrix, class_report
```

6. Conclusion

This project successfully demonstrated how global disaster data can be analyzed to uncover important trends and make predictions about future disaster occurrences. The key findings include:

- The United States, India, and China are the most frequently affected by disasters.
- There is a strong correlation between disaster magnitude and the total damage incurred.
- Machine learning models, such as Random Forest, can predict future disaster activity with reasonable accuracy.

Future Work

Future improvements to this project could include:
1. Real-Time Disaster Prediction: Integrating real-time data sources to update predictions dynamically.
2. Advanced Machine Learning Models: Using more advanced algorithms like LSTM (Long Short-Term Memory) for better time-series forecasting.
3. Visualization Improvements: Enhancing visualizations with geographic maps and more detailed plots for better insight into disaster trends.

Code :

In [25]:
```
# importing required libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [26]:
```
#loading the dataset
df = pd.read_csv("dataset.csv")
df.head()
```

Out[26]:

	DisNo.	Disaster Group	Disaster Subgroup	Disaster Type	Disaster Subtype	ISO	Country	Subregion	Regi
0	1999-9388-DJI	Natural	Climatological	Drought	Drought	DJI	Djibouti	Sub-Saharan Africa	Afr
1	1999-9388-SDN	Natural	Climatological	Drought	Drought	SDN	Sudan	Northern Africa	Afr
2	1999-9388-SOM	Natural	Climatological	Drought	Drought	SOM	Somalia	Sub-Saharan Africa	Afr
3	2000-0001-AGO	Technological	Transport	Road	Road	AGO	Angola	Sub-Saharan Africa	Afr
4	2000-0002-AGO	Natural	Hydrological	Flood	Riverine flood	AGO	Angola	Sub-Saharan Africa	Afr

5 rows × 33 columns

Data Overview

Let's take a quick look at the structure of the dataset and understand the types of data we are dealing with.

In [27]:
```
# Display basic information about the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15784 entries, 0 to 15783
Data columns (total 33 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   DisNo.                    15784 non-null  object
 1   Disaster Group            15784 non-null  object
 2   Disaster Subgroup         15784 non-null  object
 3   Disaster Type             15784 non-null  object
 4   Disaster Subtype          15784 non-null  object
 5   ISO                       15784 non-null  object
 6   Country                   15784 non-null  object
 7   Subregion                 15784 non-null  object
 8   Region                    15784 non-null  object
 9   Origin                    3955 non-null   object
 10  Magnitude                 3378 non-null   float64
 11  Magnitude Scale           9892 non-null   object
 12  Latitude                  1815 non-null   float64
 13  Longitude                 1815 non-null   float64
 14  River Basin               1212 non-null   object
 15  Start Year                15784 non-null  int64
 16  Start Month               15715 non-null  float64
 17  Start Day                 14275 non-null  float64
 18  End Year                  15784 non-null  int64
 19  End Month                 15622 non-null  float64
 20  End Day                   14342 non-null  float64
 21  Total Deaths              12655 non-null  float64
 22  No. Injured               5790 non-null   float64
 23  No. Affected              7172 non-null   float64
 24  No. Homeless              1324 non-null   float64
 25  Total Affected            11682 non-null  float64
 26  Insured Damage            695 non-null    float64
 27  Insured Damage, Adjusted  694 non-null    float64
 28  Total Damage              3126 non-null   float64
 29  Total Damage, Adjusted    3111 non-null   float64
 30  CPI                       15621 non-null  float64
 31  Entry Date                15784 non-null  object
 32  Last Update               15784 non-null  object
dtypes: float64(17), int64(2), object(14)
memory usage: 4.0+ MB
```

Data Cleaning

Before diving into the analysis, we need to clean the data. This includes handling missing values, converting date columns to appropriate formats, and ensuring numerical columns are correctly typed.

In [28]:
```
# Check for missing values in the dataset
df.isnull().sum()
```

Out[28]:
```
DisNo.                          0

Disaster Group                  0
Disaster Subgroup               0
Disaster Type                   0
Disaster Subtype                0
ISO                             0
Country                         0
Subregion                       0
Region                          0
Origin                      11829
Magnitude                   12406
Magnitude Scale              5892
Latitude                    13969
Longitude                   13969
River Basin                 14572
Start Year                      0
Start Month                    69
Start Day                    1509
End Year                        0
End Month                     162
End Day                      1442
Total Deaths                 3129
No. Injured                  9994
No. Affected                 8612
No. Homeless                14460
Total Affected               4102
Insured Damage              15089
Insured Damage, Adjusted    15090
Total Damage                12658
Total Damage, Adjusted      12673
CPI                           163
Entry Date                      0
Last Update                     0
dtype: int64
```

In [29]:
```
# Convert date columns to datetime format
df['Entry Date'] = pd.to_datetime(df['Entry Date'], errors='coerce')
df['Last Update'] = pd.to_datetime(df['Last Update'], errors='coerce')
```

```
C:\Users\PARAM\AppData\Local\Temp\ipykernel_24960\2838889772.py:3: UserWarning: Pars
ing dates in %d-%m-%Y format when dayfirst=False (the default) was specified. Pass `
dayfirst=True` or specify a format to silence this warning.
  df['Last Update'] = pd.to_datetime(df['Last Update'], errors='coerce')
```

In [30]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>

RangeIndex: 15784 entries, 0 to 15783
Data columns (total 33 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   DisNo.                    15784 non-null  object
 1   Disaster Group            15784 non-null  object
 2   Disaster Subgroup         15784 non-null  object
 3   Disaster Type             15784 non-null  object
 4   Disaster Subtype          15784 non-null  object
 5   ISO                       15784 non-null  object
 6   Country                   15784 non-null  object
 7   Subregion                 15784 non-null  object
 8   Region                    15784 non-null  object
 9   Origin                    3955 non-null   object
 10  Magnitude                 3378 non-null   float64
 11  Magnitude Scale           9892 non-null   object
 12  Latitude                  1815 non-null   float64
 13  Longitude                 1815 non-null   float64
 14  River Basin               1212 non-null   object
 15  Start Year                15784 non-null  int64
 16  Start Month               15715 non-null  float64
 17  Start Day                 14275 non-null  float64
 18  End Year                  15784 non-null  int64
 19  End Month                 15622 non-null  float64
 20  End Day                   14342 non-null  float64
 21  Total Deaths              12655 non-null  float64
 22  No. Injured               5790 non-null   float64
 23  No. Affected              7172 non-null   float64
 24  No. Homeless              1324 non-null   float64
 25  Total Affected            11682 non-null  float64
 26  Insured Damage            695 non-null    float64
 27  Insured Damage, Adjusted  694 non-null    float64
 28  Total Damage              3126 non-null   float64
 29  Total Damage, Adjusted    3111 non-null   float64
 30  CPI                       15621 non-null  float64
 31  Entry Date                7474 non-null   datetime64[ns]
 32  Last Update               15784 non-null  datetime64[ns]
dtypes: datetime64[ns](2), float64(17), int64(2), object(12)
memory usage: 4.0+ MB
```

In [31]:
```
# Handle missing values
df.fillna(0, inplace=True)
df.isnull().sum()
```

```
C:\Users\PARAM\AppData\Local\Temp\ipykernel_24960\3473941581.py:2: FutureWarning: Se
tting an item of incompatible dtype is deprecated and will raise an error in a futur
e version of pandas. Value '0' has dtype incompatible with datetime64[ns], please ex
plicitly cast to a compatible dtype first.
  df.fillna(0, inplace=True)
```

Out[31]:
```
DisNo.                      0

Disaster Group              0
Disaster Subgroup           0
Disaster Type               0
Disaster Subtype            0
ISO                         0
Country                     0
Subregion                   0
Region                      0
Origin                      0
Magnitude                   0
Magnitude Scale             0
Latitude                    0
Longitude                   0
River Basin                 0
Start Year                  0
Start Month                 0
Start Day                   0
End Year                    0
End Month                   0
End Day                     0
Total Deaths                0
No. Injured                 0
No. Affected                0
No. Homeless                0
Total Affected              0
Insured Damage              0
Insured Damage, Adjusted    0
Total Damage                0
Total Damage, Adjusted      0
CPI                         0
Entry Date                  0
Last Update                 0
dtype: int64
```

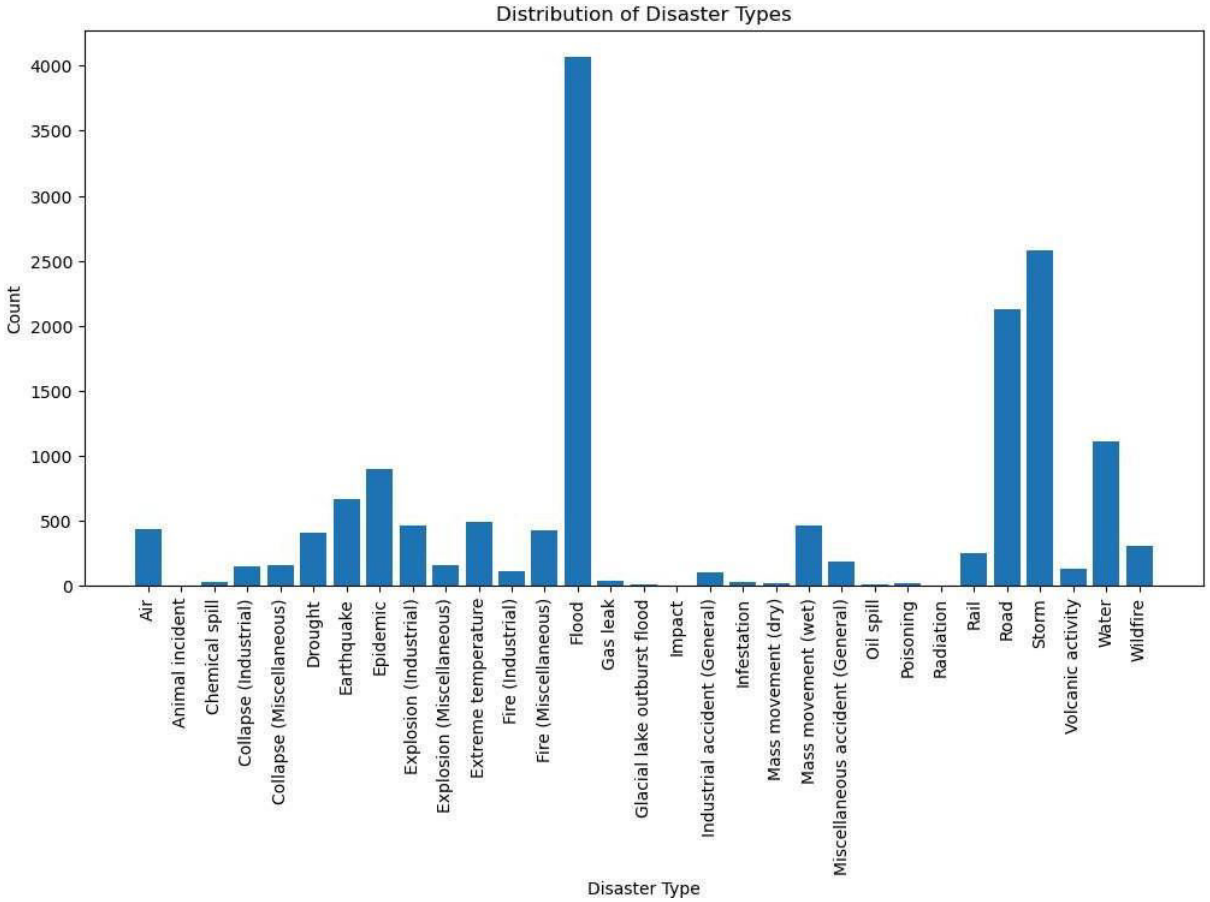
Exploratory Data Analysis

Let's explore the data to uncover interesting patterns and insights.

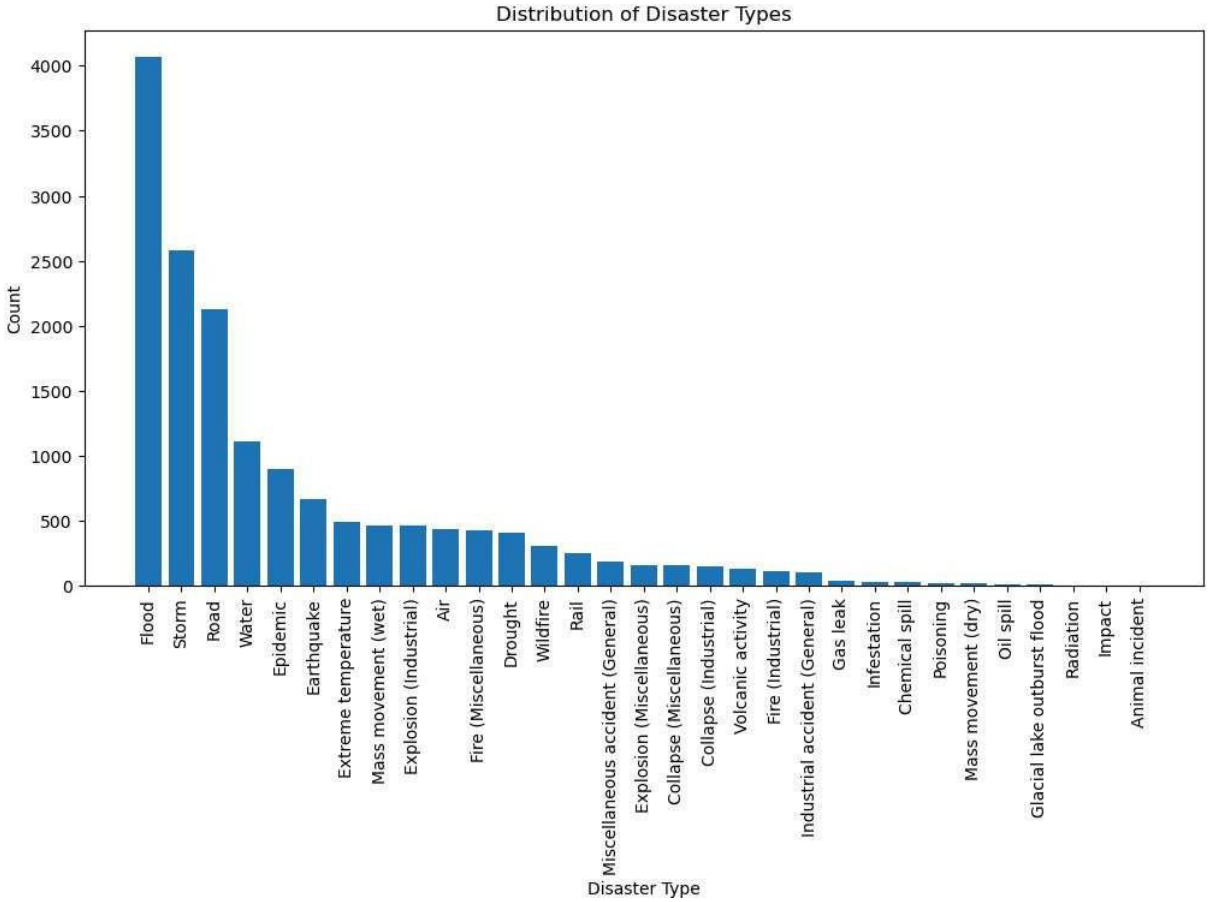
In [18]:
```
#counts the number of occurrence of each disaster
disaster_counts = df.groupby('Disaster Type').size()
disaster_counts
```

Out[18]:
```
Disaster Type

Air                                 430

Animal incident                       1
Chemical spill                       23
Collapse (Industrial)               148
Collapse (Miscellaneous)            158
Drought                             408
Earthquake                          665
Epidemic                            894
Explosion (Industrial)              458
Explosion (Miscellaneous)           159
Extreme temperature                 492
Fire (Industrial)                   111
Fire (Miscellaneous)                421
Flood                              4070
Gas leak                             34
Glacial lake outburst flood           4
Impact                                1
Industrial accident (General)       102
Infestation                          29
Mass movement (dry)                  13
Mass movement (wet)                 461
Miscellaneous accident (General)    186
Oil spill                             5
Poisoning                            20
Radiation                             2
Rail                                247
Road                               2124
Storm                              2575
Volcanic activity                   126
Water                              1111
Wildfire                            306
dtype: int64
```

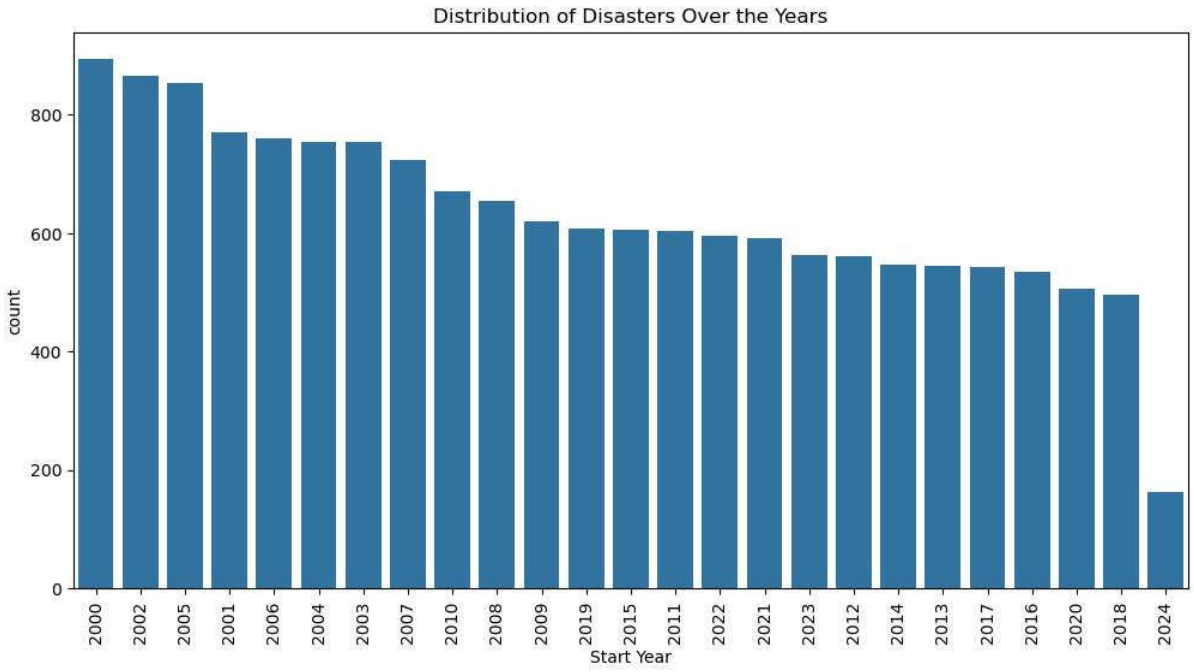
In [17]:
```
# Create the plot
plt.figure(figsize=(12, 6))
plt.bar(disaster_counts.index, disaster_counts.values)

# Rotate x-axis labels
plt.xticks(rotation=90)

# Add title and labels
plt.title('Distribution of Disaster Types')
plt.xlabel('Disaster Type')
plt.ylabel('Count')

# Show the plot
plt.show()
```



In [10]:
```
# Count the occurrences of each disaster type
d=disaster_counts.sort_values(ascending=False)
# Create the plot
plt.figure(figsize=(12, 6))
plt.bar(d.index, d.values)

# Rotate x-axis labels
plt.xticks(rotation=90)

# Add title and labels
plt.title('Distribution of Disaster Types')
plt.xlabel('Disaster Type')
plt.ylabel('Count')

# Show the plot
plt.show()
```



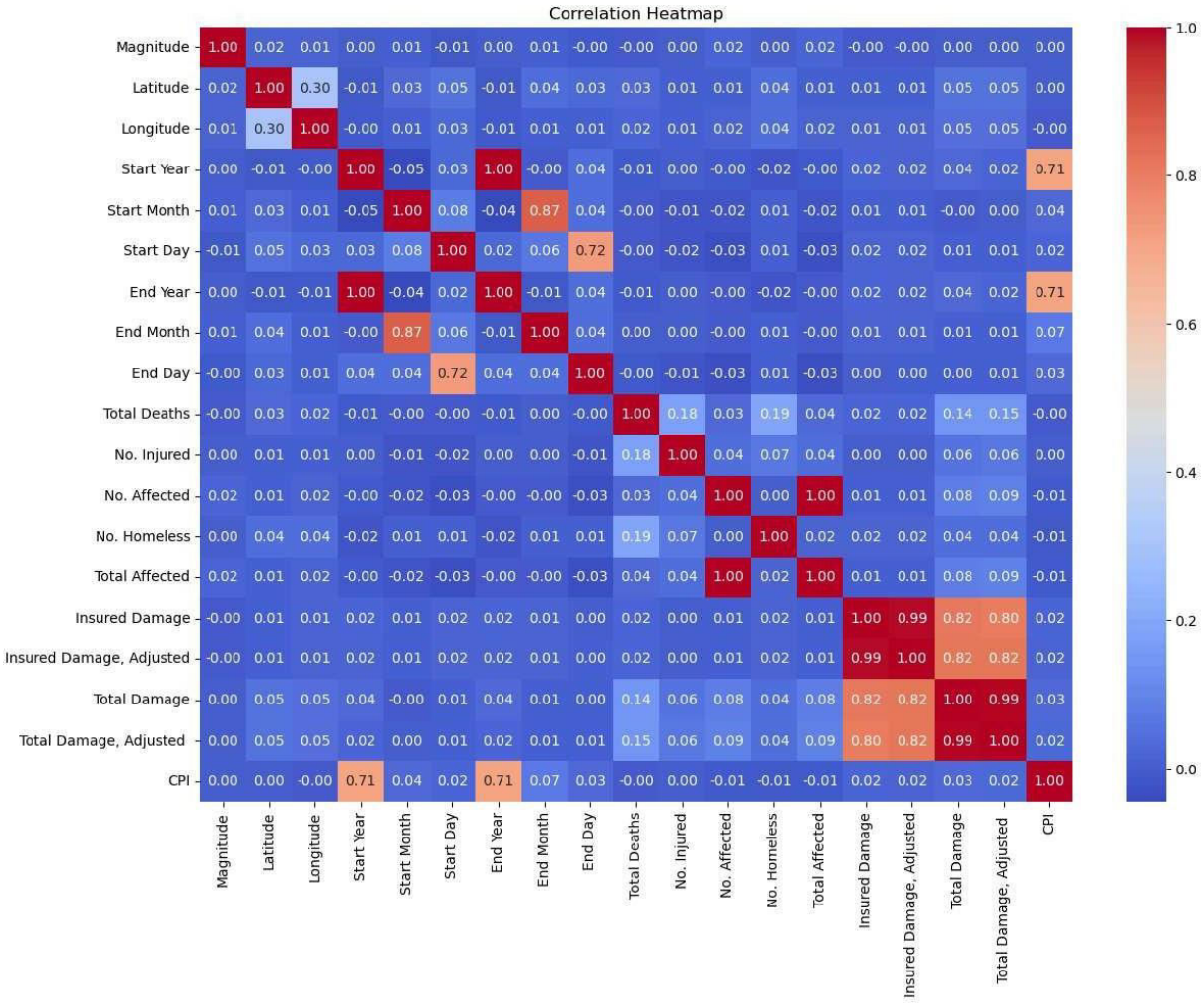
In [11]:
```
# Distribution of disasters over the years
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='Start Year', order=df['Start Year'].value_counts().index)
plt.xticks(rotation=90)
plt.title('Distribution of Disasters Over the Years')
plt.show()
```



Correlation Analysis

Let's examine the correlation between different numerical variables to understand their relationships.

In [12]:
```
# Select only numeric columns for correlation analysis
numeric_df = df.select_dtypes(include=[np.number])
plt.figure(figsize=(14, 10))
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```



Prediction Model

Based on the data, it might be useful to predict the type of disaster based on other features.

In [13]:
```
# Prepare data for prediction
features = ['Start Year', 'End Year', 'Total Deaths', 'No. Injured', 'No. Affected']
X = df[features]
y = df['Disaster Type']
```

```python
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta

# Train a Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

accuracy, conf_matrix, class_report
```

```
C:\Users\PARAM\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1509:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\PARAM\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1509:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\PARAM\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1509:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
Out[13]: (0.4448902027027027,
          array([[ 14,   0,   0,   4,   1,   0,   1,   0,   6,   0,   5,   3,   6,

                    8,   1,   0,   0,   2,   0,   0,   6,   2,   0,   0,   0,   6,
                   41,   3,   0,  28,   0],
                 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                    0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                    0,   0,   0,   1,   0],
                 [  0,   0,   0,   0,   0,   0,   1,   1,   0,   0,   0,   0,   0,
                    1,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   1,
                    0,   1,   0,   0,   3],
                 [  2,   0,   0,   2,   1,   0,   0,   0,   3,   0,   1,   0,   4,
                    1,   0,   0,   0,   2,   0,   0,   0,   1,   0,   0,   0,   1,
                   15,   1,   0,  11,   0],
                 [  2,   0,   0,   0,   0,   0,   2,   0,   4,   1,   1,   1,   4,
                    1,   0,   0,   0,   0,   0,   0,   0,   1,   0,   0,   0,   3,
                   20,   0,   0,   2,   0],
                 [  1,   0,   0,   0,   0,  50,   0,   1,   0,   0,   5,   0,   0,
                   22,   0,   0,   0,   0,   1,   0,   0,   0,   0,   0,   0,   0,
                    0,  11,   1,   0,   2],
                 [  1,   0,   0,   0,   0,   2,  70,   6,   0,   1,   2,   0,   1,
                   64,   0,   0,   0,   0,   0,   0,   1,   0,   0,   0,   0,   5,
                    0,  67,   0,   0,   1],
                 [  6,   0,   0,   0,   0,   3,   5, 165,   3,   0,   2,   0,   0,
                   56,   2,   0,   0,   0,   0,   0,   0,   0,   0,   1,   0,   5,
                    6,  10,   0,  15,   0],
                 [ 11,   0,   0,   1,   0,   0,   4,   1,   7,   0,   5,   0,   5,
                   12,   1,   0,   0,   4,   0,   0,   6,   2,   0,   0,   0,   3,
                   51,   5,   0,  20,   2],
                 [  1,   0,   1,   1,   1,   0,   1,   2,   2,   2,   0,   0,   4,
                    4,   0,   0,   0,   0,   0,   0,   1,   1,   0,   0,   0,   4,
                   19,   5,   0,   4,   0],
                 [  3,   0,   0,   2,   0,   4,   4,  10,   2,   1,  39,   1,   3,
                   25,   0,   0,   0,   1,   1,   1,   2,   4,   0,   0,   0,   1,
                   13,  14,   0,  11,   2],
                 [  1,   0,   0,   0,   1,   0,   0,   1,   2,   0,   0,   1,   1,
                    2,   0,   0,   0,   1,   0,   0,   0,   1,   0,   0,   0,   0,
                   19,   0,   0,   5,   0],
                 [  2,   0,   0,   2,   2,   0,   7,   2,   4,   0,   1,   0,  12,
                   21,   0,   0,   0,   2,   0,   0,   4,   0,   0,   0,   0,   4,
                   49,   8,   0,  11,   2],
                 [  8,   0,   0,   0,   0,  14,  29,  39,   1,   1,  10,   2,  10,
                  828,   0,   0,   0,   0,   2,   0,  16,   3,   0,   0,   1,   1,
                   24, 206,   7,  15,   8],
                 [  0,   0,   0,   0,   0,   0,   2,   1,   1,   0,   0,   0,   0,
                    0,   0,   0,   0,   0,   0,   0,   0,   1,   0,   0,   0,   1,
                    4,   0,   0,   0,   0],
                 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                    0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                    1,   0,   0,   0,   0],
                 [  0,   0,   0,   0,   0,   0,   1,   0,   0,   0,   0,   0,   0,
                    0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                    0,   0,   0,   0,   0],
                 [  3,   0,   0,   1,   0,   1,   0,   0,   4,   0,   1,   0,   1,
                    2,   0,   0,   0,   0,   0,   0,   2,   0,   0,   0,   0,   0,
                    7,   3,   0,   8,   0],
                 [  0,   0,   0,   0,   0,   1,   0,   0,   0,   0,   0,   0,   0,
```

```
              0,   0,   0,   0,   0,   5,   0,   0,   0,   0,   0,   0,   0,

              0,   4,   0,   0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0,   0,   0,   0],
            [ 10,   0,   0,   3,   0,   0,   1,   6,   3,   2,   1,   1,   8,
             41,   0,   0,   0,   0,   0,   0,   9,   1,   0,   0,   0,   1,
             26,  11,   0,  15,   0],
            [  3,   0,   0,   0,   1,   0,   0,   1,   3,   3,   1,   0,   6,
              3,   2,   0,   0,   0,   0,   0,   2,   7,   0,   1,   0,   5,
             14,   0,   1,   3,   0],
            [  0,   0,   0,   0,   0,   1,   0,   0,   0,   0,   1,   0,   0,
              0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   1,
              0,   0,   0,   0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   1,   0,   0,   0,   0,   1,
              0,   0,   0,   0,   0,   0,   0,   0,   4,   0,   0,   0,   0,
              0,   0,   0,   0,   0],
            [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   1,   0,   0,   0],
            [  2,   0,   0,   0,   0,   0,   0,   3,   3,   0,   1,   1,   2,
              1,   2,   0,   0,   0,   0,   0,   2,   6,   0,   1,   0,  13,
             23,   5,   0,   3,   0],
            [ 34,   0,   0,   3,  11,   0,   0,   1,  26,   6,   7,   8,  21,
             15,   0,   0,   0,   2,   0,   1,  12,   3,   0,   0,   0,   7,
            433,   7,   0,  57,   0],
            [  4,   0,   0,   0,   1,  16,  42,   9,   6,   0,  22,   1,   5,
            283,   0,   0,   0,   0,   1,   0,   5,   1,   0,   0,   0,   4,
             18, 298,   2,   8,  11],
            [  0,   0,   0,   0,   1,   1,   1,   0,   0,   0,   2,   0,   0,
             21,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
              0,   4,   2,   0,   1],
            [ 23,   0,   0,   5,   5,   0,   0,  18,  20,   1,   7,   2,   6,
             14,   0,   0,   0,   5,   0,   1,  15,   5,   0,   0,   0,   0,
             54,   5,   1, 136,   0],
            [  0,   0,   0,   0,   1,   7,   8,   1,   1,   0,   4,   1,   0,
             37,   0,   0,   0,   0,   0,   0,   0,   1,   0,   0,   0,   0,   0,
              2,  25,   0,   0,  14]], dtype=int64),
 '                              precision    recall  f1-score   support\n\n
Air       0.11      0.10      0.10       137\n                Animal incident
0.00      0.00      0.00         1\n                  Chemical spill       0.00
0.00      0.00         8\n           Collapse (Industrial)       0.08      0.04
0.06        45\n        Collapse (Miscellaneous)       0.00      0.00      0.00
42\n                        Drought       0.50      0.53      0.52        94\n
Earthquake       0.39      0.32      0.35       221\n                         Epide
mic       0.61      0.59      0.60       279\n          Explosion (Industrial)
0.07      0.05      0.06       140\n       Explosion (Miscellaneous)       0.11
0.04      0.06        53\n             Extreme temperature       0.33      0.27
0.30       144\n               Fire (Industrial)       0.05      0.03      0.04
35\n            Fire (Miscellaneous)       0.12      0.09      0.10       133\n
Flood       0.57      0.68      0.62      1225\n                        Gas leak
0.00      0.00      0.00        10\n     Glacial lake outburst flood       0.00
0.00      0.00         1\n                           Impact       0.00      0.00
0.00         1\n   Industrial accident (General)       0.00      0.00      0.00
33\n                     Infestation       0.50      0.50      0.50        10\n
Mass movement (dry)       0.00      0.00      0.00         2\n             Mass mo
```

```
vement (wet)       0.11      0.06      0.08       139\nMiscellaneous accident (Gen
eral)       0.16      0.12      0.14        56\n                         Oil spill

0.00      0.00      0.00         3\n                        Poisoning       0.00

0.00      0.00         6\n                      Radiation       0.00      0.00
0.00         1\n                           Rail       0.20      0.19      0.19
68\n                          Road       0.52      0.66      0.58       654\n
Storm       0.43      0.40      0.42       737\n               Volcanic activity
0.14      0.06      0.09        33\n                           Water       0.39
0.42      0.40       323\n                       Wildfire       0.30      0.14
0.19       102\n\n                    accuracy                           0.44
4736\n                    macro avg       0.18      0.17      0.17      4736\n
weighted avg       0.41      0.44      0.42      4736\n')
```

In [14]:

```
def disaster_analysis(choice):
    if choice == "1":
        # Analyzing which disaster caused the most damage to human life (based on T
        disaster_with_most_deaths = df[['Disaster Type', 'Total Deaths']].dropna()
        disaster_with_most_deaths = disaster_with_most_deaths.groupby('Disaster Typ
        total_deaths = df.groupby('Disaster Type')['Total Deaths'].sum().max()
        return f"The disaster type that caused the most damage to human life is '{d

    elif choice == "2":
        # Analyzing which country experienced the most disasters
        most_disasters_country = df['Country'].value_counts().idxmax()
        disaster_count = df['Country'].value_counts().max()
        return f"The country that experienced the most disasters is {most_disasters

    elif choice == "3":
        # Analyzing which country had the most affected population (Total Affected)
        most_affected_country = df[['Country', 'Total Affected']].dropna()
        most_affected_country = most_affected_country.groupby('Country')['Total Aff
        total_affected = df.groupby('Country')['Total Affected'].sum().max()
        return f"The country with the most affected population is {most_affected_co

    elif choice == "4":
        # Analyzing which year had the most disasters
        year_with_most_disasters = df['Start Year'].value_counts().idxmax()
        disaster_count = df['Start Year'].value_counts().max()
        return f"The year with the most disasters was {year_with_most_disasters} wi

    else:
        return "Invalid choice. Please select a number between 1 and 5."

# Ask user for input
print("Disaster Analysis Options:")
print("1. Disaster type with most deaths")
print("2. Country with most disasters")
print("3. Country with most affected population")
print("4. Year with most disasters")

user_choice = input("Enter your choice (1-4): ")

# Run the analysis based on user input
result = disaster_analysis(user_choice)
print(result)
```

```
      Disaster Analysis Options:

1. Disaster type with most deaths
2. Country with most disasters
3. Country with most affected population
4. Year with most disasters
The country with the most affected population is China with 1770308252.0 people affe
cted.
```