**Disaster Prediction and Weather App Documentation**

**Overview**

This application integrates disaster prediction with weather data visualization and automated notification sending. It uses machine learning models to predict disaster types based on user inputs and displays weather data on an interactive map. The app is built with Streamlit, using various libraries such as scikit-learn for machine learning, Folium for mapping, and Twilio for sending notifications.

**Application Structure**

The application is divided into three main sections, which are navigated using the sidebar:

1. **Disaster Prediction**
2. **Model Results**
3. **Weather Map**

**1. Disaster Prediction**

**User Interface**

The UI for disaster prediction allows users to enter details about a disaster event, such as:

* Location
* Magnitude
* Total Deaths
* Number Injured
* Number Affected
* Number Homeless
* Disaster Group
* Disaster Subgroup
* Disaster Type

Users can submit these details to predict the disaster subtype using pre-trained machine learning models.

**Model Training**

* **Dataset:** The application loads a dataset from a CSV file (disaster\_data.csv).
* **Preprocessing:** The data is preprocessed by:
  + Selecting relevant features.
  + Encoding categorical variables using one-hot encoding.
  + Filling missing values with the mean of the respective column.
* **Model Training:** Four machine learning models are trained and evaluated:
  + Random Forest Classifier
  + Gradient Boosting Classifier
  + Support Vector Machine (SVM)
  + Logistic Regression
* **Model Selection:** Users can choose from the trained models to predict the disaster subtype.

**Disaster Management Response**

Upon prediction, the app displays suggested actions and management strategies tailored to the predicted disaster type. It also sends notifications to predefined phone numbers using Twilio.

**2. Model Results**

**Data Preview**

This section displays a preview of the dataset used for training the models.

**Model Evaluation Results**

* **Accuracy:** Displays the accuracy of each model.
* **Precision, Recall, F1 Score:** Shows the macro-averaged precision, recall, and F1 score for each model.
* **Suggested Actions:** Provides detailed management strategies for the predicted disaster type.

**3. Weather Map**

**Weather Data Retrieval**

* **API Integration:** Uses the WeatherAPI to retrieve current weather data for a specified location.
* **Location Handling:** Retrieves latitude and longitude of the location using the Geopy library.

**Interactive Map**

* **Folium Map:** Creates an interactive map centered on the specified location, displaying detailed weather information using Folium.
* **Weather Details:** Includes data on temperature, condition, wind speed, humidity, pressure, visibility, UV index, and feels-like temperature.

**Notification System**

The app uses Twilio to send notifications to a list of predefined phone numbers. The notifications include details about the predicted disaster and suggested management strategies.

**Libraries and Tools Used**

* **Streamlit:** For building the web application interface.
* **Pandas:** For data manipulation and analysis.
* **Requests:** For making API calls to retrieve weather data.
* **Folium:** For creating interactive maps.
* **Geopy:** For geocoding locations to obtain latitude and longitude.
* **scikit-learn:** For training and evaluating machine learning models.
* **Twilio:** For sending SMS notifications.
* **Streamlit Folium:** For integrating Folium maps with Streamlit.

## Setting Up the Application

1. **Install Dependencies:**
   * **pip install streamlit pandas requests folium geopy scikit-learn twilio streamlit-folium**
2. **Run the Application:**
   * streamlit run app.py
3. **Provide API Keys and Twilio Credentials:**
   * Replace API\_KEY with your actual WeatherAPI key.
   * Replace account\_sid and auth\_token with your Twilio credentials.

**Conclusion**

This application combines disaster prediction with real-time weather data visualization and automated notifications to provide a comprehensive tool for disaster management and response. It leverages machine learning models for accurate predictions and integrates various libraries for a seamless user experience.

**Importing Libraries and Initial Setup**

The application imports several libraries needed for different functionalities:

* streamlit: For creating the web application interface.
* pandas: For data manipulation and analysis.
* requests: For making API calls to retrieve weather data.
* folium and streamlit\_folium: For creating and displaying interactive maps.
* geopy: For geocoding locations to obtain latitude and longitude.
* scikit-learn: For training and evaluating machine learning models.
* twilio: For sending SMS notifications.

**Twilio Notification Function**

The send\_notification function sends SMS notifications using the Twilio API:

def send\_notification(message\_body, to\_phone\_number):

try:

message = client.messages.create(

body=message\_body,

from\_=from\_phone\_number,

to=to\_phone\_number

)

st.success("Notification sent successfully.")

except Exception as e:

st.error(f"Failed to send notification: {e}")

**Weather Data Functions**

* get\_weather\_data(location): Fetches current weather data for a given location using the WeatherAPI.
* get\_lat\_lon(location): Retrieves the latitude and longitude of a location using the Geopy library.
* create\_weather\_map(location, weather\_data): Creates an interactive map centered on the specified location and displays detailed weather information.

**Preprocessing and Model Training**

* preprocess\_data(data, target\_variable, features): Preprocesses the dataset by selecting relevant features, encoding categorical variables, and filling missing values.
* train\_and\_evaluate\_models(X\_train, X\_test, y\_train, y\_test): Trains and evaluates multiple machine learning models, returning the results for comparison.

**User Input Preprocessing**

* preprocess\_user\_input(user\_input, X\_columns): Preprocesses user input to match the format of the training data.

**Main Application Logic**

The application is divided into three main sections, which are navigated using the sidebar:

1. **Disaster Prediction**
2. **Model Results**
3. **Weather Map**

Each section handles different parts of the functionality, as described earlier.

**Running the Application**

To run the application, install the necessary dependencies, replace the placeholder API keys and credentials with your actual keys, and execute the following command:

**streamlit run app.py**