Anomaly Detection and Prediction in Weather

Funny Mud Pee
Ziyuan(Peter) Ye, Jiayi Deng, Yu-Chih (Wisdom) Chen

Agenda

- ➢ Problem Statement
- ➢ Project Purpose
- ➢ Methodology
- ➢ Result
- ➢ Modeling Results
- ➢ Conclusions & Recommendations

Problem Statement

- Context: Sudden and severe changes in weather conditions often greatly affect sectors like agriculture, cultivation, transportation, etc.
- Goal: Design a system to detect and analyze anomalies in weather data to predict sudden and severe changes in weather conditions.

Project Purpose

- Anomaly Detection

  The system will utilize statistical models to identify inconsistencies and abnormalities in weather data, including parameters like temperature, relative humidity, and dewpoint.

- Real-Time Visualization:

  Our system will feature a user-friendly interface displaying real-time weather data and anomaly threshold. Data points exceeding these thresholds are identified as anomalies.

- Autocorrection Over Time:

  The system is designed to refine and optimize its performance over time using an Exponential Decay Weighted Score mechanism.

Project Purpose - Data Set Description

- Location
- Coordinates (Latitude & Longitude)
- Summary (Temperature summary)
- Temperature
- Units (Temperature Units → F)
- Daytime (False → Night)
- Dewpoint (Temperature to which air must be cooled to become saturated with water vapor)
- Probability of Precipitation (Likelihood of precipitation occurring at a specific location)
- Relative Humidity (Measure of the amount of moisture)
- Wind Speed (Speed of the Wind → mph)
- Start (Weather changes of the start time)
- End (Weather changes of the end time)
- End - Start → Time windows feature (Windows Size → 3)

Methodology - Tools

01 Backend

- pyensign
- pandas
- numpy
- River
- exponential decay weighting

02 Frontend

- html
- javascript
- plotly

03 Connection

- pyensign
- Flask
- socketio

Methodology - Framework

Weather Publisher

1. Insert Latitude and Longitude
2. Ping API from NOAA
3. Fetch future nearly 7*24 weather data
4. Subtract useful features
5. Publish each data to Weather topic

Weather Subscriber / Score Publisher

1. Subscribe to Weather topic
2. Feature engineering and transformation
3. Train the Model
4. Get an anomaly score
5. Publish each original weather data plus the score to Score topic

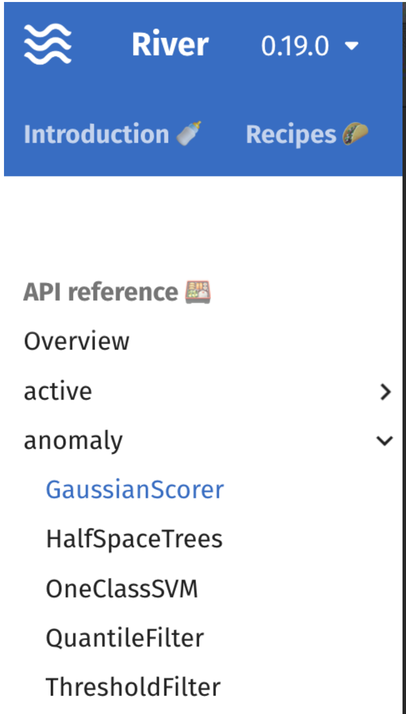
Score Subscriber / Weighted Score Publisher

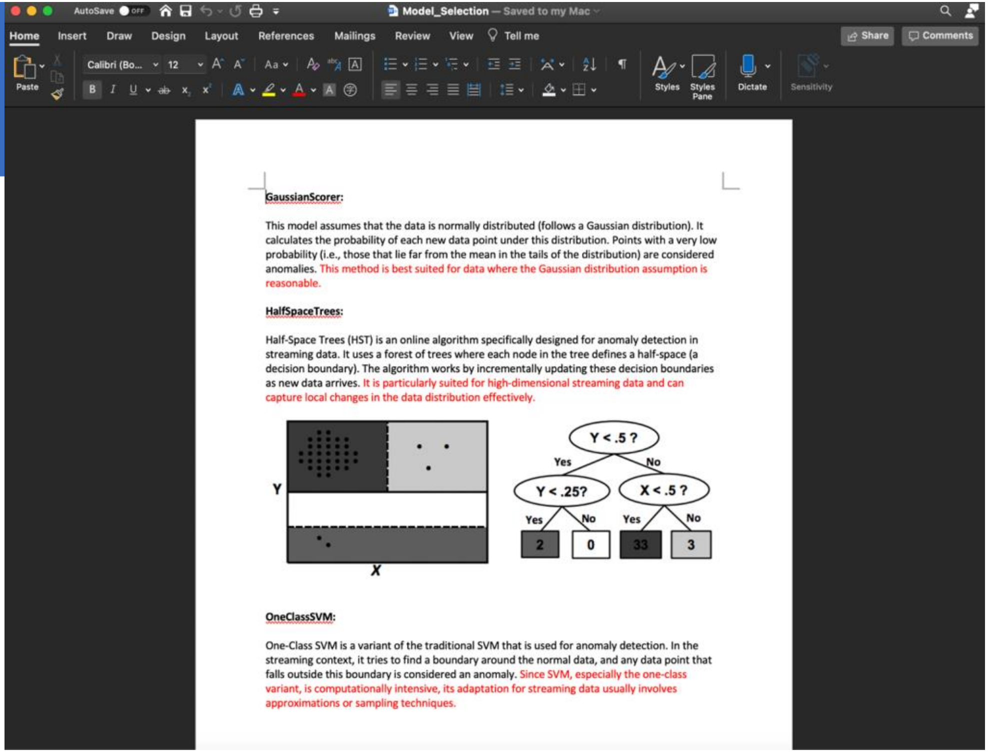
1. Subscribe to Score topic
2. Store the score in cache
3. Retrieve the cache and weighting each time point scores
4. Publish each original weather data plus the weighted score to weighted score topic

Weighted Score Publisher

1. Subscribe to Weighted score topic
2. Emit the original weather data and weighted score to dashboard
3. Check whether the time point exists, if it exists, update it with the new one

Methodology - Model Justification





GaussianScorer:

This model assumes that the data is normally distributed (follows a Gaussian distribution). It calculates the probability of each new data point under this distribution. Points with a very low probability (i.e., those that lie far from the mean in the tails of the distribution) are considered anomalies. This method is best suited for data where the Gaussian distribution assumption is reasonable.

HalfSpaceTrees:

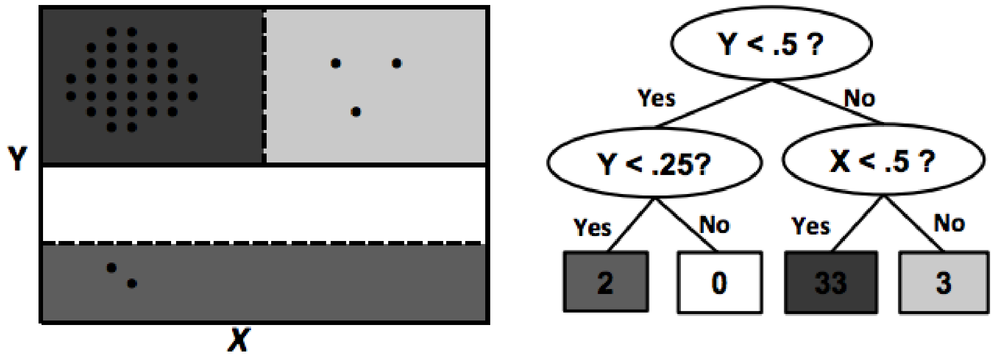
Half-Space Trees (HST) is an online algorithm specifically designed for anomaly detection in streaming data. It uses a forest of trees where each node in the tree defines a half-space (a decision boundary). The algorithm works by incrementally updating these decision boundaries as new data arrives. It is particularly suited for high-dimensional streaming data and can capture local changes in the data distribution effectively.

OneClassSVM:

One-Class SVM is a variant of the traditional SVM that is used for anomaly detection. In the streaming context, it tries to find a boundary around the normal data, and any data point that falls outside this boundary is considered an anomaly. Since SVM, especially the one-class variant, is computationally intensive, its adaptation for streaming data usually involves approximations or sampling techniques.

Half Spcace Trees

Methodology - Model Description



1. Half Space Tree
   - Used for anomaly detection in data streams
   - Partition the data space into several windows and make predictions based on the number of data points in each window
   - Depth h is a full binary tree consisting of 2^(h+1) - 1 nodes, in which all leaves are at the same depth, h

Methodology - Feature Engineering and Transformation

- Feature Transformation
  - Conversion of 'daytime' from boolean to numerical: True/False converted to 1/0.
  - Conversion of 'windSpeed' to numerical: Extracts the numerical part from a string representation.
- Feature Engineering
  - LDA (Latent Dirichlet Allocation) on 'summary' field : The 'summary' column, which is likely a textual description, is transformed into two numerical components using LDA.
  - Min-Max Scaling: Scales the features based on Min-Max scaling, ensuring that the final features are in the range of [0,1]
  - **Weighted Anomaly Score Calculation:** model produces anomaly scores for each hour over a forecasted period of 7 days (168 hours).Need to be combined into a single score, especially with a time-based weightage (more recent scores are more meaningful because more recent weather forecasts are more precisely empirically.

Formula for exponential decay:

$$w(t) = e^{-\lambda x}$$

w(t) is the weight at time
λ is the decay rate (a positive value, higher values mean faster decay).
t is the time since the score (in hours)

Formula for weighed Average:

$$\text{Weighetd Average Score} = \frac{\sum_{i=1}^{n} w_i * s_i}{\sum_{i=1}^{n} w_i}$$

w_i is the weight for the score at time
s_i is the score at time i,
n is the total number of scores.

Methodology - Model Engineering

```
def reset_all_process(self):
    print('Reset model')
    self.LDA = compose.Pipeline(
        *steps: (feature_extraction.BagOfWords()),
        (preprocessing.LDA(n_components=2, number_of_documents=168))
    )
    self.scaler = preprocessing.MinMaxScaler()
    self.hst = anomaly.HalfSpaceTrees(n_trees=25, height=6, window_size=2)
```

More trees = More opinions on what's normal and what's not.
A taller tree can look at data in more details but will make score sensitive.
Window_size:Number of recent weather data to decide if the new one is weird

For Hawaii

n_trees, height, and window_size

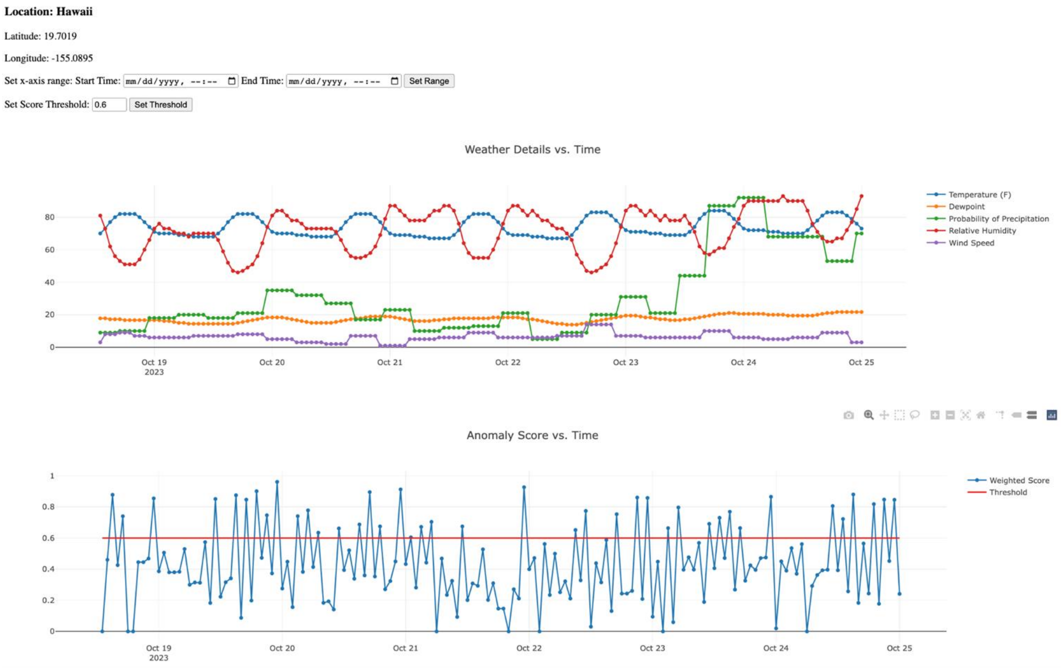
The parameters should depend on the weather data!
The weather pattern of the location

Methodology - Validation Methods

Dashboard Example

Since the weather anomaly detection is unsupervised, one of the effective way to validate is using graph. We use our frontend dashboard to display streaming weather details and score over time.

We scrutinizes each anomaly point and compare it with weather details to judge

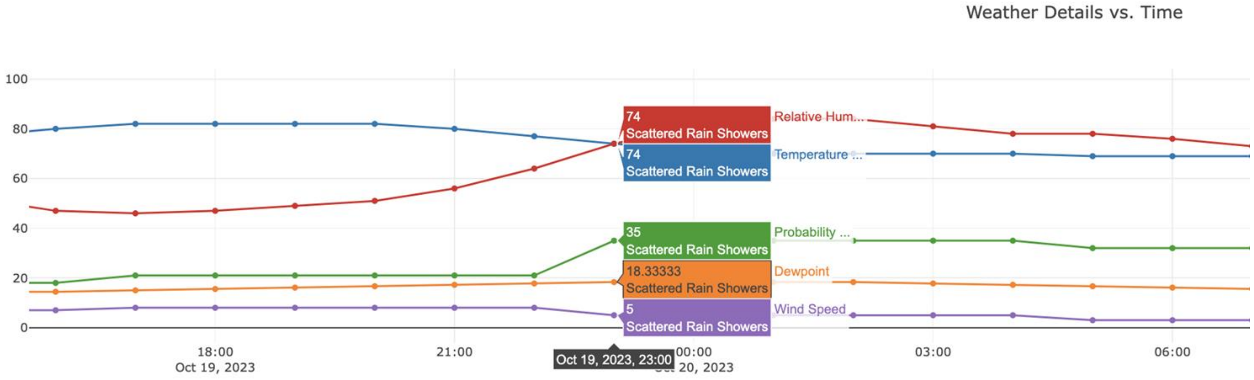


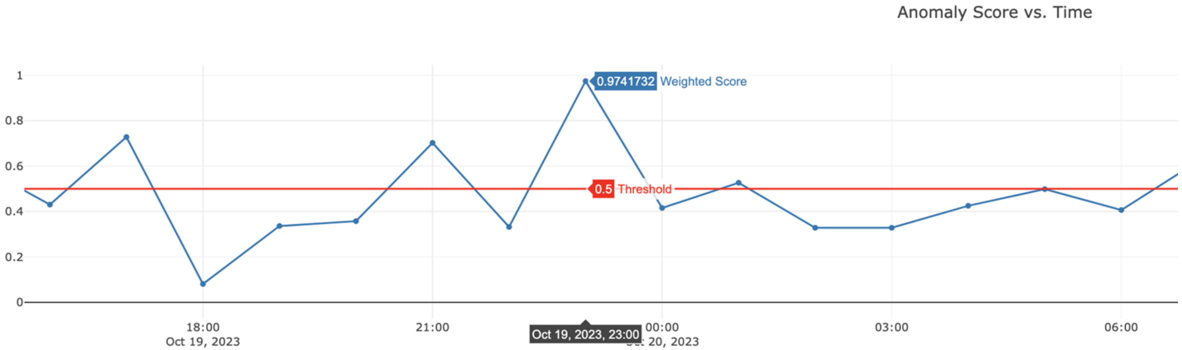
Methodology - Frontended

- Using Flask and Socketio to emit the data once received
- Using Javascript and Ploty to construct a simple dashboard on web in a local port
  - Showing location details and providing interacting tools to set time range and anomaly threshold based on user's preference
  - Two graphs in one window
  - Auto updating and overriding plots if a new weather detail or a weighted score comes

More interaction tools provided by Ploty

Modeling Result - Example A (Hawaii)





Several weather details change at 23:00 and the weighted score is around 0.97.

Conclusions & Recommendations

- Actionable Recommendations
  - Find a way to update the parameters itself
  - We don’t have to find specific parameters for individual location based off its weather pattern
- Outlines Limitations
  - Potential for false positives in anomaly detection for lack of labels
  - Effectiveness is constrained by timeliness of incoming data
- Considerations for Next Steps
  - Refine the anomaly detection algorithms
  - Add searchable locations on the webpage
- Proposed Future Extensions
  - Expand to global weather data
  - Customize alerts for different sectors
  - Integrate with IoT for real-time data collection and AI

Reference

1. River

   https://riverml.xyz/0.19.0/

2. Pyensing

   https://github.com/rotationalio/pyensign

3. Weather Data Playground

   https://github.com/rotationalio/data-playground/tree/main/weather

4. Weather API

   https://www.weather.gov/documentation/services-web-api

Thank you