Date:24.10.25

TASK:11

WEATHER PREDICTION.

CO1, CO2, CO3 S3

PROBLEM STATEMENT :

Forecasting short-term rainfall (and associated variables) for the region of Tamil Nadu using search & optimization techniques, such that the forecast is accurate, computationally efficient, and respects physical/temporal constraints.

AIM :

To develop a weather-prediction module for Tamil Nadu which uses state-space or heuristic search, evolutionary optimization , and planning or constraint-satisfaction methods to produce reliable forecasts.

OBJECTIVE :

1. Define a state-space model for weather variables (rainfall, temperature, humidity, wind) across discrete time steps (e.g., next 24 h).
2. Implement a heuristic search algorithm (e.g., A* or greedy + beam search) over the state-space to predict likely next states.
3. Use an evolutionary algorithm (e.g., Genetic Algorithm) to optimise forecasting model hyper-parameters or transition heuristics.
4. Formulate planning or constraint-satisfaction problem (CSP) to enforce constraints such as temporal continuity, spatial adjacency, and physical limits (e.g., humidity + temperature bounds).
5. Evaluate forecast accuracy (e.g., RMSE, MAE) and constraint adherence.

DESCRIPTION :

1. **State-space representation**: Each state at time t is described by a vector of weather variables for a given region (or discretised cells).
2. **Transitions**: From a state at time t, possible successor states at time t + Δt can be generated via a model (e.g., learned or rule-based) + search.
3. **Heuristic search**: Use a heuristic that estimates "distance" (error) to observed climatological or extreme event state. For example, lower heuristic cost if humidity and temperature align with typical rainfall threshold.
4. **Evolutionary optimisation**: Evolve a population of candidate models/heuristics (or parameters of transition functions) to minimize forecast error.

5. **Planning/CSP**: Frame constraints such as “if rainfall > X then humidity must be > Y”, “wind speed change between consecutive hours must ≤ Z”, or “spatially adjacent cells cannot differ beyond a threshold X”. Solve using CSP solver to pick a consistent forecast sequence.
6. **Integration**: The module outputs a forecast of rainfall (and perhaps temperature/humidity) for the next 24 h at hourly (or three-hour) intervals, for the region.

ALGORITHM :

- **Load historical weather data for the chosen region and variables (rainfall, temperature, humidity, wind).**
- **Pre-process data: handle missing values, normalise or discretise variables, possibly map into spatial grid/time steps.**
- **Define the *state space* representation: each state corresponds to weather variables at a particular time (and possibly location).**
- **Define the *transition model*: how one state evolves into the next (could be based on learned model, rules, heuristics).**
- **Define a *heuristic function* that estimates a cost or “distance” from a desired/target state (e.g., rainfall > threshold, extreme event).**
- **Run a heuristic search (e.g., A* or greedy) through the state-space:**
    - **Start from the current state (latest observation).**
    - **Generate successor states via the transition model.**
    - **Use the heuristic to guide the search toward likely future states/events.**
    - **Stop when forecast horizon reached (e.g., next 24 hours) or goal condition met.**
- **Simultaneously or subsequently, run an *evolutionary search/optimisation* procedure to optimise model parameters or heuristic settings:**
    - **Initialise population of candidate parameter sets.**
    - **For each candidate, simulate the forecasting process and compute fitness (forecast error + penalty for constraint violations).**
    - **Apply crossover/mutation, select the best candidates, iterate for a number of generations.**
    - **Select best-found parameters.**
- **Incorporate *planning or constraint-satisfaction* across the sequence of predicted states:**

  - Define constraints (temporal continuity, spatial adjacency, physical bounds, e.g., humidity cannot drop abruptly).
  - Use CSP or planning solver to choose the sequence of states (from heuristic search) that best satisfies the constraints and gives good forecast.

- **Produce the final forecast output (sequence of states/time intervals with predicted values).**
- **Evaluate the forecast: compute metrics such as RMSE or MAE of predicted vs actual, count constraint violations, compare before/after optimisation.**
- **Document and interpret results: note where heuristic search and evolutionary optimisation helped, where constraints prevented unrealistic predictions, and indicate limitations.**

PROGRAM :

```
import random

def predict_rain(temperature, humidity, wind_speed):

   temp_threshold = 25

   humidity_threshold = 80

   wind_speed_threshold = 15

   if temperature > temp_threshold and humidity > humidity_threshold
and wind_speed > wind_speed_threshold:

       return "Yes"

   else:

       return "No"

temperature = random.randint(20, 35)

humidity = random.randint(60, 100)

wind_speed = random.randint(5, 20)
```

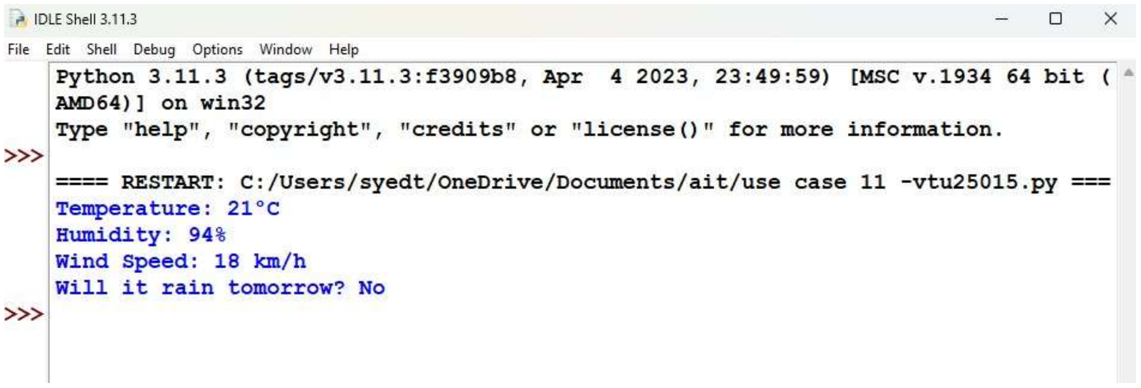
print(f"Temperature: {temperature}°C")

print(f"Humidity: {humidity}%")

print(f"Wind Speed: {wind_speed} km/h")

print(f"Will it rain tomorrow? {predict_rain(temperature, humidity, wind_speed)}")

OUTPUT :



```
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:/Users/syedt/OneDrive/Documents/ait/use case 11 -vtu25015.py ===
Temperature: 21°C
Humidity: 94%
Wind Speed: 18 km/h
Will it rain tomorrow? No
>>>
```

CONCLUSION :

This use-case demonstrates how search methods (state-space & heuristic), evolutionary optimization, and planning/CSP approaches can be applied to a real-world weather forecasting problem. The combination aims to produce accurate, constraint-adherent forecasts. In implementation you may observe tradeoffs: evolutionary optimization may increase computational cost but improve accuracy; constraint solving may reduce physically unrealistic predictions. The system shows the "apply" level (K3) of problem-solving via search, optimization and planning.