**Hail prediction**

**Hail Prediction Project Report**

**Scope**: Desktop prototype for near-real-time hail threat visualization around Islamabad using Meteosat-9 imagery. Includes GUI, periodic data retrieval, synthetic fallback, feature engineering, probabilistic hail estimation, and map rendering.

**System Overview**

* Entry script: test.py implements HailPredictionSystem with a Tkinter GUI and Matplotlib map embedded.
* Data sources attempted (in order):
  + http://203.135.4.150:3333/images/HAIL/
  + .../2025-05-07/, .../2025-05-27/, .../2024-04-17/ (historic hail/active systems)
* Update cadence: every 15 minutes (or on manual refresh).
* Geographic focus: Islamabad (33.6844, 73.0479), 100 km radius.

**Data Layer**

* fetch\_real\_data() scrapes directory listings to find first available .bmp or .webm asset.
* For .webm, saves to temp file and decodes first frame with OpenCV; for images, decodes into ndarray.
* If all sources fail, system switches to synthetic data mode.

**Processing Pipeline**

* process\_data() routes either real image arrays or generates synthetic meteorological fields.
* Real image path:
  + Convert to grayscale; resize to 100×100 grid.
  + Build lat/lon grids around Islamabad.
  + Derive fields:
    - Cloud-top temperature: base 240–260K minus image-intensity-influenced cooling with radial decay.
    - Brightness temperature differences: edges/gradients mapped to btd\_39\_108, btd\_108\_120.
  + Pack into xarray.Dataset with lat/lon coords.
* Synthetic path:
  + Generates physically inspired fields on a 100×100 grid for testing and demos.

**Prediction Logic**

* Placeholder model: RandomForestClassifier instance as a stub for future training.
* Current prototype computes pseudo-probabilities:
  + Sample-wise uniform random probabilities, enhanced quadratically near Islamabad based on distance weighting.
  + Produces hail\_probability (0–1) and size proxy hail\_size (mm) as xarray.Dataset.
* Islamabad area stats:
  + Extracts max probability and size within 100 km; displays in GUI and toggles alert text color when above threshold (30%).

**Visualization**

* Tkinter window with embedded Matplotlib plot in PlateCarree projection via Cartopy.
* Overlays coastlines, borders, rivers, lakes.
* Renders hail probability heatmap (0–100%) and marks Islamabad point with a 100 km dashed circle.
* Status bar shows last update time and data source type (REAL/SYNTHETIC).

**Auxiliary Tooling**

* data\_ing\_test.py: directory listing debugger for a fixed date (TEST\_DATE=2025-06-20). Parses image types, extracts latest timestamps per channel, tests sample download, and verifies image payload.
* HAIL\_pred.ipynb: exploratory notebook (not analyzed here).
* HAIL/ contains working documents and drafts; vcpkg/ is a third-party package manager tree not tied to the Python prototype.

**How to Run**

1. Ensure Python dependencies for GUI and geospatial stack are installed (tkinter, matplotlib, cartopy, xarray, requests, pillow, opencv-python-headless).
2. Run: python test.py.
3. Click "Manual Refresh" or wait for periodic updates.

**Notes and Decisions**

* Prioritized robust data retrieval with graceful fallback to synthetic mode to keep the UI responsive during outages.
* Used distance-weighted enhancement to emphasize local hail risk for Islamabad as a prototype heuristic.
* Deferred audio alarms and model persistence; left hooks in place for future enablement.

**Next Steps (Optional)**

* Replace heuristic probabilities with a trained classifier/regressor on labeled hail events.
* Ingest multi-channel satellite data and engineered predictors (e.g., CAPE proxies, texture metrics).
* Persist recent frames for temporal inference and short-term nowcasting.
* Package app as an executable for operational use; add logging and error telemetry.