UHICast

Predict. Prepare. Protect.

Our Team

Ryan Brown

ML Engineer &
Model Evaluation

Betty Zhu

Product Manager
& ML Engineer

Javier Rodriguez

EDA & Application
Developer

Andrea Domiter

Infrastructure &
Data Engineer

Predict.
Prepare.
Protect.

Motivation

April 2024 - Cebu City, Philippines

Washington Post

Southeast Asia is enduring a brutal, record-setting heat wave

The New York Times

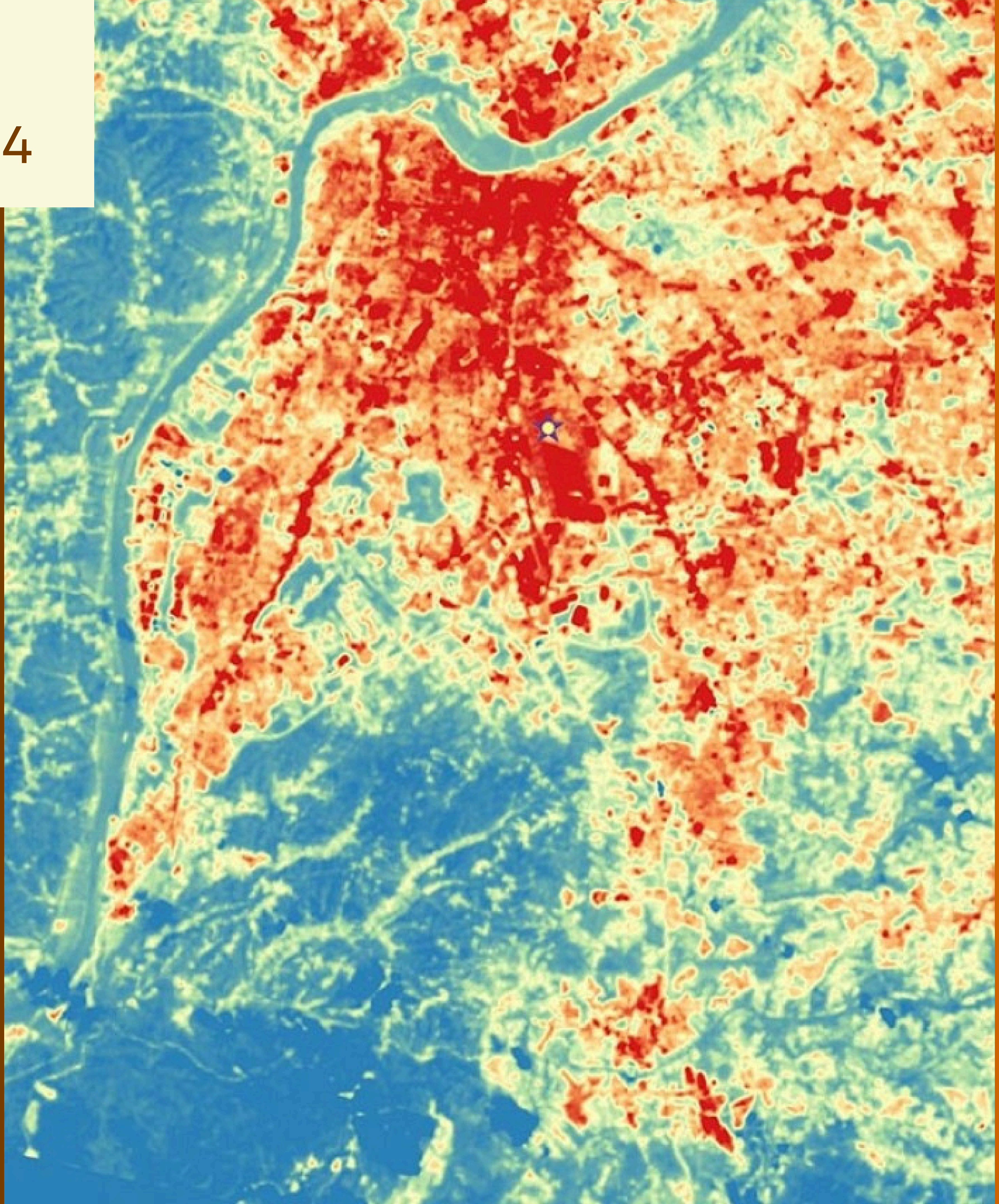
Philippines Closes Schools as Heat Soars to ‘Danger’ Level

Bloomberg

Intense Heat Risks Power Supplies in Philippines, Marcos Says

Andrea’s grandma cooling herself in front of a fan in Cebu, Philippines.

A student doing school work at home in front of a fan in Manila, Philippines, April 26, 2024. Lisa Marie David/Reuters



Predict.
Prepare.
Protect.

Urban Heat Island (UHI)

The UHI effect refers to the phenomenon where urban areas experience higher temperatures than their rural surroundings due to human activities and alterations to the natural environment.

Urban heat in Louisville, Ky. measured by satellite. Source: Climate Central

Axios

"Heat islands" worsening extreme temperatures across the U.S.

Predict.
Prepare.
Protect.

A Global Issue

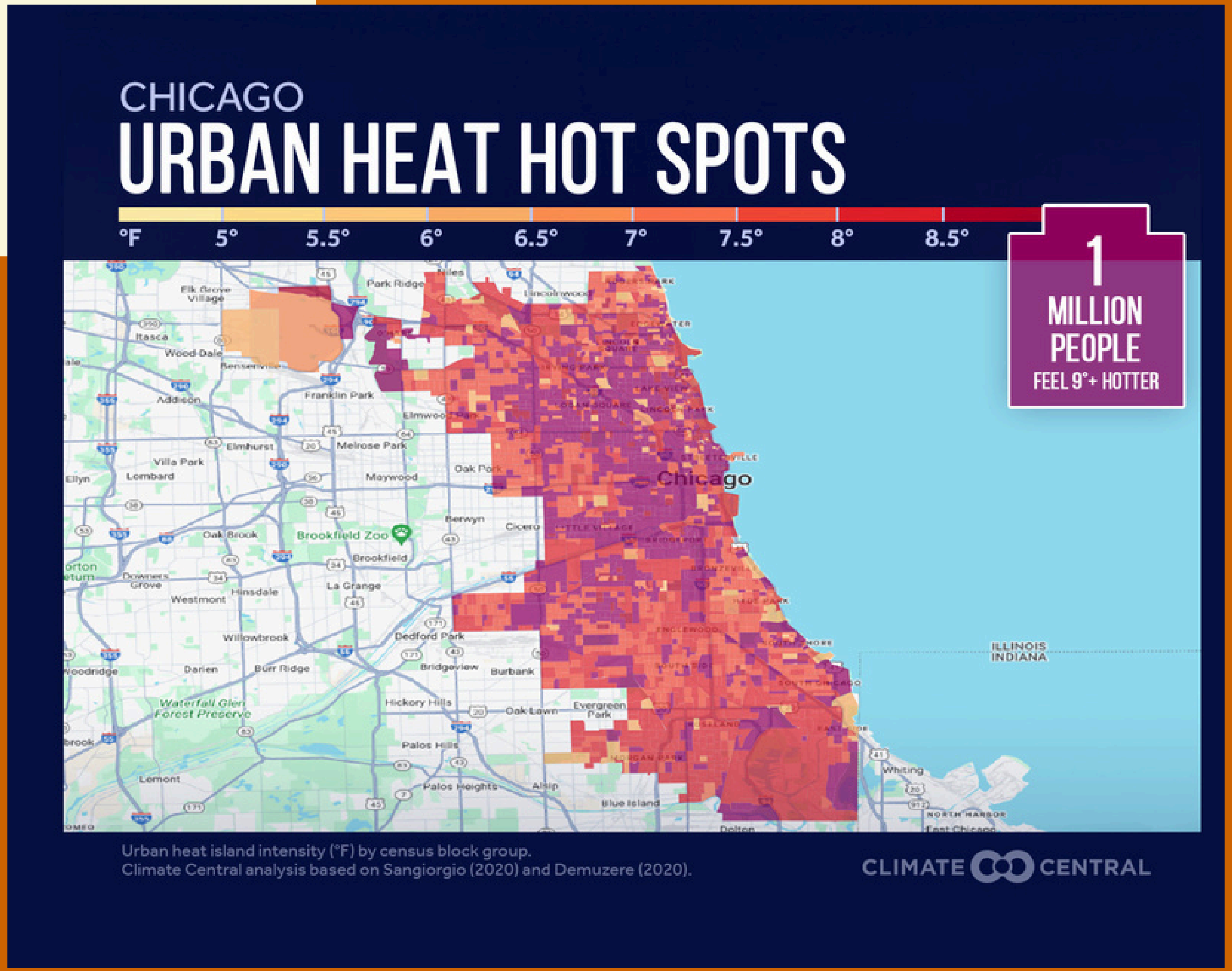
A boy pours water on himself to cool down at a public well in a densely populated area in Jakarta, Indonesia, May 16, 2024. Willy Kurniawan/Reuters

Student Lim Sokha, 15, uses a fan to cool down during her class in Phnom Penh, Cambodia, on May 2, 2024. Chan Tha Lach/Reuters

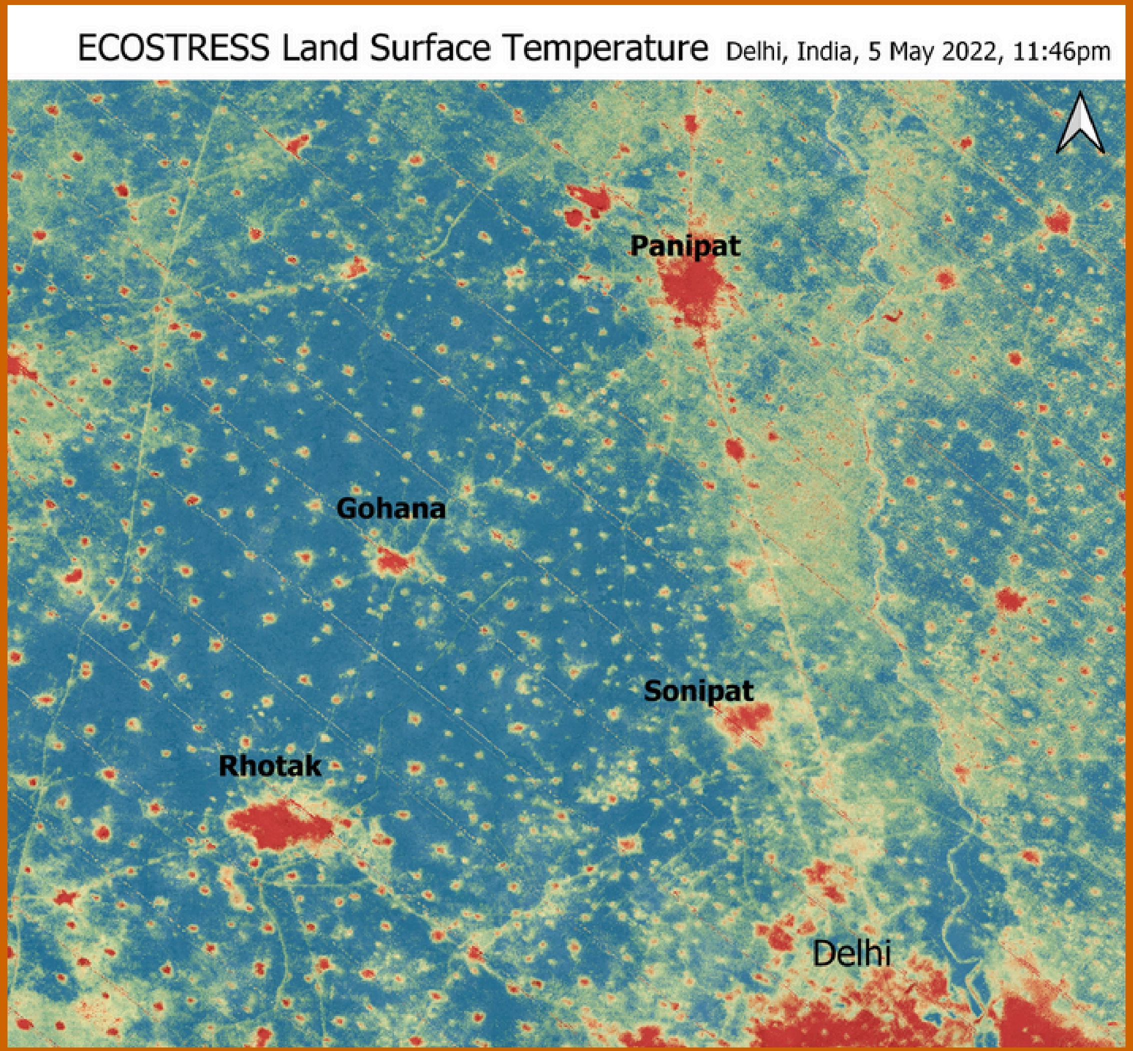
06

Current Initiatives

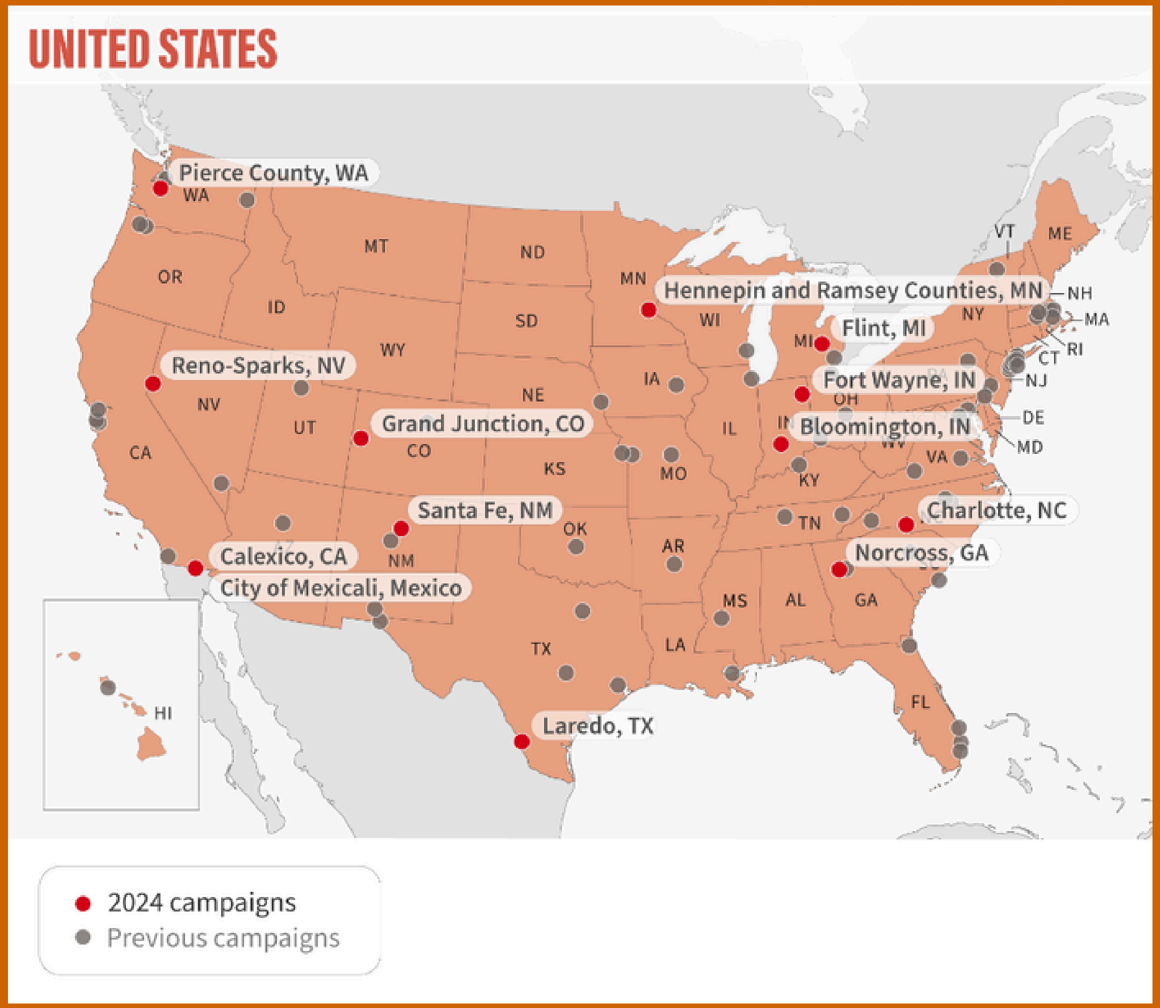
Predict.
Prepare.
Protect.



Climate Central



NASA's ECOSTRESS



NOAA + CAPA

07

Solution: UHICast

Predict.
Prepare.
Protect.

UHICast accurately forecasts land surface temperatures (LST), helping urban planners, policymakers, and energy companies identify high-risk heat areas and allocate resources effectively.

Dynamic Forecasts

Real-time & Cost-effective

Global Coverage

UHICast

Predict.
Prepare.
Protect.

UHICast

Home Features Prediction Benchmark Research Team

High Precision

Our AI models provide granular predictions at the 30 meter level.

10-Day Forecast

Plan ahead with our accurate 10-day UHI temperature predictions.

Interactive Maps

Visualize heat distribution across cities with our interactive maps.

UHI Prediction

City

Select a city

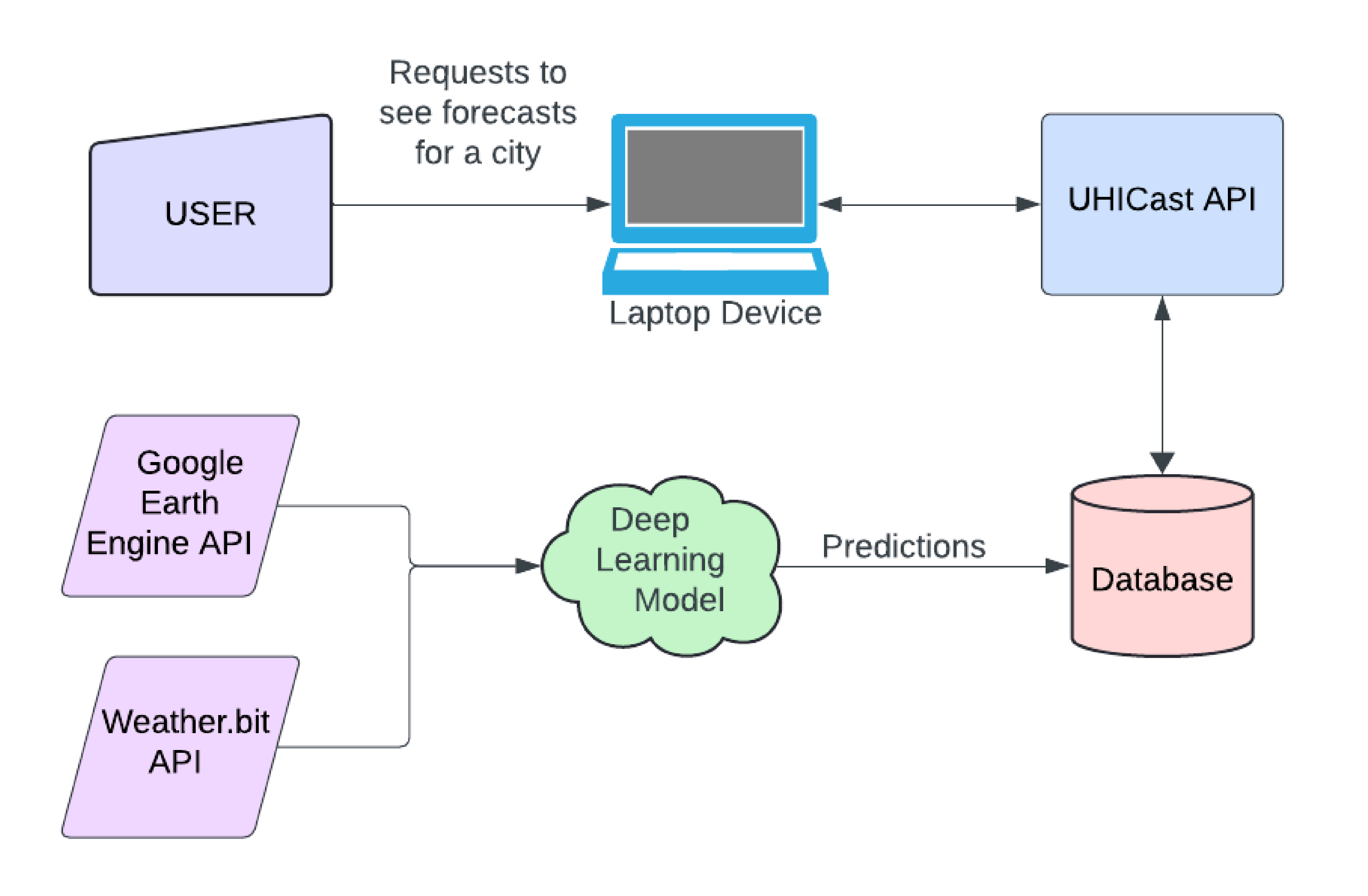
Date

Predict

https://uhicast.com/#prediction

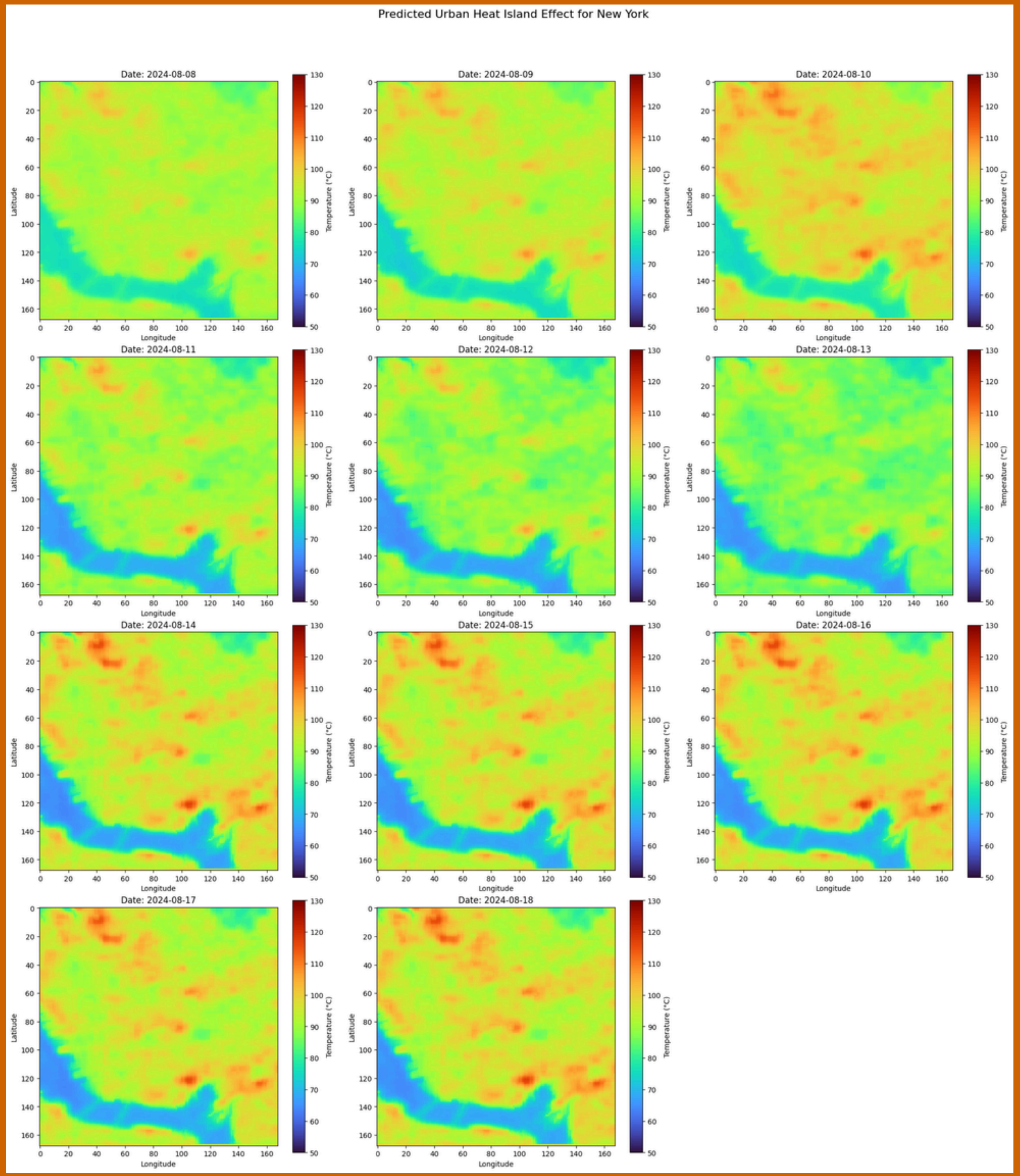
Predict.
Prepare.
Protect.

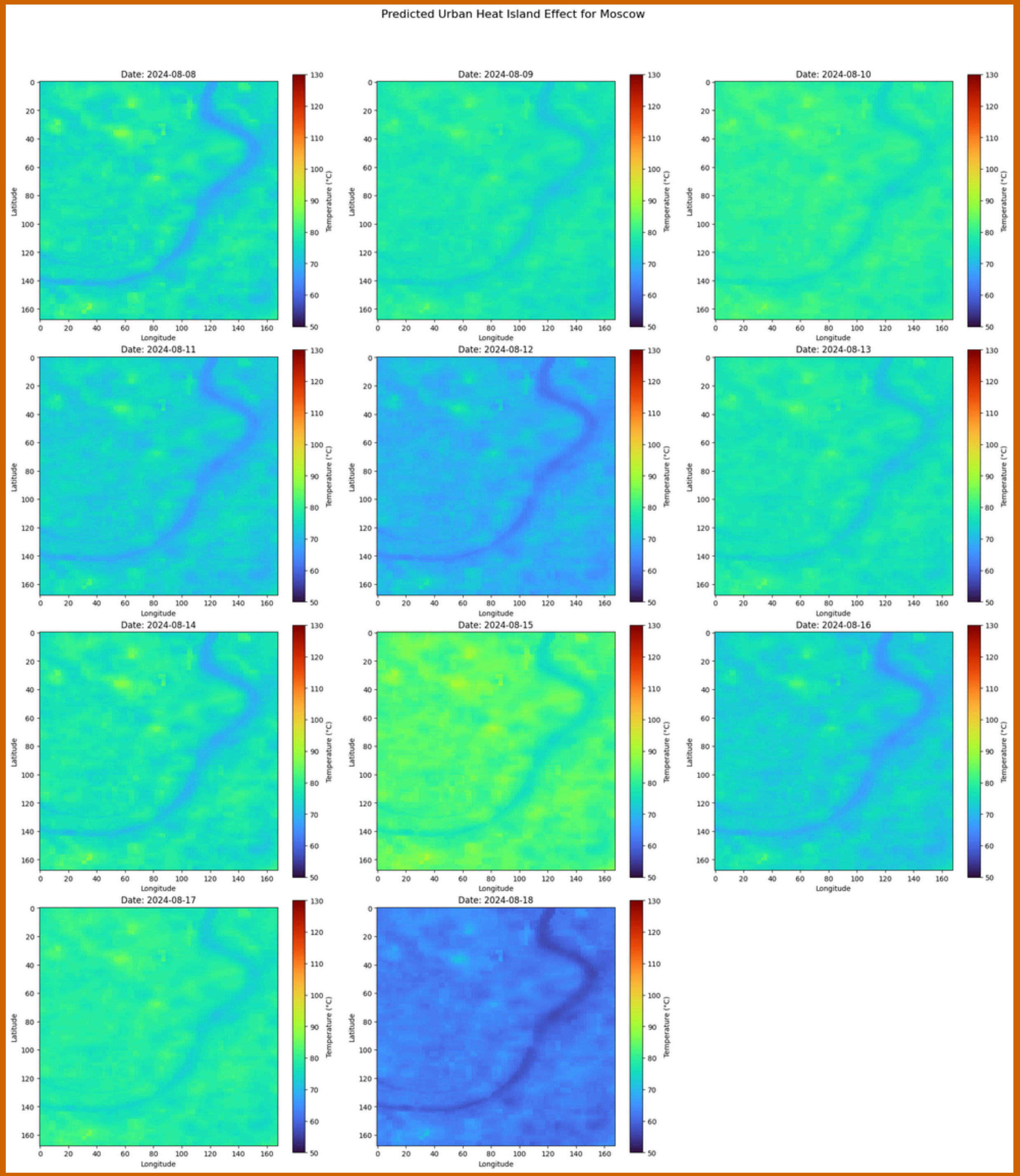
Front End Work Flow

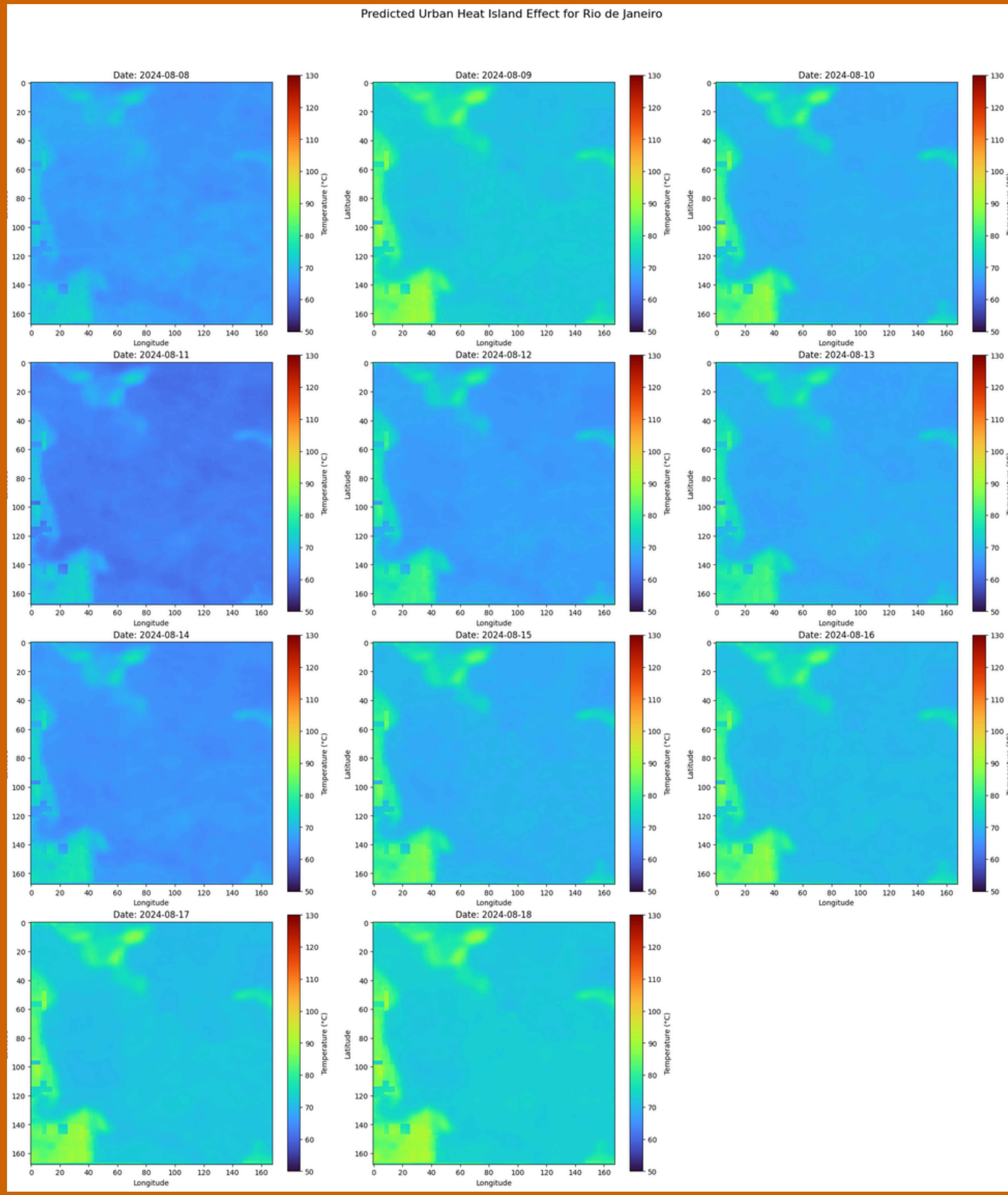


Predict.
Prepare.
Protect.

Behind the Scenes







Predictions For The Next Ten Days update Daily

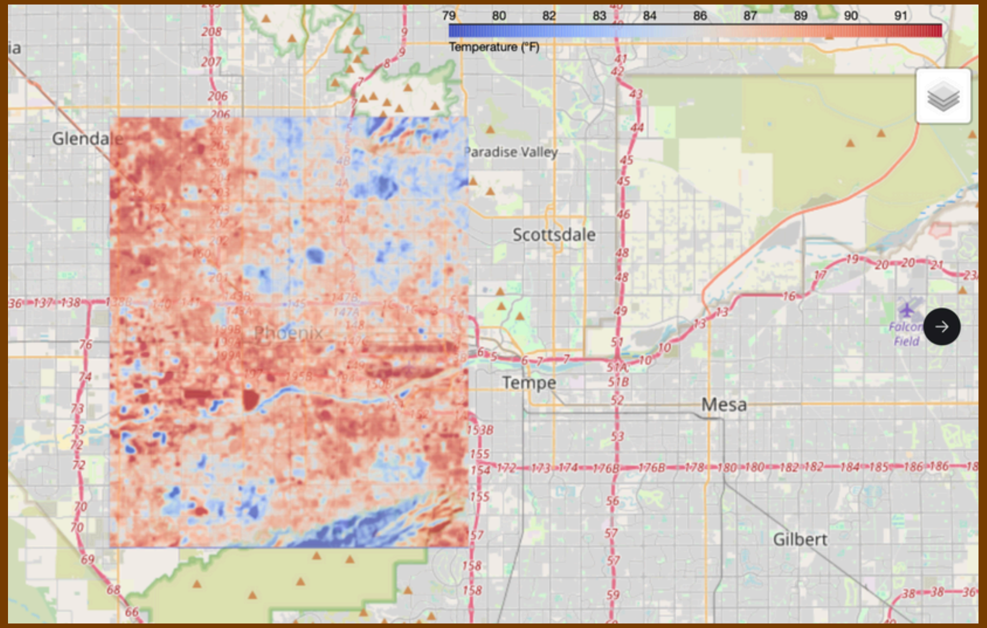
Predict.
Prepare.
Protect.

Data Pipeline

Source: Google Earth Engine, Landsat 8 2013-2024

```python
def process_landsat_data(credentials_path, service_account, city, coords, start_date, end_date):
    initialize_ee(credentials_path, service_account)

    region = ee.Geometry.Polygon(coords)

    landsat8 = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2').filterDate(start_date, end_date).filterBounds(region)
    landsat_masked = landsat8.map(maskL8sr).map(apply_scale_factors)

    download_folder = f'{city}_images_all'
    if not os.path.exists(download_folder):
        os.makedirs(download_folder)

    for file in os.listdir(download_folder):
        os.remove(os.path.join(download_folder, file))

    if landsat_masked.size().getInfo() > 0:
        for i in range(landsat_masked.size().getInfo()):
            ee_image = ee.Image(landsat_masked.toList(landsat_masked.size()).get(i))
            if has_non_zero_values(ee_image, region):
                download_image(ee_image, region, download_folder, city)
    else:
        print("No images found for the specified criteria.")

    output_directory = f'{city}_images_full_only'
    filter_geotiffs(download_folder, output_directory)
```

Process Landsat Data using Python



Example Satellite Image

Source: Station observations

```python
RATE_LIMIT_CALLS = 60  # Number of allowed API calls
RATE_LIMIT_PERIOD = 60  # Time period in seconds
CLUSTER_DISTANCE = 0.5  # Distance in kilometers to form clusters

def fetch_weather_data(latitude, longitude, date_time):
    base_url = "https://api.weatherbit.io/v2.0/history/subhourly"
    date_str = date_time.strftime('%Y-%m-%d')
    params = {
        "lat": latitude,
        "lon": longitude,
        "start_date": date_str,
        "end_date": (date_time + timedelta(days=1)).strftime('%Y-%m-%d'),
        "key": API_KEY
    }
    response = requests.get(base_url, params=params)
    data = response.json()
    closest_time = None
    min_time_diff = timedelta.max
    closest_entry = None

    for entry in data['data']:
        entry_time = datetime.fromisoformat(entry['timestamp_local'])
        entry_time = date_time.tzinfo.localize(entry_time)
        time_diff = abs(entry_time - date_time)
        if time_diff < min_time_diff:
            min_time_diff = time_diff
            closest_time = entry_time
            closest_entry = entry
```

Use Weather.bit API to get data

Example UHI and station variables heatmap

Data

Predict.
Prepare.
Protect.

Satellite

The Model Looks at the Current Month's Average and the Past Month's

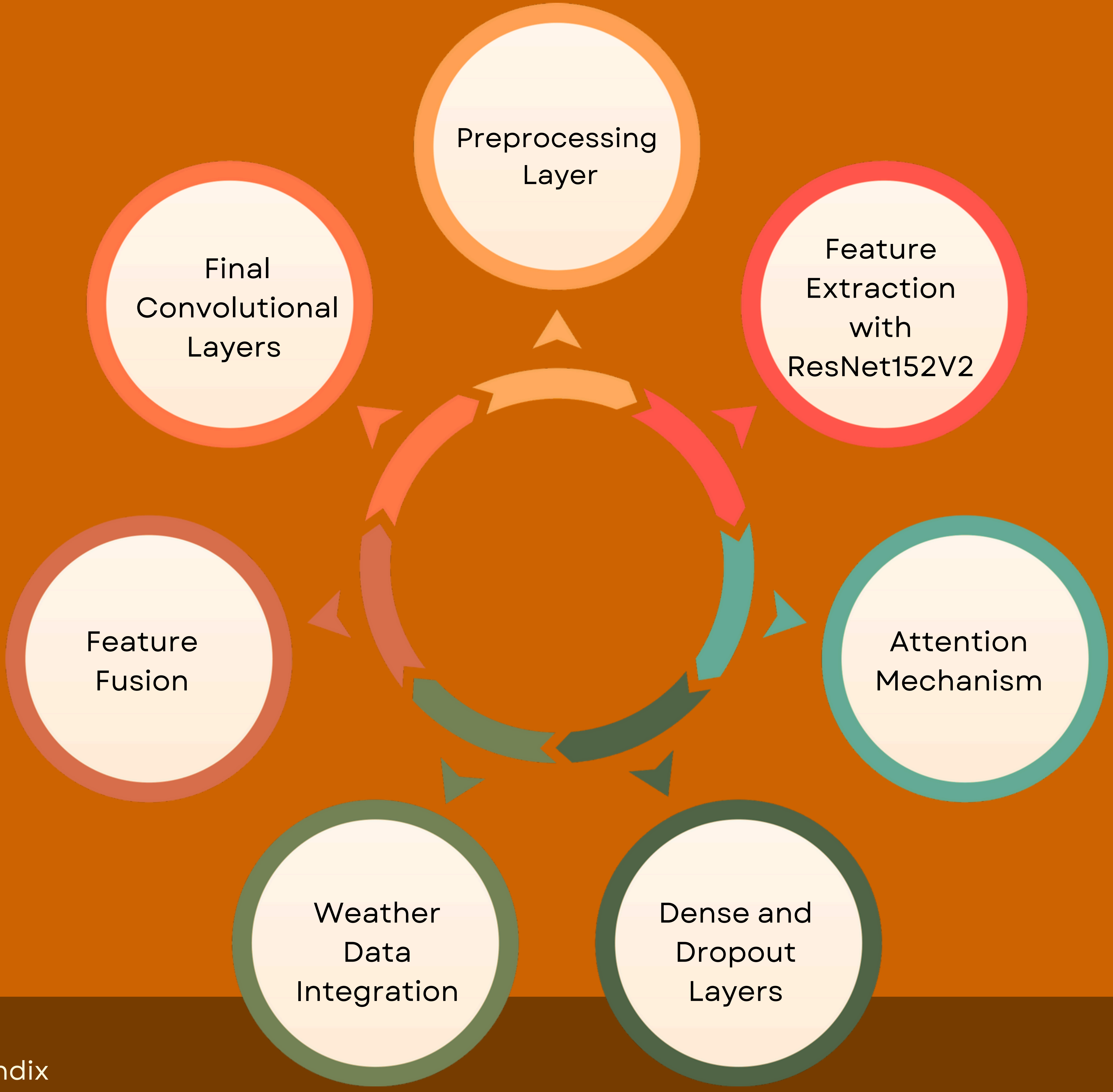
1. Latitude
2. Longitude
3. Vegetation Indices: NDVI, NDBI, EVI, SAVI
4. Albedo: Measure of surface reflectivity
5. NDWI: Detects water bodies and moisture content
6. LST: Thermal measurement of ground temperature
7. Impervious Surface: Higher for areas that don't absorb water (e.g., concrete)

Sensor

Leverages Sensor Data From When the Satellite Image Was Taken

1. Temperature (°F)
2. Apparent Temperature: "Feels like" temperature
3. Cloud Cover
4. Dew Point
5. Atmospheric pressure
6. Relative Humidity
7. Sea Level Pressure
8. Solar Radiation
9. UV Index
10. Visibility: Distance at which objects can be seen
11. Wind Direction (degrees)
12. Wind Gust Speed
13. Wind Speed
14. Month: Numerical representation of the month (1-12)

DeepSatNet: Advanced Spatio-Temporal Prediction Model*

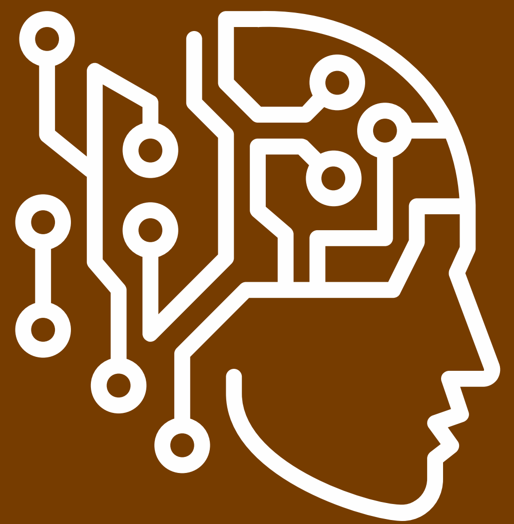


Predict. Prepare. Protect.

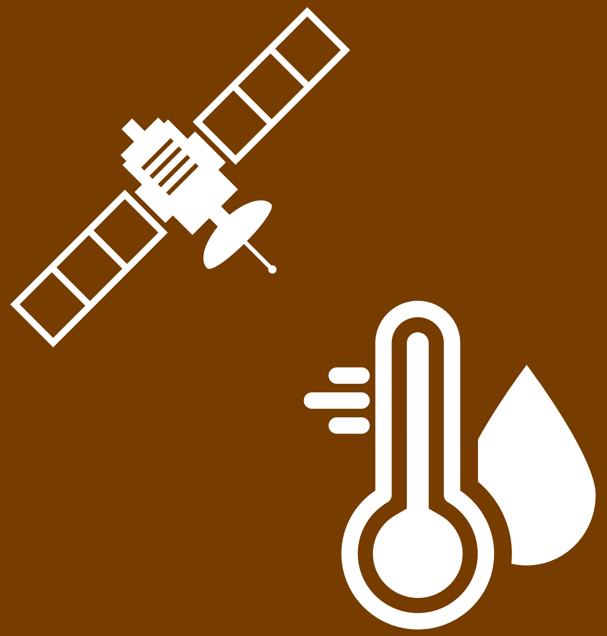
*More information can be found in the appendix

Predict.
Prepare.
Protect.

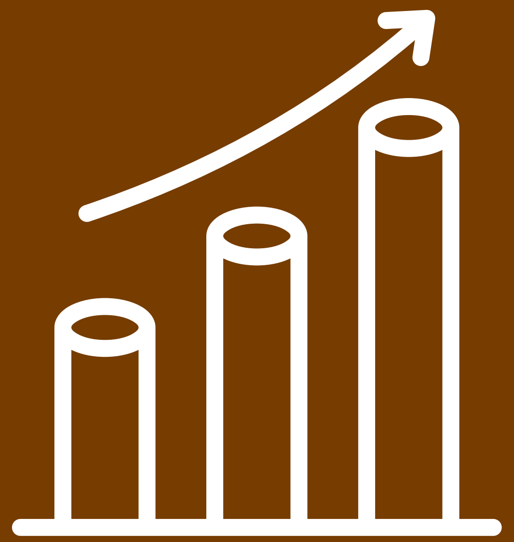
Key Features



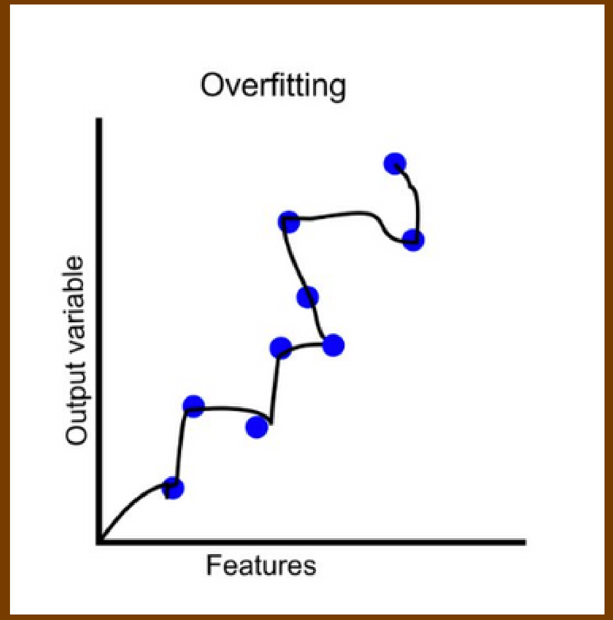
State-of-the-Art Architecture: Combines the power of ResNet152V2 and attention mechanisms



Multi-Modal: Integrates satellite imagery and weather data for comprehensive analysis.



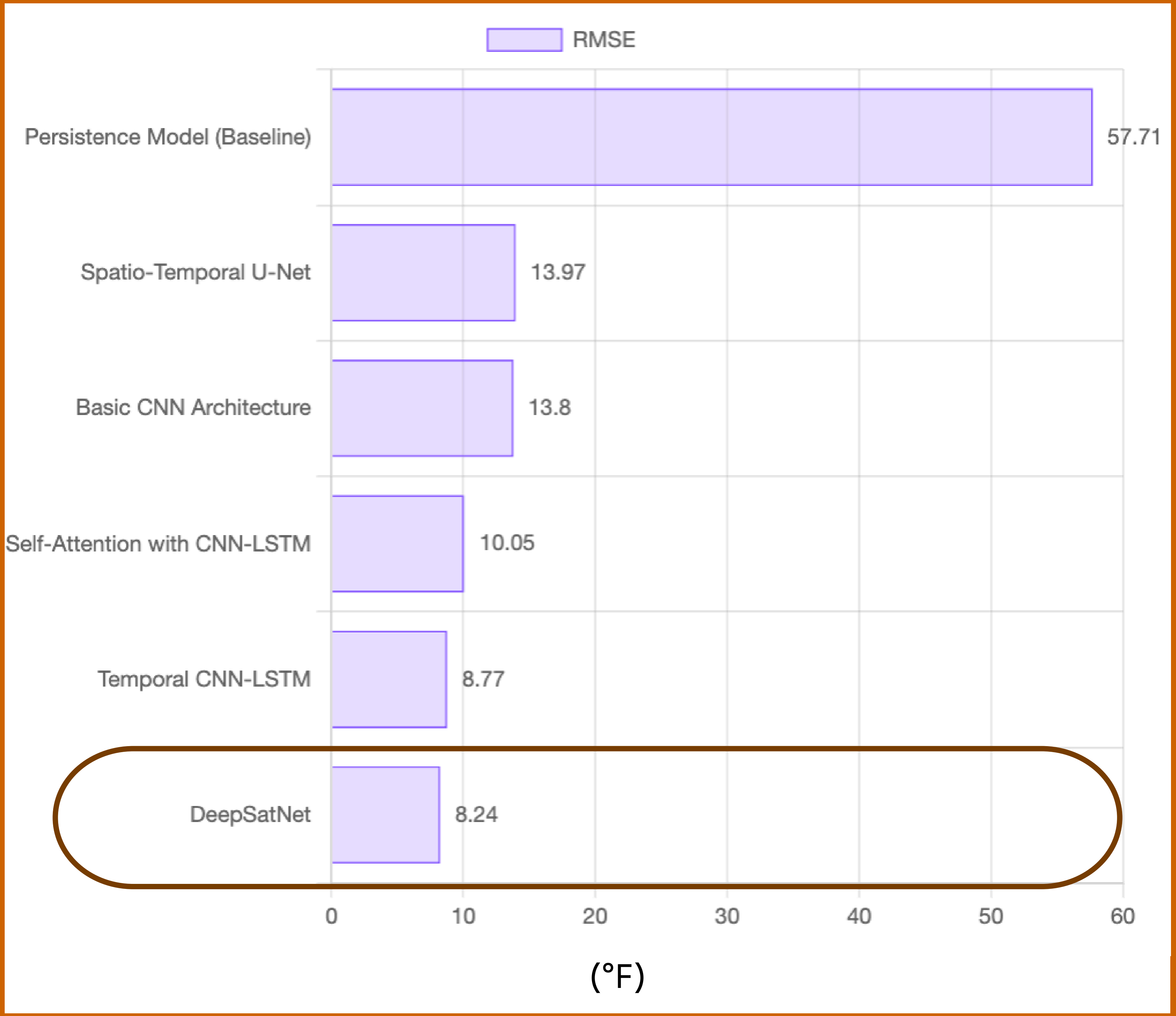
Scalable and Flexible: Designed to handle large-scale spatio-temporal data, making it suitable for various prediction tasks.



Prevent Overfitting: Incorporates dropout layers and regularization techniques

Predict.
Prepare.
Protect.

Model Evaluation

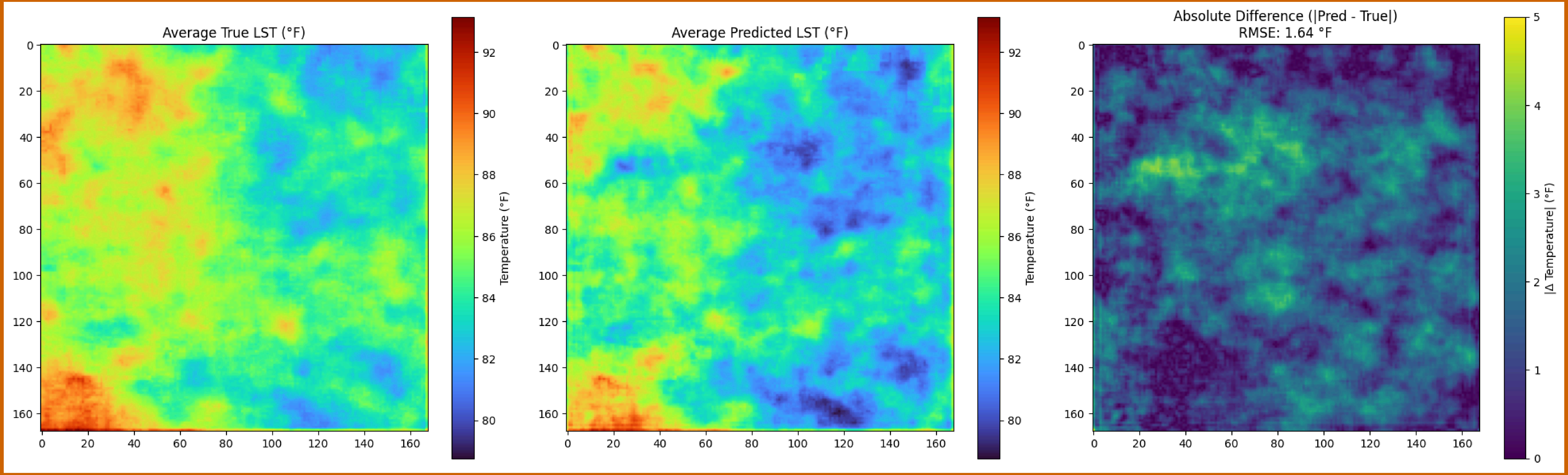


Test set is the past year for all 14 cities within our current network

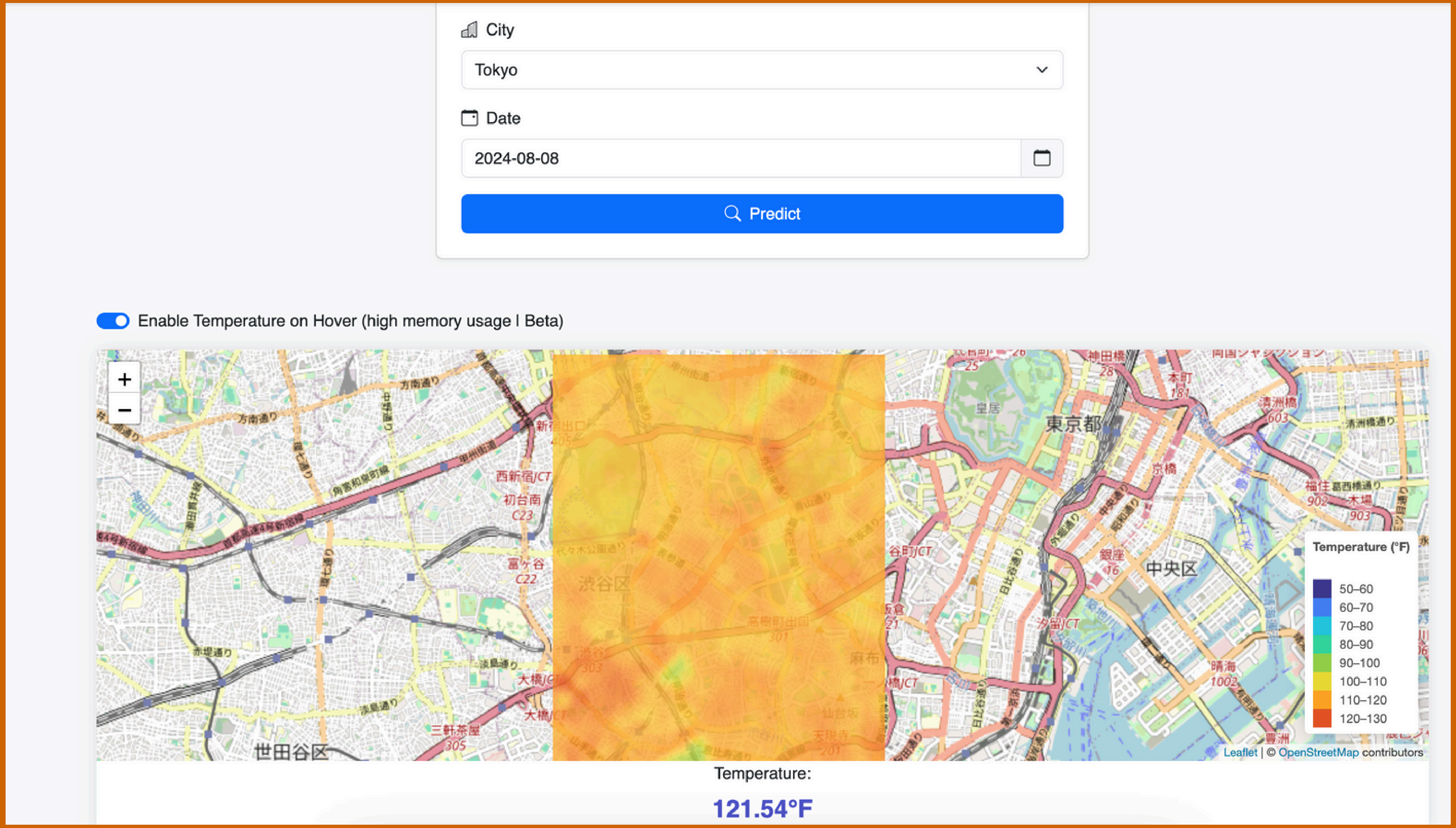
Limitations of the Persistence Model: The model's effectiveness diminishes in cloudy areas due to the necessity of forward-filling data, which can lead to significant errors.

Predict.
Prepare.
Protect.

Model Evaluation



Model Evaluation




More than 120 people died in Tokyo from heatstroke in July as average temperatures hit record highs

by Mari Yamaguchi

Pedestrian holding parasols stand under an intense sun at Ginza shopping street in Tokyo, on July 8, 2…

Predict.
Prepare.
Protect.

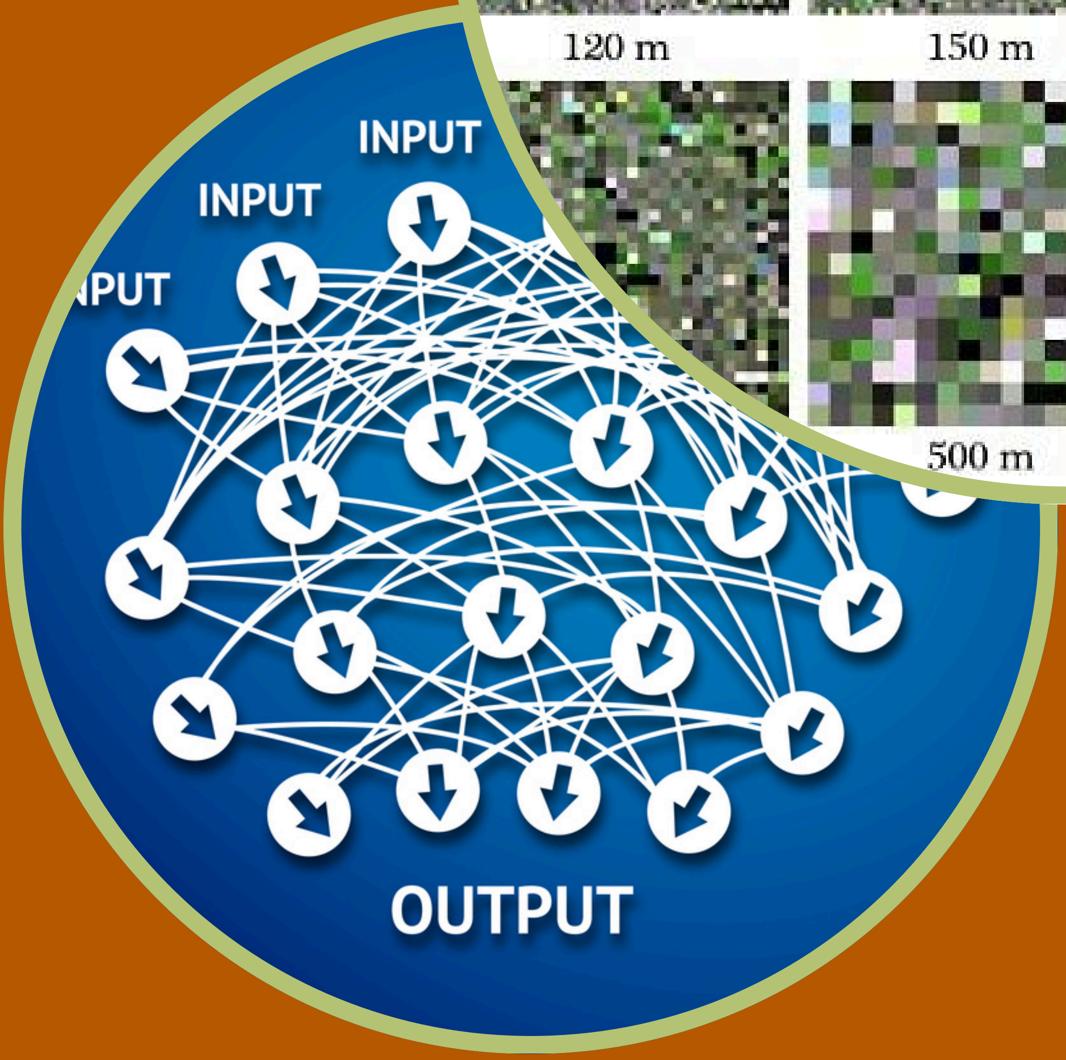
Key Learnings

It is possible to forecast Land Surface Temperature

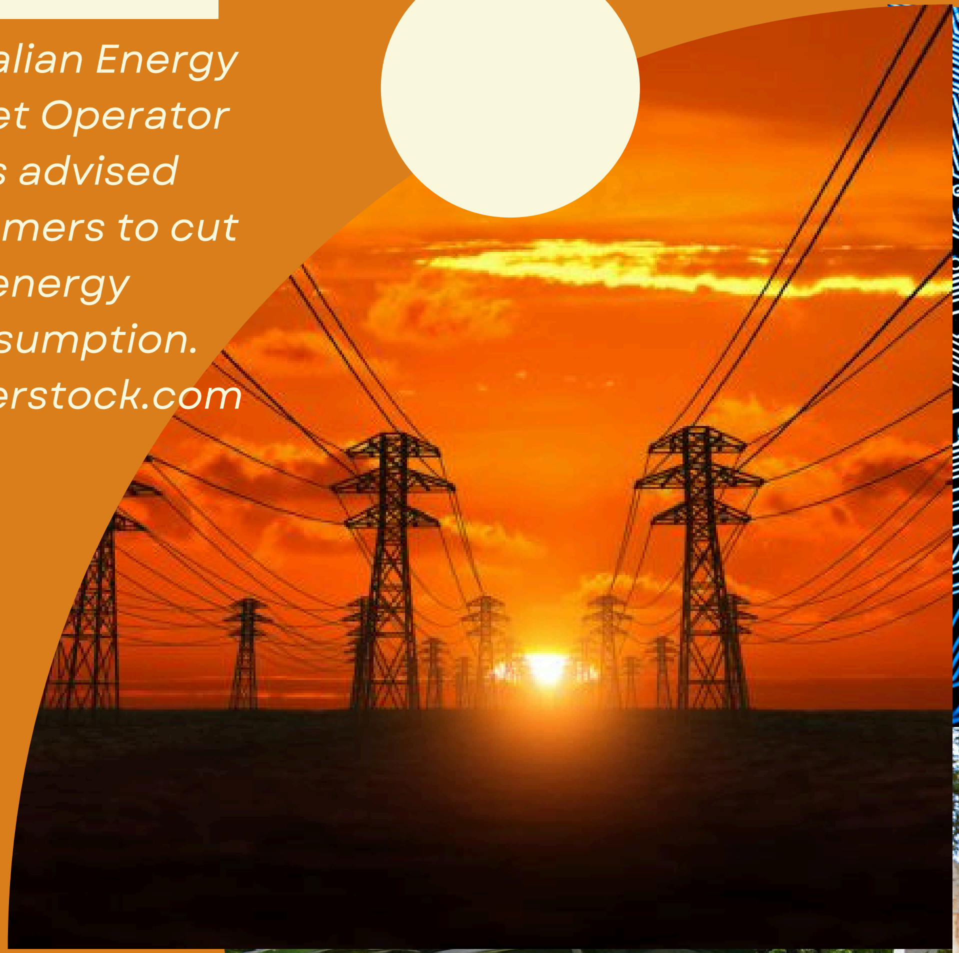
A general model that works for many cities is desirable

Deep Learning is likely the path to solving this problem

At a 30 meter resolution the size of the data grows quickly - investment is needed in this space



Predict.
Prepare.
Protect.



Australian Energy Market Operator has advised consumers to cut energy consumption. Shutterstock.com

Photo Credit: Alta.com

The cooling center at the Mid Valley Senior Center in Panorama City (Ashley Balderrama/ LAist)

Roadmap

Scale!

We are working on publishing our modeling approach and are seaking funding to further this research.

In the meantime we are also exploring pixel-wise conformal prediction.

Our goal is to have a performant model that can forecast land surface temperature at low temproal and spatial resoultions for every city on earth.

The Galeana family plays in the Belle Haven Pool to cool down on July 2, 2024.
Photo by Anna Hoch-Kenney.

People seek relief from a dangerous heat wave at the Crown Fountain and wading pool in Chicago, June 15, 2022.
Tannen Maury/EPA via Shutterstock

This photo taken on May 22, 2023, shows a man transporting containers of water on his motorbike in Hanoi, Vietnam.
Nhac Nguyen/AFP/Getty Images

Workers move blocks of ice into a storage unit at a fresh market during heat wave conditions in Bangkok on April 25.
Lillian Suwanrumpha/AFP/Getty Images

Our Mission

Predict. Prepare. Protect.

Thank you!

Do you have any questions?

Ryan Brown - rbrown55@berkeley.edu
Betty Zhu - bettyzhu912@berkeley.edu
Javier Rodriguez - jrodriguezjr@berkeley.edu
Andrea Domiter - adomiter@berkeley.edu

Appendix

Data

Predict.
Prepare.
Protect.

Satellite

The Model Looks at the Current Month's Average and the Past Month's

1. Latitude
2. Longitude
3. NDVI: Indicates Vegetation
4. NDBI: Indicates urban/built-up areas
5. EVI: Another vegetation index, less sensitive to atmospheric conditions
6. SAVI: Accounts for soil influence in vegetation monitoring
7. Albedo: Measure of surface reflectivity
8. NDWI: Detects water bodies and moisture content
9. LST: Thermal measurement of ground temperature
10. Impervious Surface: Higher for areas that don't absorb water (e.g., concrete)

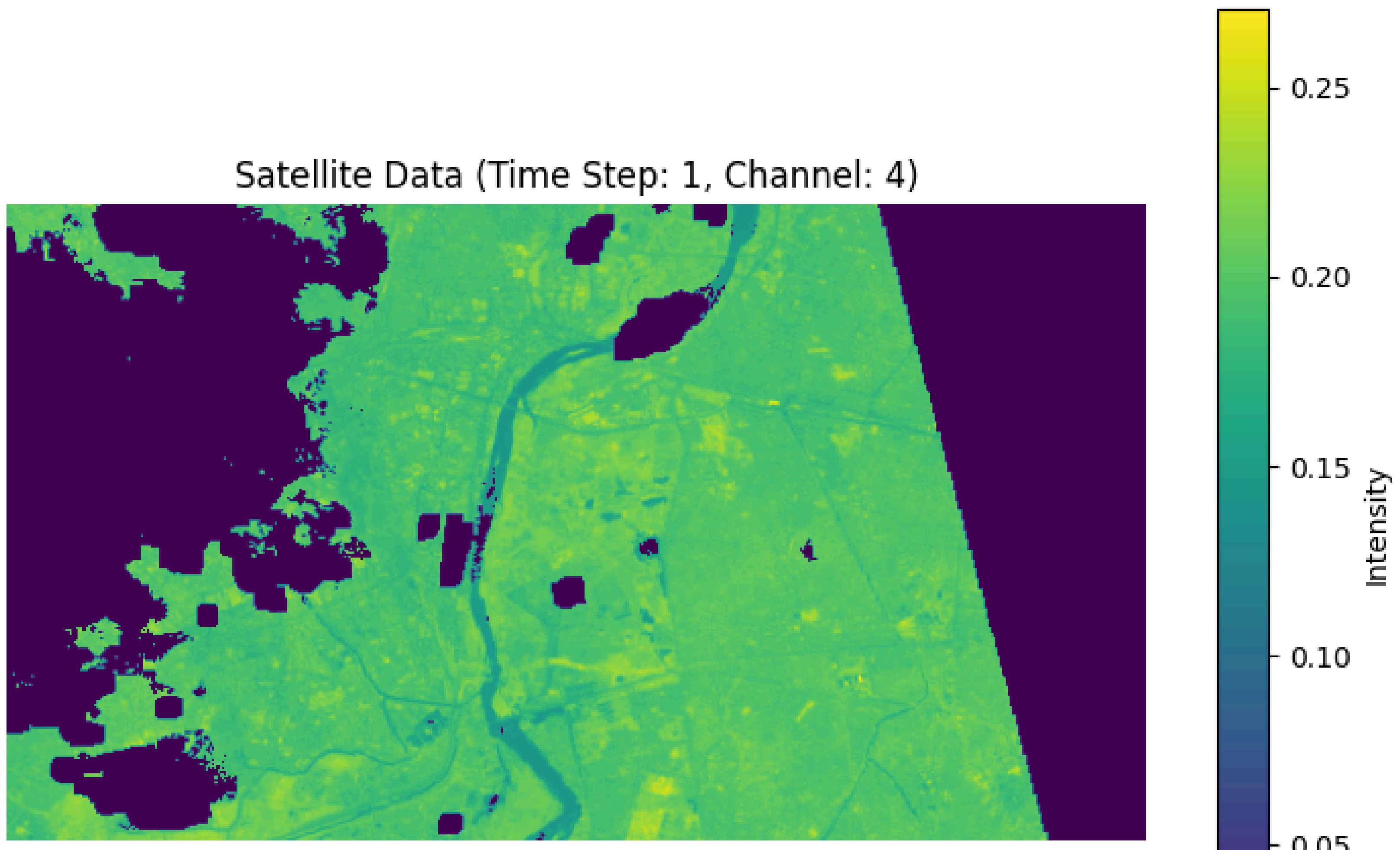
Sensor

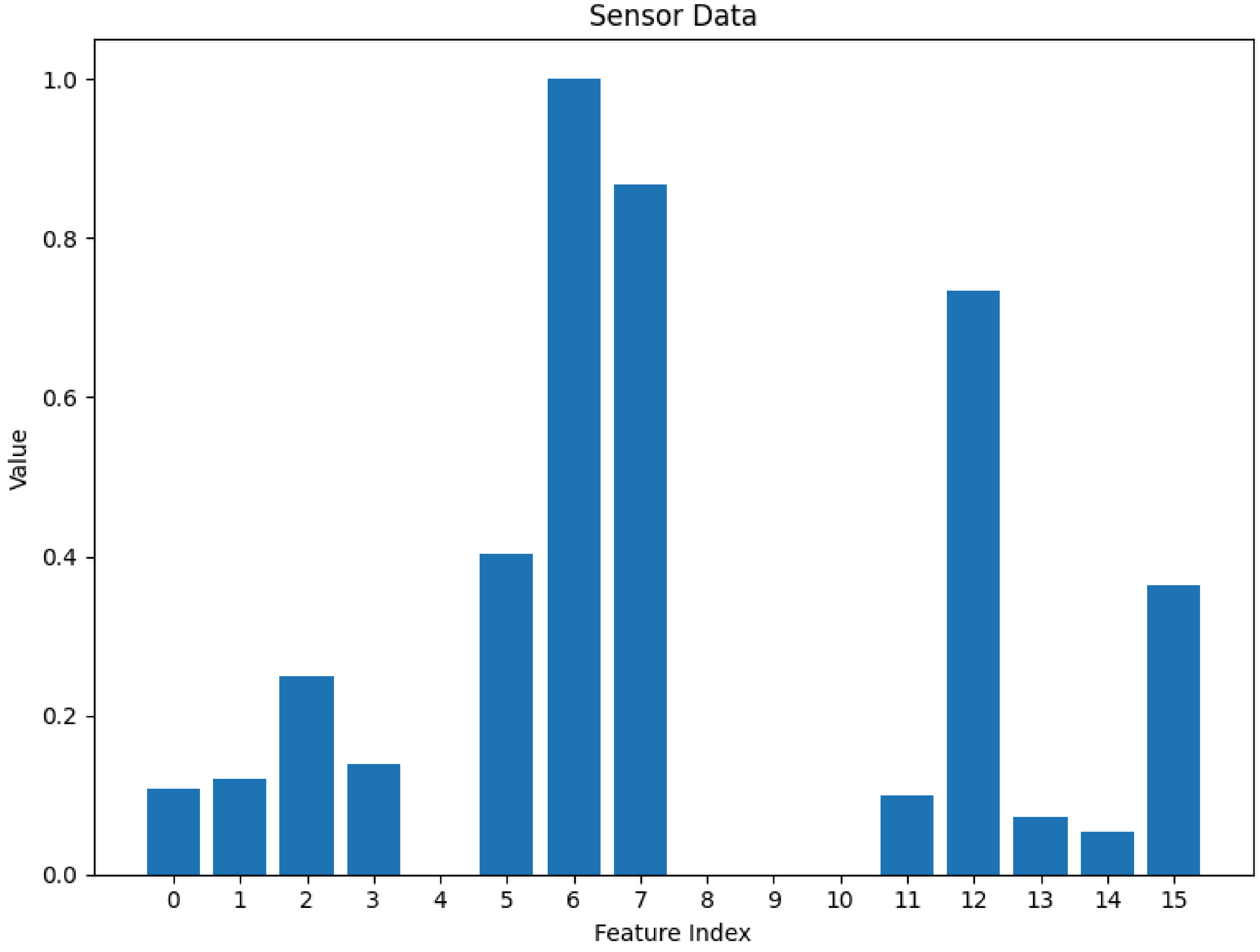
Leverages Sensor Data From When the Satellite Image Was Taken

1. Temperature: Air temperature (°F)
2. Apparent Temperature: "Feels like" temperature
3. Cloud Cover: Percentage of sky covered by clouds
4. Dew Point: Where air becomes saturated with water vapor
5. Pressure: Atmospheric pressure
6. Relative Humidity: Amount of water vapor in air
7. Sea Level Pressure: Pressure adjusted to sea level
8. Solar Radiation: Solar energy reaching the surface
9. UV Index: Measure of ultraviolet radiation intensity
10. Visibility: Distance at which objects can be seen
11. Wind Direction: Direction wind is blowing (degrees)
12. Wind Gust Speed: Peak wind speed in short bursts
13. Wind Speed: Sustained wind speed
14. Month: Numerical representation of the month (1-12)

Data

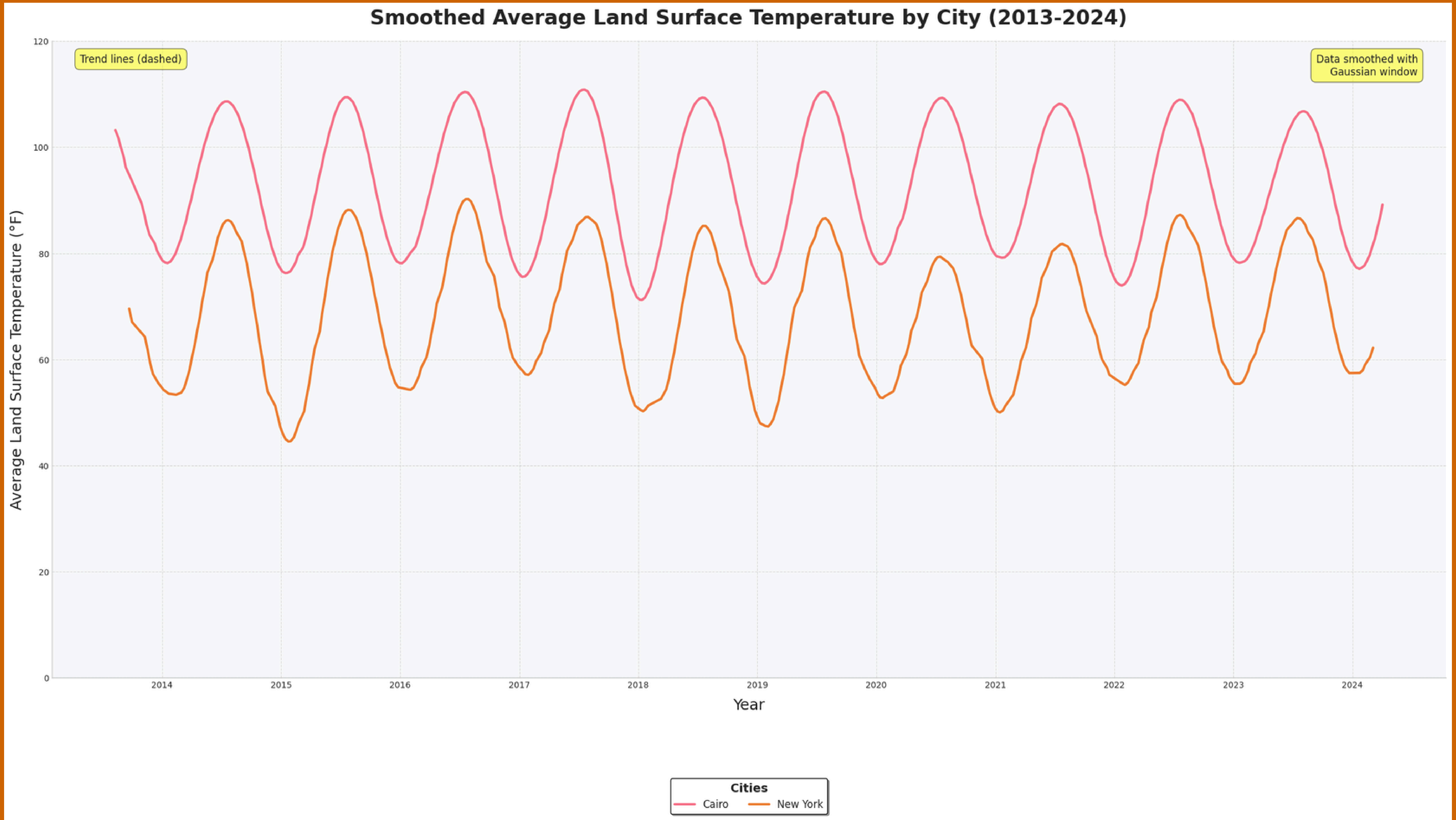
Predict.
Prepare.
Protect.





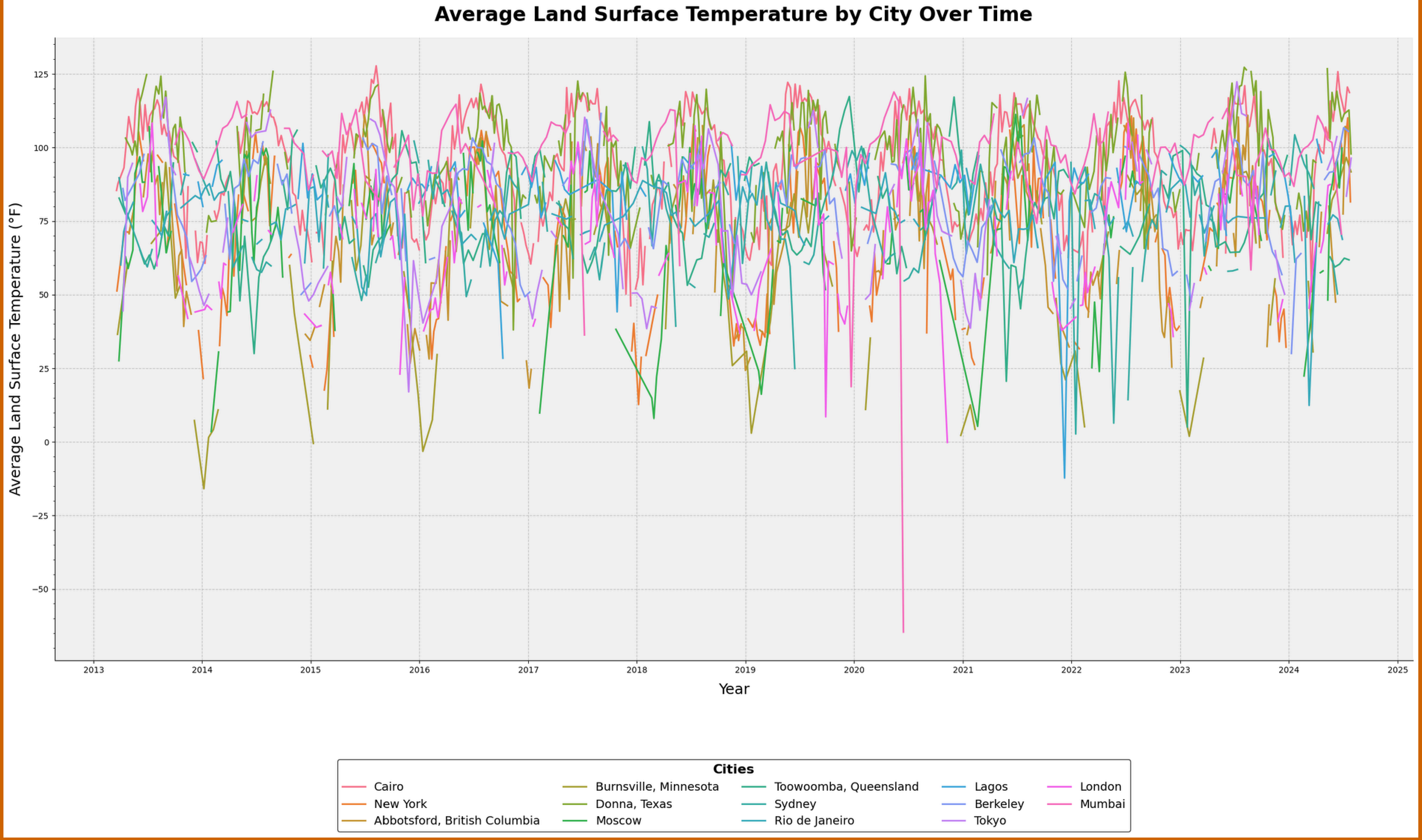
EDA

Predict.
Prepare.
Protect.

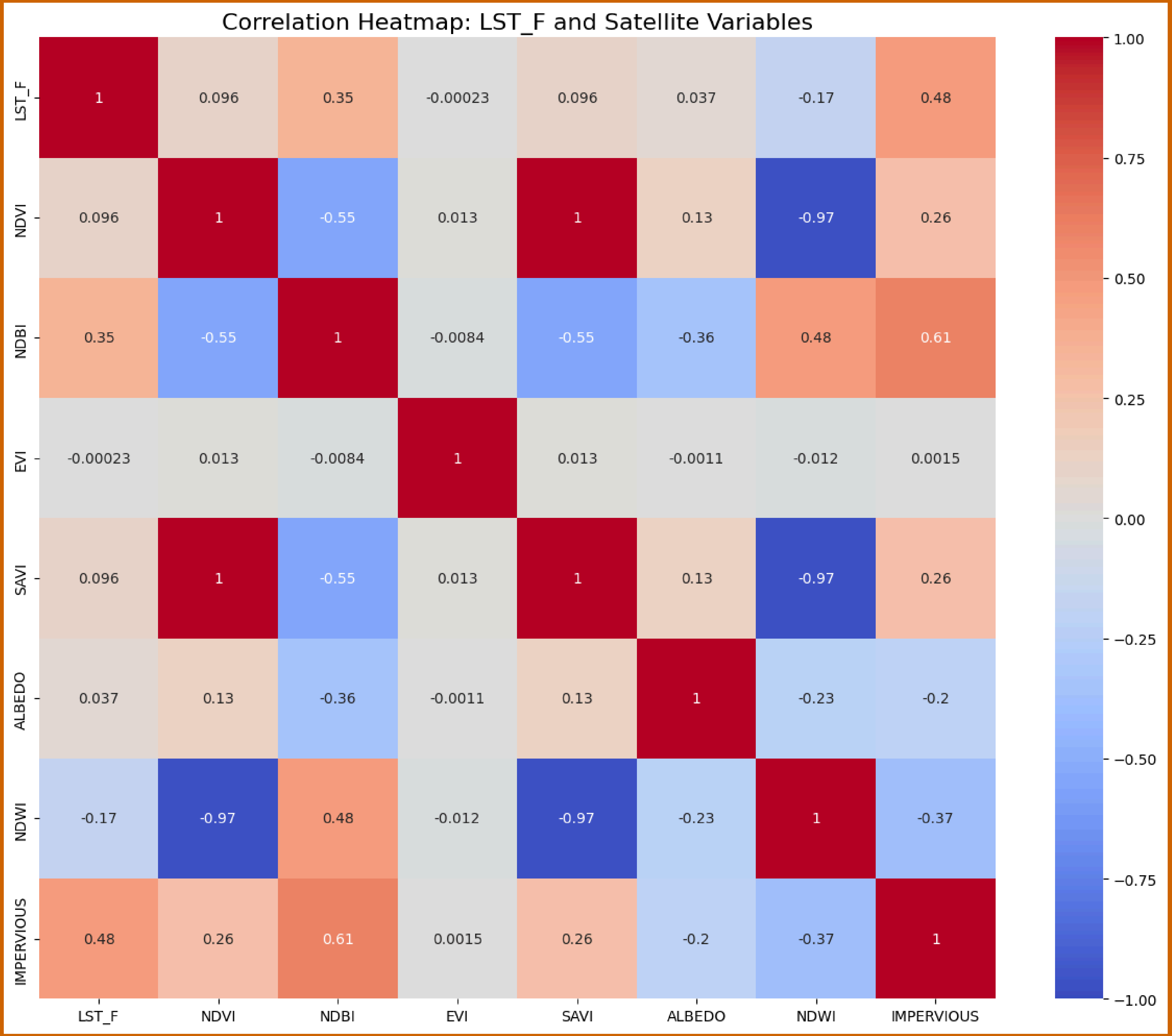


EDA

Predict.
Prepare.
Protect.

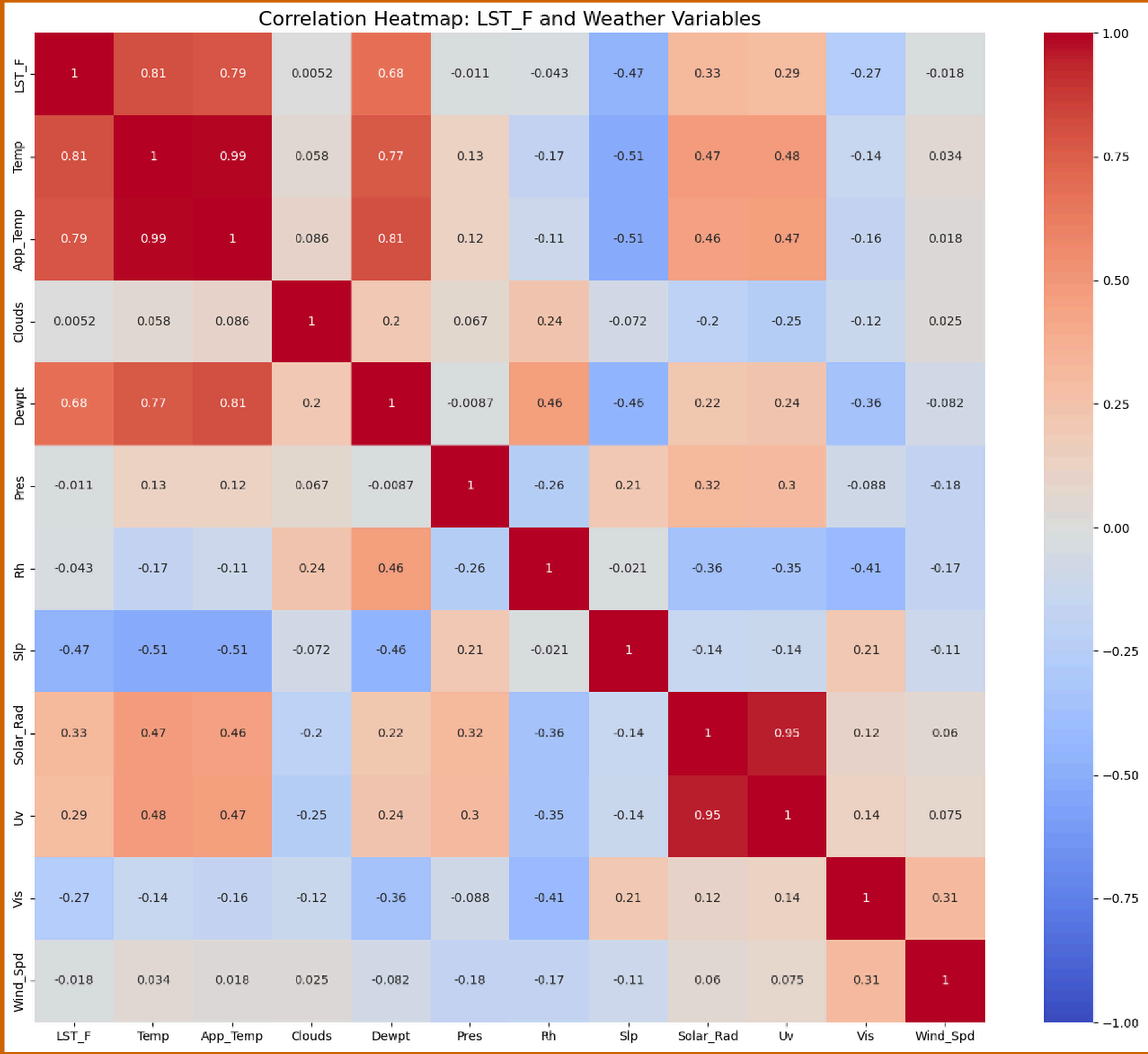


EDA



Predict.
Prepare.
Protect.

EDA



Predict.
Prepare.
Protect.

Model

```python
class PreprocessLayer(layers.Layer):
    def __init__(self, **kwargs):
        super(PreprocessLayer, self).__init__(**kwargs)
        self.conv = layers.Conv2D(3, (1, 1), padding='same', activation='relu')
        self.resize = layers.Resizing(224, 224)  # ResNet152V2 expects 224x224 input

    def call(self, inputs):
        x = self.conv(inputs)
        x = self.resize(x)
        return x

    def compute_output_shape(self, input_shape):
        return (input_shape[0], 224, 224, 3)

class AttentionLayer(layers.Layer):
    def __init__(self, **kwargs):
        super(AttentionLayer, self).__init__(**kwargs)

    def build(self, input_shape):
        self.W = self.add_weight(name='attention_weight', shape=(input_shape[-1], 1),
                                 initializer='random_normal', trainable=True,
                                 regularizer=regularizers.l1_l2(l1=1e-6, l2=1e-5))
        self.b = self.add_weight(name='attention_bias', shape=(input_shape[1], 1),
                                 initializer='zeros', trainable=True,
                                 regularizer=regularizers.l1_l2(l1=1e-6, l2=1e-5))
        super(AttentionLayer, self).build(input_shape)

    def call(self, x):
        e = tf.tanh(tf.matmul(x, self.W) + self.b)
        a = tf.nn.softmax(e, axis=1)
        output = x * a
        return tf.reduce_sum(output, axis=1)

    def compute_output_shape(self, input_shape):
        return (input_shape[0], input_shape[-1])
```

1. Preprocessing Layer:
    - Converts satellite images to the required format and size.
    - Ensures compatibility with the ResNet152V2 architecture by resizing images to 224x224 pixels.
2. Feature Extraction with ResNet152V2:
    - Utilizes a pre-trained ResNet152V2 model to extract high-level features from the satellite images.
    - Applies TimeDistributed layers to handle sequences of satellite images.
3. Attention Mechanism:
    - Implements an attention layer to focus on the most relevant features in the data.
    - Enhances the model's ability to learn from important regions in the input sequences.
4. Dense and Dropout Layers:
    - Applies multiple dense layers to transform the extracted features.
    - Uses dropout layers to prevent overfitting and improve generalization.
5. Weather Data Integration:
    - Processes weather data through dense and dropout layers.
    - Reshapes weather features to match the spatial dimensions of the satellite features.
6. Feature Fusion:
    - Combines satellite and weather features using concatenation.
    - Merges different sources of information to improve prediction accuracy.
7. Final Convolutional Layers:
    - Uses several convolutional layers to further process the combined features.

Model

```python
def create_uhi_model(satellite_shape, weather_shape, output_shape):
    # Satellite input
    satellite_input = layers.Input(shape=satellite_shape)

    # Pretrained model for feature extraction (back to ResNet152V2)
    base_model = ResNet152V2(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
    base_model.trainable = False  # Freeze the pretrained model

    # Preprocess and apply base model to each time step
    x = layers.TimeDistributed(PreprocessLayer())(satellite_input)
    x = layers.TimeDistributed(base_model)(x)

    # Flatten the spatial dimensions
    x = layers.TimeDistributed(layers.GlobalAveragePooling2D())(x)

    # Apply attention layer
    x = AttentionLayer()(x)

    # Dense layers to prepare for upsampling (increased complexity)
    x = layers.Dense(512, activation='relu', kernel_regularizer=regularizers.l1_l2(l1=1e-6, l2=1e-5))(x)
    x = layers.Dropout(0.3)(x)  # Added dropout
    x = layers.Dense(256, activation='relu', kernel_regularizer=regularizers.l1_l2(l1=1e-6, l2=1e-5))(x)
    x = layers.Dropout(0.3)(x)  # Added dropout
    x = layers.Dense(output_shape[0] * output_shape[1], activation='relu', kernel_regularizer=regularizers.l1_l2(l1=1e-6, l2=1e-5))(x)
    x = layers.Reshape((output_shape[0], output_shape[1], 1))(x)

    # Weather input processing (increased complexity)
    weather_input = layers.Input(shape=weather_shape)
    y = layers.Dense(128, activation='relu', kernel_regularizer=regularizers.l1_l2(l1=1e-6, l2=1e-5))(weather_input)
    y = layers.Dropout(0.2)(y)  # Added dropout
    y = layers.Dense(64, activation='relu', kernel_regularizer=regularizers.l1_l2(l1=1e-6, l2=1e-5))(y)
    y = layers.Dropout(0.2)(y)  # Added dropout

    # Reshape weather features to match spatial dimensions
    y = layers.Dense(output_shape[0] * output_shape[1], activation='relu', kernel_regularizer=regularizers.l1_l2(l1=1e-6, l2=1e-5))(y)
    y = layers.Reshape((output_shape[0], output_shape[1], 1))(y)

    # Combine satellite features with weather features
    combined = layers.Concatenate()([x, y])

    # Final convolutional layers (increased complexity)
    z = layers.Conv2D(64, (3, 3), padding='same', activation='relu', kernel_regularizer=regularizers.l1_l2(l1=1e-6, l2=1e-5))(combined)
    z = layers.Dropout(0.2)(z)  # Added dropout
    z = layers.Conv2D(32, (3, 3), padding='same', activation='relu', kernel_regularizer=regularizers.l1_l2(l1=1e-6, l2=1e-5))(z)
    z = layers.Dropout(0.2)(z)  # Added dropout
    z = layers.Conv2D(16, (3, 3), padding='same', activation='relu', kernel_regularizer=regularizers.l1_l2(l1=1e-6, l2=1e-5))(z)
    outputs = layers.Conv2D(1, (1, 1), padding='same')(z)

    # Create model
    model = models.Model(inputs=[satellite_input, weather_input], outputs=outputs)
    return model
```

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 2, 168, 168, 10)	0	-
time_distributed (TimeDistributed)	(None, 2, 224, 224, 3)	0	input_layer[0][0]
time_distributed_1 (TimeDistributed)	(None, 2, 7, 7, 2048)	58,331,648	time_distributed[0][0]
time_distributed_2 (TimeDistributed)	(None, 2, 2048)	0	time_distributed_1[0]…
attention_layer (AttentionLayer)	(None, 2048)	2,050	time_distributed_2[0]…
input_layer_2 (InputLayer)	(None, 14)	0	-
dense (Dense)	(None, 512)	1,049,088	attention_layer[0][0]
dense_3 (Dense)	(None, 128)	1,920	input_layer_2[0][0]
dropout (Dropout)	(None, 512)	0	dense[0][0]
dropout_2 (Dropout)	(None, 128)	0	dense_3[0][0]
dense_1 (Dense)	(None, 256)	131,328	dropout[0][0]
dense_4 (Dense)	(None, 64)	8,256	dropout_2[0][0]
dropout_1 (Dropout)	(None, 256)	0	dense_1[0][0]
dropout_3 (Dropout)	(None, 64)	0	dense_4[0][0]
dense_2 (Dense)	(None, 28224)	7,253,568	dropout_1[0][0]
dense_5 (Dense)	(None, 28224)	1,834,560	dropout_3[0][0]
reshape (Reshape)	(None, 168, 168, 1)	0	dense_2[0][0]
reshape_1 (Reshape)	(None, 168, 168, 1)	0	dense_5[0][0]
concatenate (Concatenate)	(None, 168, 168, 2)	0	reshape[0][0], reshape_1[0][0]
conv2d_1 (Conv2D)	(None, 168, 168, 64)	1,216	concatenate[0][0]
dropout_4 (Dropout)	(None, 168, 168, 64)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 168, 168, 32)	18,464	dropout_4[0][0]
dropout_5 (Dropout)	(None, 168, 168, 32)	0	conv2d_2[0][0]
conv2d_3 (Conv2D)	(None, 168, 168, 16)	4,624	dropout_5[0][0]
conv2d_4 (Conv2D)	(None, 168, 168, 1)	17	conv2d_3[0][0]

Total params: 68,636,739 (261.83 MB)
Trainable params: 10,305,091 (39.31 MB)
Non-trainable params: 58,331,648 (222.52 MB)