SOMAIYA VIDYAVIHAR UNIVERSITY

K J Somaiya College of Engineering

Somaiya TRUST

WEATHER TRACKING AND FORECASTING

FY IT Sem II (2022-23)

Batch P6-1

Prachi Gandhi - 16010422233

Chandana Galgali - 16010422234

Mahek Thakkar - 16010422235

Problem Statement

The goal of this mini project is to develop a weather forecasting system that predicts the probabilities of various weather events based on the current weather conditions of a specified city.

SOMAIYA VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering

Somaiya TRUST

System Architecture

01

Step one

Retrieve current weather data of the city inputted by the user using WeatherStack API.

02

Step two

Process the weather data to extract relevant information (temperature, humidity, weather code).

03

Step three

Calculate the probabilities of rain, sleet, snow, thunderstorms, cyclones, blizzards, hailstorms and fog based on the extracted information.

04

Step four

Display the calculated probabilities to the user.

Implementation Details

"WEATHER TRACKING AND FORECASTING PROGRAM"

```
#Importing the necessary libraries

import requests #A library for making HTTP requests
import json #A library for working with JSON data
import math #A library for mathematical operations

#To retrieve current weather data for a specified city

def get_current_weather(api_key, city): #The function takes 2 arguments: the API key and the city

    #Setting a variable to the URL of the Weatherstack API's current weather endpoint
    base_url = "http://api.weatherstack.com/current"

    #Creating a dictionary that contains the API key and the city name as query parameters
    params = {
        "access_key": api_key,
        "query": city,
    }

    #Making an HTTP GET request to the Weatherstack API's current weather endpoint then passing the URL and the query
parameters as arguments
    response = requests.get(base_url, params=params)

    #Returning the JSON data from the API response
    return response.json()

api_key = "f9374dfb90b487d7f7e273755528716e"
#Prompting the user to enter the name of the city for which they want to retrieve the weather data
city = input("Find the weather forecast of (enter city name): ")

#Calling the function and passing in the API key and the city name as arguments
current_weather = get_current_weather(api_key, city)

print(json.dumps(current_weather, indent=1))
```

Implementation Details


```
#To scale the probability percentage in the range 0-100
def sigmoid(x):
    return 1 / (1 + math.exp(-x))

#To calculate the probabilities of various weather events such as rain, sleet, snow, thunderstorms, cyclones, blizzards, hail, and fog

def forecast_probabilities(current_weather): #The function takes the JSON object as an argument

    #Extracting the temperature, humidity, and weather code from the JSON object
    temperature = current_weather["current"]["temperature"]
    humidity = current_weather["current"]["humidity"]
    weather_code = current_weather["current"]["weather_code"]

    #Initializing the probabilities of various weather events to 0
    rain_probability = 0
    sleet_probability = 0
    snow_probability = 0
    thunderstorm_probability = 0
    cyclone_probability = 0
    blizzard_probability = 0
    hail_probability = 0
    fog_probability = 0
```

Implementation Details


```
if weather_code in [113]: #Clear/Sunny
        rain_probability = humidity * 0
    elif weather_code in [116, 119, 122, 143]: #Partly Cloudy/Cloudy/Overcast/Mist
        rain_probability = humidity * 0.1
    elif weather_code in [176, 263, 266, 293, 296, 353, 386]: #Light Rainfall
        rain_probability = humidity * 0.25
    elif weather_code in [299, 302, 356, 389]: #Moderate Rainfall
        rain_probability = humidity * 0.50
    elif weather_code in [305, 308, 359]: #Heavy Rainfall
        rain_probability = humidity * 0.75

    rain_probability = sigmoid(rain_probability)

    if weather_code in [182, 185, 281, 311, 317, 362, 374]: #Light Sleet showers
        sleet_probability = humidity * 0.25
    elif weather_code in [284, 314, 320, 350, 365, 377]: #Moderate to Heavy Sleet Showers
        sleet_probability = humidity * 0.75

    sleet_probability = sigmoid(sleet_probability)

    if weather_code in [179, 227, 323, 326, 392]: #Light Snowfall
        snow_probability = humidity * 0.25
    elif weather_code in [329, 332, 368]: #Moderate Snowfall
        snow_probability = humidity * 0.50
    elif weather_code in [335, 338, 371, 395]: #Heavy Snowfall
        snow_probability = humidity * 0.75

    snow_probability = sigmoid(snow_probability)
```

Implementation Details


```
if weather_code in [200, 386, 389, 392, 395]:
        thunderstorm_probability = 0.5

    if weather_code in [362, 365, 368, 371, 374, 377]:
        cyclone_probability = 0.5

    if weather_code in [230]:
        blizzard_probability = 1

    if temperature <= 0 and humidity >= 80:
        hail_probability = 0.5

    if weather_code in [248, 260]:
        fog_probability = 1

    #Returning the calculated probabilities
    return rain_probability, sleet_probability, snow_probability, thunderstorm_probability, cyclone_probability, blizzard_probability,
hail_probability, fog_probability

#Calling the forecast_probabilities function and printing the probabilities of the weather events:
rain_prob, sleet_prob, snow_prob, thunderstorm_prob, cyclone_prob, blizzard_prob, hail_prob, fog_prob =
forecast_probabilities(current_weather)
print(f"Rainfall probability: {rain_prob * 100:.2f}%")
print(f"Sleet-shower probability: {sleet_prob * 100:.2f}%")
print(f"Snow probability: {snow_prob * 100:.2f}%")
print(f"Thunderstorm probability: {thunderstorm_prob * 100:.2f}%")
print(f"Cyclone probability: {cyclone_prob * 100:.2f}%")
print(f"Blizzard probability: {blizzard_prob * 100:.2f}%")
print(f"Hail probability: {hail_prob * 100:.2f}%")
print(f"Fog probability: {fog_prob * 100:.2f}%")
```

Features of Designed System

- Fetches real-time weather data using WeatherStack API.
- Calculates the probabilities of different weather events (for e.g. rainfall, sleet-showers, snowfall, thunderstorms, cyclones, blizzards, hailstorms, fog, etc.)
- Supports forecasting for any city supported by the WeatherStack API.
- Easy to integrate with other applications or services.

Output Snapshots

For Mumbai:


```
{
 "request": {
  "type": "City",
  "query": "Mumbai, India",
  "language": "en",
  "unit": "m"
 },
 "location": {
  "name": "Mumbai",
  "country": "India",
  "region": "Maharashtra",
  "lat": "18.975",
  "lon": "72.826",
  "timezone_id": "Asia/Kolkata",
  "localtime": "2023-05-29 21:07",
  "localtime_epoch": 1685394420,
  "utc_offset": "5.50"
 },
 "current": {
  "observation_time": "03:37 PM",
  "temperature": 31,
  "weather_code": 143,
  "weather_icons": [
   "https://cdn.worldweatheronline.com/images/wsymbols01_png_64/wsymbol_0006_mist.png"
  ],
```

Output Snapshots


```
  "weather_descriptions": [
   "Haze"
  ],
  "wind_speed": 15,
  "wind_degree": 300,
  "wind_dir": "WNW",
  "pressure": 1010,
  "precip": 0,
  "humidity": 75,
  "cloudcover": 50,
  "feelslike": 39,
  "uv_index": 1,
  "visibility": 4,
  "is_day": "no"
 }
}
Rainfall probability: 99.94%
Sleet-shower probability: 50.00%
Snow probability: 50.00%
Thunderstorm probability: 0.00%
Cyclone probability: 0.00%
Blizzard probability: 0.00%
Hail probability: 0.00%
Fog probability: 0.00%
```

Output Snapshots

For New York:


```
{
 "request": {
  "type": "City",
  "query": "New York, United States of America",
  "language": "en",
  "unit": "m"
 },
 "location": {
  "name": "New York",
  "country": "United States of America",
  "region": "New York",
  "lat": "40.714",
  "lon": "-74.006",
  "timezone_id": "America/New_York",
  "localtime": "2023-05-29 11:43",
  "localtime_epoch": 1685360580,
  "utc_offset": "-4.0"
 },
 "current": {
  "observation_time": "03:43 PM",
  "temperature": 21,
  "weather_code": 113,
  "weather_icons": [
   "https://cdn.worldweatheronline.com/images/wsymbols01_png_64/wsymbol_0001_sunny.png"
  ],
```

Output Snapshots


```
  "weather_descriptions": [
   "Sunny"
  ],
  "wind_speed": 15,
  "wind_degree": 70,
  "wind_dir": "ENE",
  "pressure": 1018,
  "precip": 0,
  "humidity": 46,
  "cloudcover": 0,
  "feelslike": 21,
  "uv_index": 6,
  "visibility": 16,
  "is_day": "yes"
 }
}
Rainfall probability: 50.00%
Sleet-shower probability: 50.00%
Snow probability: 50.00%
Thunderstorm probability: 0.00%
Cyclone probability: 0.00%
Blizzard probability: 0.00%
Hail probability: 0.00%
Fog probability: 0.00%
```

Results

The system first displays the current weather conditions of the city inputted by the user. It then successfully calculates and displays the probabilities of rainfall, sleet-showers, snowfall, thunderstorms, cyclones, blizzards, hailstorms and fog for the specified city based on the current weather conditions such as temperature, humidity and the weather code.

Conclusion

The developed weather forecasting system provides a simple and effective way to predict the likelihood of various weather events. It can be further improved by incorporating more advanced algorithms and additional data sources to increase the accuracy of the predictions.

References

- WeatherStack API Documentation: https://weatherstack.com/documentation
- Sigmoid Function: https://en.wikipedia.org/wiki/Sigmoid_function
- Weather Codes: https://weatherstack.com/documentation#weather_codes

THANK YOU