* ML prediction model:

# Install necessary libraries (if not already installed)

!pip install scikit-learn pandas numpy matplotlib seaborn

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, classification\_report

# Load the dataset (uploaded file)

from google.colab import files

uploaded = files.upload()  # Upload file manually

file\_path = list(uploaded.keys())[0]  # Get uploaded file name

df = pd.read\_csv(file\_path)

# Select important features for analysis

df = df[['Disaster Type', 'Start Year', 'Total Deaths', 'No. Injured', 'No. Affected', 'Total Damage (\'000 US$)']]

# Handle missing values (fill with median for numerical, mode for categorical)

for col in df.columns:

    if df[col].dtype == 'object':  # Categorical column

        df[col] = df[col].fillna(df[col].mode()[0])  # No inplace

    else:  # Numerical column

        df[col] = df[col].fillna(df[col].median())  # No inplace

# Encode categorical target variable (Disaster Type)

label\_encoder = LabelEncoder()

df['Disaster Type'] = label\_encoder.fit\_transform(df['Disaster Type'])

# --- Graph 1: Frequency of disasters over the years ---

plt.figure(figsize=(12, 6))

sns.countplot(x='Start Year', data=df, palette='viridis')

plt.title('Frequency of Disasters Over the Years')

plt.xlabel('Year')

plt.ylabel('Number of Disasters')

plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability

plt.tight\_layout()

plt.show()

# --- Graph 2: Frequency of Each Disaster Type (Which Disasters are More Frequent) ---

plt.figure(figsize=(12, 6))

sns.countplot(y='Disaster Type', data=df, order=df['Disaster Type'].value\_counts().index, palette='coolwarm')

plt.title('Frequency of Each Disaster Type')

plt.xlabel('Number of Occurrences')

plt.ylabel('Disaster Type')

plt.yticks(ticks=range(len(label\_encoder.classes\_)), labels=label\_encoder.classes\_)  # Convert encoded labels to text

plt.tight\_layout()

plt.show()

# --- Graph 3: Total Deaths by Disaster Type ---

plt.figure(figsize=(12, 6))

sns.barplot(x='Disaster Type', y='Total Deaths', data=df, palette='Set2')

plt.title('Total Deaths by Disaster Type')

plt.xlabel('Disaster Type')

plt.ylabel('Total Deaths')

plt.xticks(ticks=range(len(label\_encoder.classes\_)), labels=label\_encoder.classes\_, rotation=45, ha='right')

plt.tight\_layout()

plt.show()

# --- Graph 4: Total Damage by Disaster Type ---

plt.figure(figsize=(12, 6))

sns.barplot(x='Disaster Type', y='Total Damage (\'000 US$)', data=df, palette='Set3')

plt.title('Total Damage by Disaster Type')

plt.xlabel('Disaster Type')

plt.ylabel('Total Damage (\'000 US$)')

plt.xticks(ticks=range(len(label\_encoder.classes\_)), labels=label\_encoder.classes\_, rotation=45, ha='right')

plt.tight\_layout()

plt.show()

# --- Graph 5: Distribution of Total Damage (Histogram) ---

plt.figure(figsize=(12, 6))

sns.histplot(df['Total Damage (\'000 US$)'], kde=False, color='blue', bins=30)

plt.title('Distribution of Total Damage (\'000 US$)')

plt.xlabel('Total Damage (\'000 US$)')

plt.ylabel('Frequency')

plt.tight\_layout()

plt.show()

# Train-test split (80-20) for SVM model

X = df.drop(columns=['Disaster Type'])

y = df['Disaster Type']

# Standardize numerical features

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Train-test split (80-20)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

# Train SVM model

svm\_model = SVC(kernel='rbf', C=1.0, gamma='scale')

svm\_model.fit(X\_train, y\_train)

# Predictions

y\_pred = svm\_model.predict(X\_test)

# Evaluate model

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

# Use unique labels from y\_test for target\_names to avoid mismatch error

target\_names = label\_encoder.classes\_[np.unique(y\_test).astype(int)]  # Ensure target\_names matches unique classes

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred, target\_names=target\_names))

