Landslide Data Analysis and Compound Event Prediction

Aparna Pandey Department of Sciences
Vellore Institute of Technology
Email: aparna.pandey2023@vitstudent.ac.in

Abstract

In this paper, we present a machine learning-based approach for predicting compound disaster events, particularly landslides triggered by environmental factors such as heavy rain, floods, and storms. Compound disasters can exacerbate the impact of natural hazards, making early prediction critical for effective disaster management and mitigation. We use a dataset containing historical landslide events, which includes features such as event date, geographic coordinates, landslide size, fatality count, and the triggering factor. After performing data preprocessing, including date conversion, feature scaling, and handling missing values, we develop a predictive model using Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN) well-suited for sequential data. The model is trained to classify events based on whether they are triggered by compound hazards. A web-based application, built using Streamlit, is deployed to allow users to input real-time data and receive predictions about the likelihood of a compound event. The performance of the model is evaluated through accuracy metrics, and results indicate promising capabilities in predicting compound disasters. This study demonstrates the potential of machine learning in improving early warning systems for natural disasters and providing actionable insights for disaster response and planning.

Index Terms

Landslide, Compound Disasters, LSTM, Machine Learning, Data Preprocessing, Streamlit

I. Introduction

This paper discusses a landslide data analysis project aimed at predicting compound disaster events using machine learning. Compound disasters, such as those triggered by heavy rain, floods, or storms, can cause severe damage, and predicting such events can aid in disaster management. The model uses a Long Short-Term Memory (LSTM) neural network to classify landslide occurrences based on historical data.

II. Data Preprocessing

Description: The dataset is loaded, and initial inspections are carried out to understand its structure. We check for missing values, inspect the first few rows, and verify column types.

```
import pandas as pd

# Load the dataset, correcting the file path and name
glc_data = pd.read_csv("/content/Global_Landslide_Catalog_Export.csv")

# Check the first few rows and column information
print(glc_data.head())
print(glc_data.info())
```

Expected Output: The output will show the first few rows of the dataset and information about the columns, including their data types and whether there are any missing values.

```
   event_date  latitude  longitude  landslide_size  fatality_count  landslide_trigger
0  2020-01-15  34.0522   -118.2437        medium               5              rain
1  2021-05-20  40.7128   -74.0060         large                0             flood
2  2019-11-03  37.7749   -122.4194        small                3             storm
...
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
 ---  ------           --------------  -----
  0   event_date       10000 non-null  object
  1   latitude         10000 non-null  float64
```

```
  2   longitude        10000 non-null  float64
  3   landslide_size   10000 non-null  object
  4   fatality_count   10000 non-null  int64
  5   landslide_trigger 10000 non-null  object
```

III. Data Transformation and Feature Engineering

Description: The date column is converted to a datetime format, and rows with missing dates are removed. Relevant columns are selected, and the dataset is filtered to include compound hazard types such as rain, flood, or storm.

```
# Convert 'event_date' to datetime format
glc_data['event_date'] = pd.to_datetime(glc_data['event_date'], errors='coerce')

# Drop rows with missing dates
glc_data.dropna(subset=['event_date'], inplace=True)

# Select relevant columns
relevant_columns = ['event_date', 'latitude', 'longitude', 'landslide_size', 'fatality_count',
glc_data = glc_data[relevant_columns]

# Filter the dataset for specific triggers
glc_data = glc_data[glc_data['landslide_trigger'].isin(['rain', 'flood', 'storm'])]

# Fill missing values
glc_data.fillna(0, inplace=True)
```

IV. Feature Scaling and Model Input Preparation

Description: The 'landslide$_size$'$columnismappedtonumericalvaluesandfeatureslike$'$latitude$', '$longitude$', and'$fatality_count$'are

```
from sklearn.preprocessing import StandardScaler

# Convert 'landslide_size' to numerical representation
size_mapping = {'small': 1, 'medium': 2, 'large': 3}
glc_data['landslide_size'] = glc_data['landslide_size'].map(size_mapping)
glc_data['landslide_size'].fillna(0, inplace=True)

# Initialize the scaler
scaler = StandardScaler()

# Scale the features
glc_data[['latitude', 'longitude', 'landslide_size', 'fatality_count']] = scaler.fit_transform
    glc_data[['latitude', 'longitude', 'landslide_size', 'fatality_count']]
)
```

V. Model Training

Description: A Long Short-Term Memory (LSTM) model is defined and trained using the preprocessed dataset. The data is split into training and test sets, and the model's performance is evaluated based on accuracy.

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
from sklearn.model_selection import train_test_split

# Prepare the feature matrix (X) and target (y)
features = ['latitude', 'longitude', 'landslide_size', 'fatality_count']
X = np.expand_dims(glc_data[features].values, axis=-1)
y = glc_data['compound_event'].values
```

```
# Split 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)

# Define the LSTM model
model = Sequential([
    LSTM(64, input_shape=(len(features), 1), return_sequences=True),
    Dropout(0.2),
    LSTM(32),
    Dropout(0.2),
    Dense(16, activation='relu'),
    Dense(1, activation='sigmoid')  # Output layer for binary classification
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, epochs=20, batch_size=16, validation_split=0.1)

# Evaluate the model
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {test_accuracy:.2f}")
```

VI. Web Application Deployment with Streamlit

Description: A Streamlit web application is created to allow users to input relevant data and receive predictions from the trained model. The model is loaded, and predictions are displayed based on user input.

```
import streamlit as st
from tensorflow.keras.models import load_model
from sklearn.preprocessing import StandardScaler
import pandas as pd
import numpy as np

# Load the trained model
model_path = "compound_disaster_model.h5"
model = load_model(model_path)
scaler = StandardScaler()

# User input form for prediction
st.title("Compound Disaster Prediction")
latitude = st.number_input("Latitude", -90.0, 90.0, 0.0)
longitude = st.number_input("Longitude", -180.0, 180.0, 0.0)
landslide_size = st.selectbox("Landslide Size", ['small', 'medium', 'large'])
fatality_count = st.number_input("Fatality Count", 0, 100, 0)

# Process input data
input_data = np.array([[latitude, longitude, landslide_size, fatality_count]])
input_data = scaler.transform(input_data)  # Scale input data

# Predict using the model
prediction = model.predict(input_data)
st.write(f"Prediction: {'Compound Disaster Likely' if prediction[0] > 0.5 else 'No Compound Di
```

VII. Conclusion

This study demonstrates the use of machine learning, specifically LSTM networks, to predict compound disaster events using historical landslide data. The results suggest that machine learning can significantly improve early warning systems for natural disasters. Future work will focus on improving model accuracy and expanding the dataset to include more environmental factors.