GIS Project

Renewable Energy Site Selection for Efficient Power Generation in Karnataka

Visit website

IMT2021038 - Ansh Avi Khanna
IMT2021098 - Ritik Kumar Gupta
IMT2021105 - Pranav Bhutada

Problem Statement

Develop a GIS-based solution to identify and recommend optimal sites for wind and solar energy projects in Karnataka, ensuring:

- *Efficient power generation.*
- *Maximized ROI for investors.*
- *Sustainable development.*

Current Challenges

01

Lack of systematic, data-driven approaches for selecting renewable energy sites.

02

Challenges in analyzing diverse datasets including weather patterns, geographical features, and infrastructure proximity.

03

Inefficient site selection resulting in suboptimal energy output and underperforming projects.

04

Increased costs and underutilized potential leading to poor ROI for investors and missed opportunities in renewable energy development.

Solution Overview

01 Identifying Optimal Locations

Use GIS and MCDM techniques to find the best sites for wind and solar energy projects in Karnataka.

02 Prioritizing Suitability Criteria

Apply AHP to assign weights to key factors like weather patterns, terrain, land use, and proximity to infrastructure.

03 Visualizing Results

Display the identified wind and solar energy locations on an interactive map using a React app.

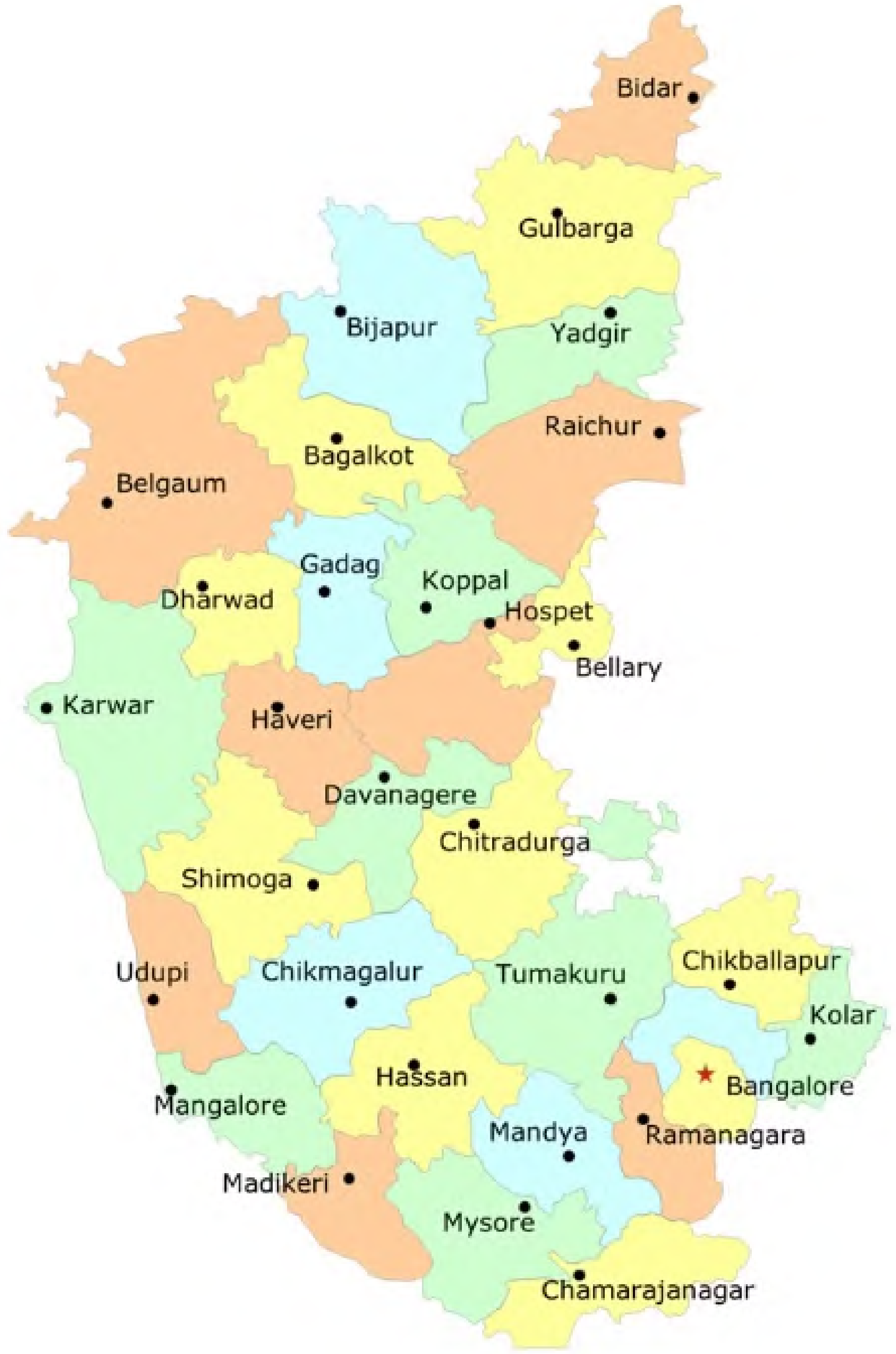
04 Delivering an Intuitive Platform

The *React app* provides an easy-to-use interface for stakeholders to explore the best locations for wind and solar energy development.

Data Collection

Karnataka Districts

We have collected data for 27 districts.



Bagalkot	Chitradurga	Kolar
Bangalore Rural	Dakshin Kannad	Koppal
Bangalore Urban	Davanagere	Mandya
Belgaum	Dharwad	Mysore
Bellary	Gadag	Raichur
Bidar	Gulbarga	Shimoga
Bijapur	Hassan	Tumkur
Chamrajnagar	Haveri	Udupi
Chikmagalur	Kodagu	Uttar Kannad

Data Collection

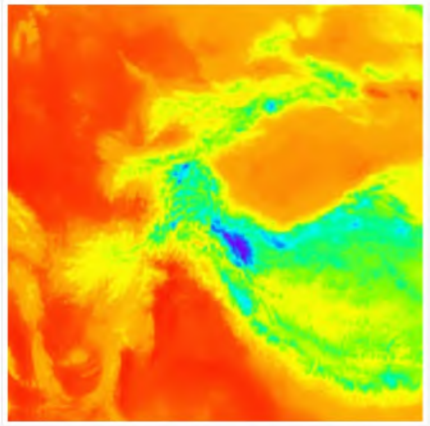
Open-source APIs and Platforms

Open-source APIs and Platforms

1. OpenStreetMap
2. Google Earth Engine
   a. ECMWF/ERA5_LAND/DAILY_AGGR
   b. ESA/WorldCover/v200
3. Open Elevation API
4. sunrise-senset.org

Data (day-wise) of 11 years [2013 - 2023] collected for 27 districts.

ERA5-Land Daily Aggregated - ECMWF Climate Reanalysis



Dataset Availability

1950-01-02T00:00:00Z–2024-11-22T00:00:00Z

Dataset Provider

Daily Aggregates: Google and Copernicus Climate Data Store

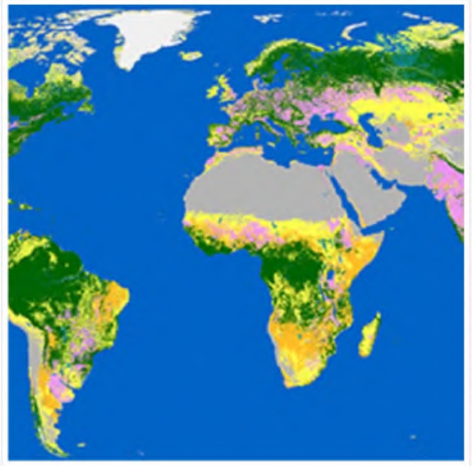
Earth Engine Snippet

`ee.ImageCollection("ECMWF/ERA5_LAND/DAILY_AGGR")`

Tags

cds, climate, copernicus, ecmwf, era5-land, evaporation, heat, lakes, precipitation, pressure, radiation, reanalysis, runoff, snow, soil-water, temperature, vegetation, wind

ESA WorldCover 10m v200



Dataset Availability

2021-01-01T00:00:00Z–2022-01-01T00:00:00Z

Dataset Provider

ESA/VITO/Brockmann Consult/CS/GAMMA Remote Sensing/IIASA/WUR

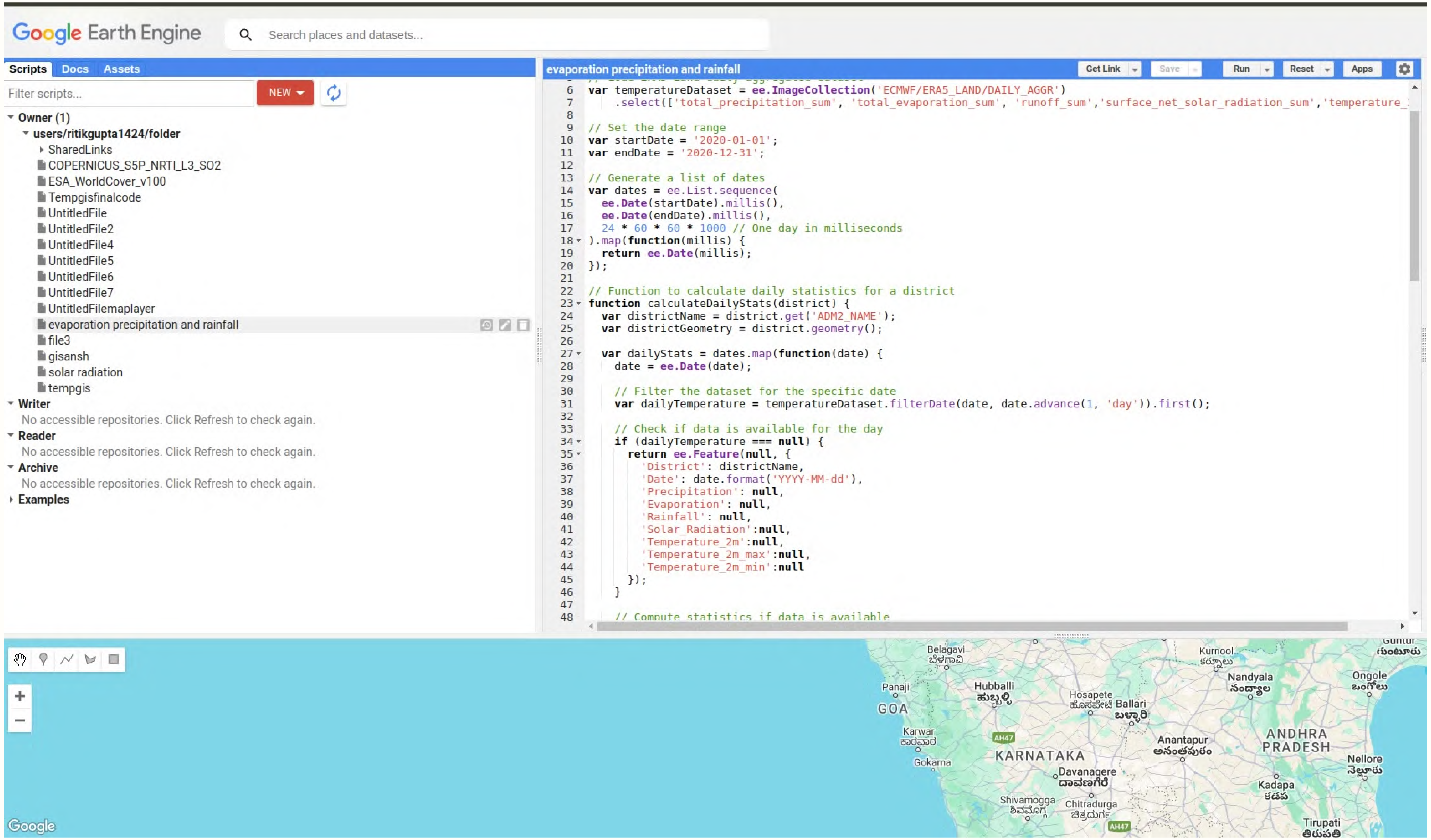
Earth Engine Snippet

`ee.ImageCollection("ESA/WorldCover/v200")`

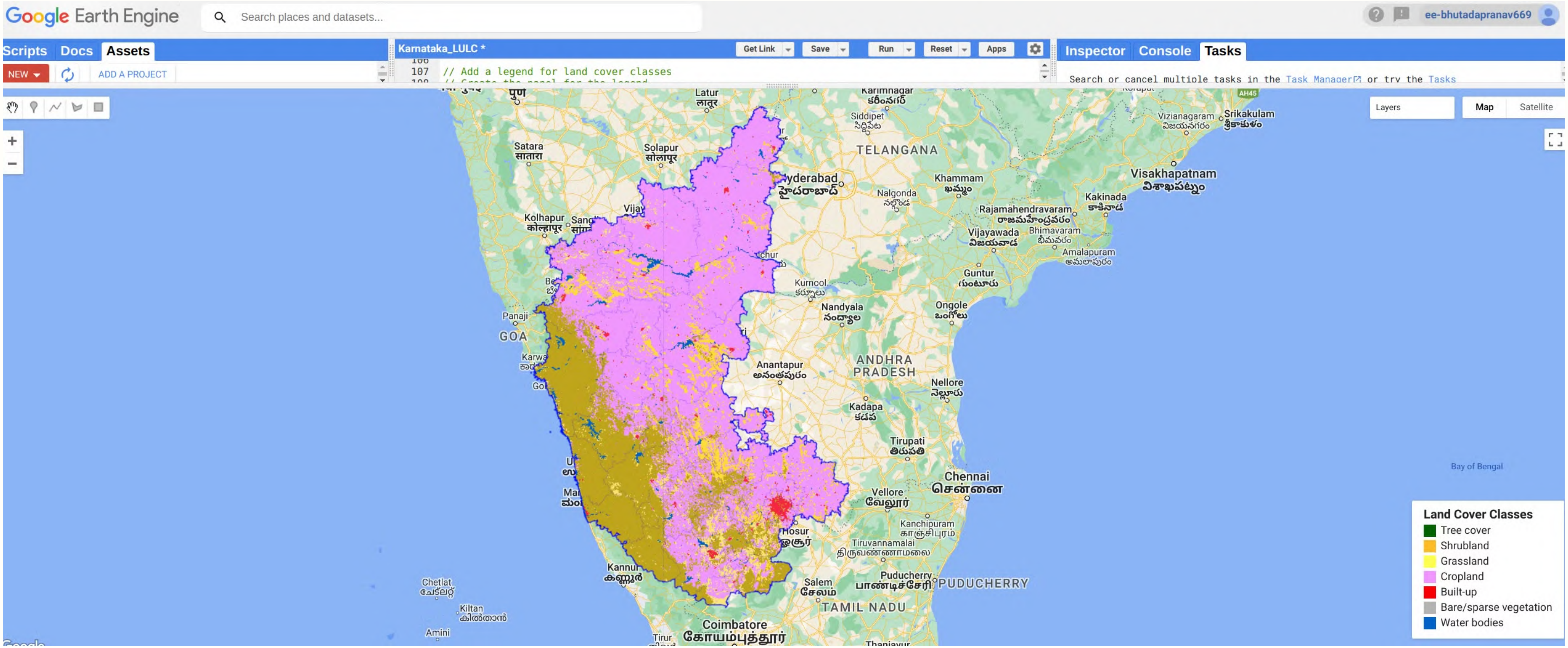
Tags

esa, landcover, landuse, sentinel1-derived, sentinel2-derived

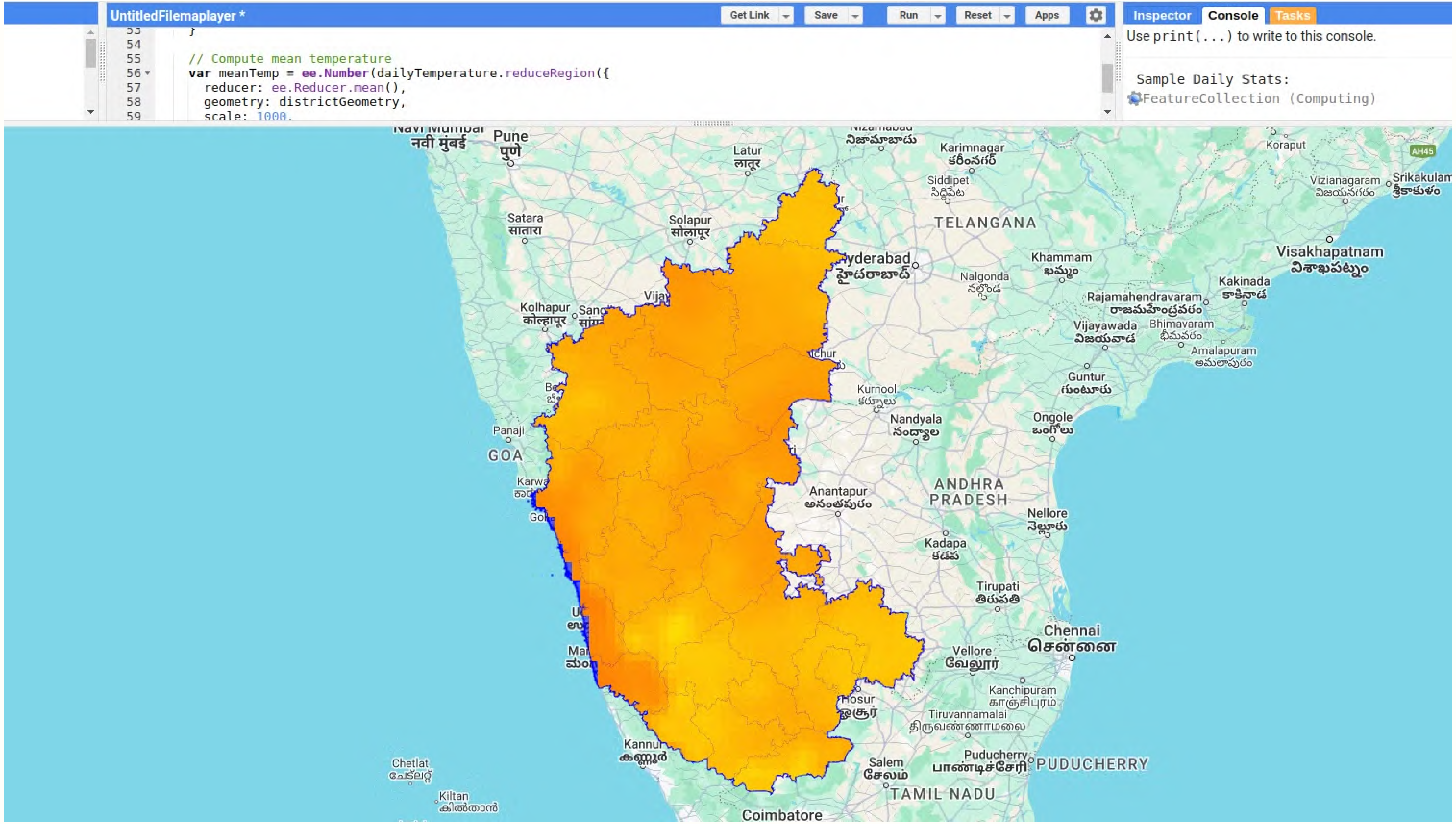
Google Earth Engine: Data Collection



Google Earth Engine: Data Collection



Google Earth Engine: Data Collection



Data Processing

Features

- Date
- Wind u component
- Wind v component
- Precipitation
- Evaporation
- Rainfall
- Solar Radiation
- Temperature_2m
- Temperature_2m_max
- Temperature_2m_min
- Day Length
- Elevation
- Population Density
- Slope
- Highways
- LULC (Land Use Land Cover)
- City Encoded
- Windspeed_10m

Technical Tools and Technologies

- Programming Languages :
  - Python
  - JavaScript
- Platforms :
  - Google Earth Engine
- Libraries :
  - Pandas
  - NumPy
  - Folium
  - Requests
  - Overpy
  - Geopy
- APIs :
  - OpenStreetMap
  - Open Elevation

What is AHP? and Why use it?

- **Analytical Hierarchy Process (AHP)** is a structured decision-making methodology used to solve complex problems by breaking them down into smaller, manageable components.

- It helps **compare various criteria** (e.g., wind speed, solar irradiance, terrain) by assigning relative importance to each, allowing for prioritized decision-making.

- **Handles Multiple, Conflicting Factors**: AHP allows the integration of both qualitative and quantitative factors (e.g., wind speed, terrain, infrastructure) and helps resolve conflicts between competing criteria, such as land availability vs. energy generation potential.

2.6.2. Scenario 2: Analytical Hierarchy Process (AHP)

The AHP is one of the most widely used MCDM methods [44,57] to solve different problems with different approaches [17,58]. The AHP is a mathematical approach developed by Saaty in 1977 [23]. This method reduces complex decisions to a series of side-by-side comparisons. In addition, the method allows checking the consistency of the decision, thus reducing bias in decision making [59].

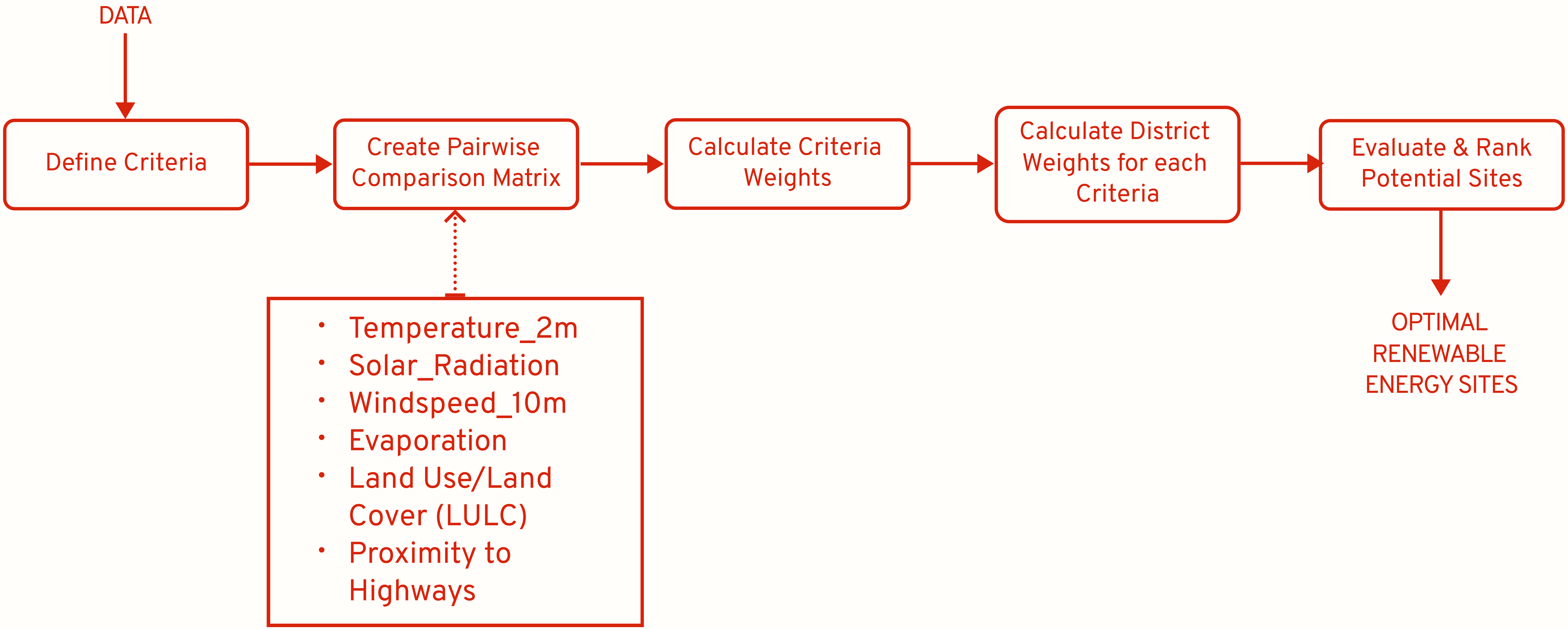
At the beginning of each AHP process, a goal, as well as the alternatives, are defined, and the criteria are selected. A pair-wise comparison matrix (A) is then generated. For instance, Equation (1) represents a comparison matrix when the criteria are three (a, b, and c).

$$A = \begin{bmatrix} 1 & a & b \\ \frac{1}{a} & 1 & c \\ \frac{1}{b} & \frac{1}{c} & 1 \end{bmatrix} \quad (1)$$

If n is the number of criteria, then the matrix (A) will be a matrix where each entry a_{ij} of the matrix describes the importance of the i_{th} criterion with respect to the j_{th} criterion. The relative importance of the two criteria is measured according to a numerical scale from 1 to 9 (Table 4).

https://www.researchgate.net/publication/349498794_A_Regional_GIS-Assisted_Multi-Criteria_Evaluation_of_Site-Suitability_for_the_Development_of_Solar_Farms

Analytic hierarchy process (AHP) Framework



AHP - Wind Parameter Weight Calculation for Site Selection

Generating Pairwise Comparison Values:

- To apply the **Analytical Hierarchy Process (AHP)**, a set of pairwise comparison values between wind-related factors and similarly for solar-related factors. These values are assigned on a scale from 1 to 9, where 1 means equal importance and 9 means extreme importance of one criterion over another.
- This process is crucial for determining how different factors (e.g., wind speed, terrain, proximity to infrastructure) compare in terms of their significance to site suitability.

Table 4. Values used in the pair-wise comparison to evaluate the suitability of sites for the development of solar farms [23].

Verbal Judgments of Preferences between Alternatives	Numerical Rating	Explanation
Extremely preferred	9	The evidence favoring one criterion over another is of the highest possible order of affirmation.
Very strongly preferred	7	A criterion is favored very strongly, and its dominance is demonstrated in practice.
Strongly preferred	5	Experience and judgment strongly favor one criterion over another.
Moderately preferred	3	Experience and judgment slightly favor one criterion over another.
Equally preferred	1	Two criteria contribute equally to the objective.
Intermediate values	2, 4, 6, 8	When compromise is needed.

AHP - Wind Parameter Weight Calculation for Site Selection

Pairwise Comparison Matrix:

- The generated comparison values are then paired with **wind-related factors** and **solar-related factors**, such as wind speed, terrain, solar irradiation, mean temperature and other environmental conditions that influence wind and solar energy potential.
- A **comparison matrix** is formed, where each factor is compared with every other factor, generating a set of pairwise comparison scores (windComparison and solarComparison). These are the inputs for the AHP model.

Criteria	LULC	DMR	SI	AMT	VP	Slope	Aspect	WS	Soil Texture	Landform
LULC	1	1/6	1/7	1/4	2	1/7	1/9	1/2	1/2	1/4
DMR	6	1	1/5	1/2	2	1/2	1/4	2	2	1/2
SI	7	5	1	2	7	2	2	6	9	2
AMT	4	2	1/2	1	3	2	1/2	4	5	1/3
VP	1/2	1/2	1/7	1/3	1	1/5	1/5	2	4	1/6
Slope	7	2	1/2	1/2	5	1	1/3	5	6	1/2
Aspect	9	4	1/2	2	5	3	1	3	4	2
Wind	2	1/2	1/6	1/4	1/2	1/5	1/3	1	4	1/7
Soil Texture	2	1/2	1/9	1/5	1/4	1/6	1/4	1/4	1	1/5
Landform	4	2	1/2	3	6	2	1/2	7	5	1

LULC = Land use/land cover, DMR = Distance to main roads, SI = Solar irradiance, AMT = Annual mean temperature, VP = Vapor pressure, WS = Wind speed.

	A_1	A_2	A_3	A_4	A_5	A_6
A_1	1	5	4	8	3	6
A_2	1/5	1	1/4	4	1/3	2
A_3	1/4	4	1	4	3	2
A_4	1/8	1/4	1/4	1	1/6	1/3
A_5	1/3	3	1/3	6	1	4
A_6	1/6	1/2	1/2	3	1/4	1

Where A1, A2, A3, A4, A5 and A6 wind speed, land cover or land use, slope, distance from urban places, power lines and roads respectively.

AHP - Wind Parameter Weight Calculation for Site Selection

AHP Analysis:

- The *ahpy.Compare* function is used to perform the AHP analysis. This method evaluates the matrix and calculates the relative weights of each factor, which signifies their importance in the decision-making process.
- The AHP model considers the relative importance of each wind and solar-related factor, using the pairwise comparison values, to provide an optimal decision.

Consistency Ratio Check:

- A critical part of the AHP process is ensuring that the pairwise comparisons are consistent. The consistency ratio measures how logically consistent the pairwise comparisons are. If the error is too high, it means that the comparisons may have been inconsistent and need to be adjusted.
- In the code, the consistency ratio (error) is continuously monitored. If the ratio is above a threshold (0.1), new comparisons are generated until the error falls below the acceptable limit.

Districts with Latitude and Longitude

ADM1_NAME	ADM2_CODE	ADM2_NAME	DISP_AREA	EXP2_YEAR	STATUS	STR2_YEAR	Shape_Area	Shape_Leng	latitude	longitude
Karnataka	17679	Belgaum	NO	3000	Member State	1000	1.126517	9.391778	16.117203	74.827594
Karnataka	17680	Bellary	NO	3000	Member State	1997	0.708337	6.838866	15.105000	76.530074
Karnataka	17681	Bidar	NO	3000	Member State	1000	0.463418	4.668253	17.950116	77.223345
Karnataka	17682	Bijapur	NO	3000	Member State	1997	0.888710	6.235993	16.791240	75.953502
Karnataka	17683	Chikmagalur	NO	3000	Member State	1000	0.600027	4.978839	13.449441	75.689389
Karnataka	17685	Dakshin Kannad	NO	3000	Member State	1997	0.379897	4.254837	12.833083	75.266769
Karnataka	17686	Dharwad	NO	3000	Member State	1997	0.358696	4.037479	15.386440	75.156243
Karnataka	17687	Gulbarga	NO	3000	Member State	1000	1.377263	8.978818	17.051385	76.881373
Karnataka	17688	Hassan	NO	3000	Member State	1000	0.565800	5.528016	12.989513	76.104821
Karnataka	17689	Kodagu	NO	3000	Member State	1000	0.340391	3.897782	12.318636	75.799240
Karnataka	17690	Kolar	NO	3000	Member State	1000	0.684285	6.826551	13.355679	78.012571
Karnataka	17691	Mandya	NO	3000	Member State	1000	0.409784	4.424021	12.603550	76.790633
Karnataka	17692	Mysore	NO	3000	Member State	1997	0.523124	5.510739	12.202152	76.436149
Karnataka	17693	Raichur	NO	3000	Member State	1997	0.712555	4.850781	16.085568	76.889890
Karnataka	17694	Shimoga	NO	3000	Member State	1997	0.647171	5.892604	14.052066	75.176149
Karnataka	17695	Tumkur	NO	3000	Member State	1000	0.884369	9.155249	13.513107	76.941107
Karnataka	17696	Uttar Kannand	NO	3000	Member State	1000	0.863034	8.612038	14.787415	74.623735
Karnataka	70157	Bagalkot	NO	3000	Member State	1997	0.553822	5.777876	16.217983	75.626940
Karnataka	70158	Bangalore Rural	NO	3000	Member State	1000	0.483937	7.177115	12.887042	77.422237
Karnataka	70159	Bangalore Urban	NO	3000	Member State	1000	0.181517	2.772938	12.942246	77.586849
Karnataka	70160	Chamrajnagar	NO	3000	Member State	1997	0.469210	5.050475	11.948784	77.090106
Karnataka	70161	Chitradurga	NO	3000	Member State	1997	0.705669	5.785206	14.161754	76.512237
Karnataka	70162	Davanagere	NO	3000	Member State	1997	0.553736	5.593154	14.351222	75.931068
Karnataka	70163	Gadag	NO	3000	Member State	1997	0.389915	4.902701	15.427535	75.666688
Karnataka	70164	Haveri	NO	3000	Member State	1997	0.402374	4.074156	14.734347	75.418637
Karnataka	70165	Koppal	NO	3000	Member State	1997	0.468086	4.852234	15.557948	76.220409
Karnataka	70166	Udupi	NO	3000	Member State	1997	0.324776	4.473987	13.463712	74.883476

gis (1).ipynb

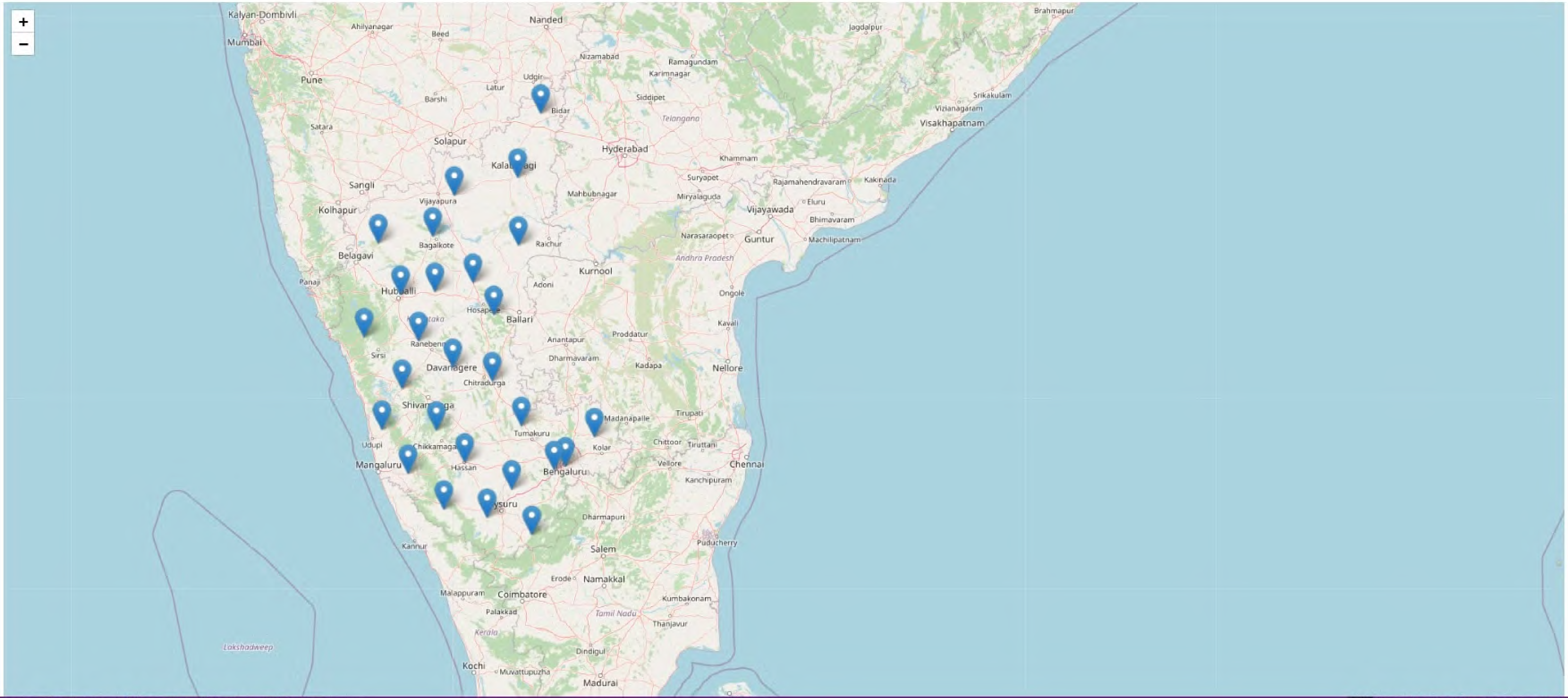
home > ritik > Downloads > gis (1).ipynb > ...

Generate + Code + Markdown Run All Clear All Outputs Outline ... Select Kernel

```python
    return city_map

# Show the locations of cities in city_dict on a map
show_locations_on_map(city_dict)
```

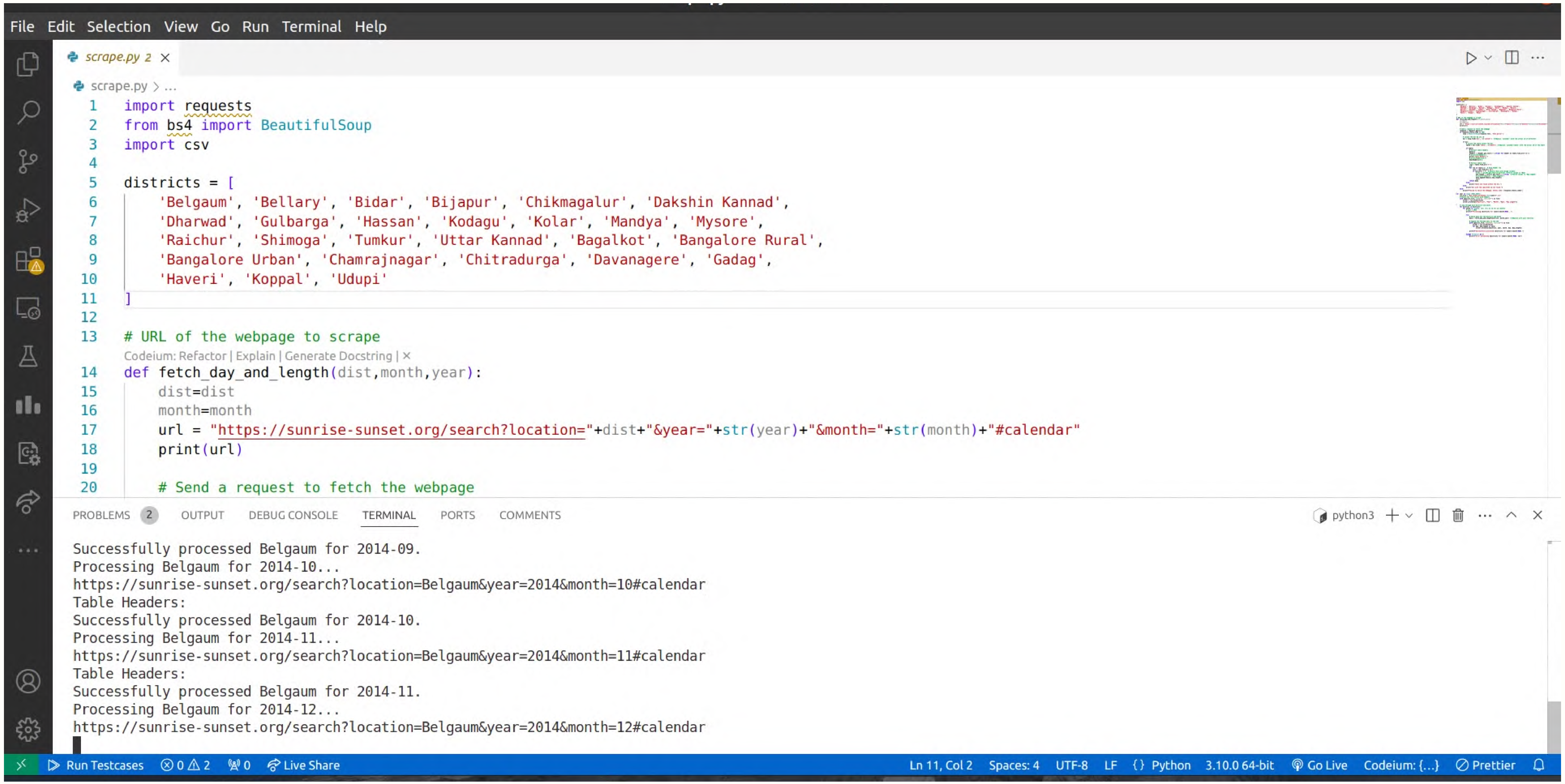
[5] Python



Run Testcases CodeTogether ⊗ 2 ⚠ 22 0 Live Share Activating Extensions... Spaces: 4 Cell 1 of 48 Go Live

Try Pitch

Web scraping code for length of day



Extracting Elevation and Slope

```python
# Define a function to calculate the slope
def calculate_slope(point1, point2):
    lat1, lon1 = point1
    lat2, lon2 = point2
    distance = math.sqrt((lat2 - lat1) ** 2 + (lon2 - lon1) ** 2)
    elevation_difference = elevations_dict[point2] - elevations_dict[point1]
    slope = math.degrees(math.atan(elevation_difference / distance))
    return slope

# Initialize an empty dictionary to store results
city_results = {}

# Iterate through each city in city_dict
for city, (lat, lon) in city_dict.items():
    nearby_points = []  # Store nearby point data
    elevations = []     # Store elevations for nearby points

    # Create 8 nearby points by adding/subtracting 0.01 from latitude and longitude
    for i in range(-1, 2):
        for j in range(-1, 2):
            new_lat = lat + i * 0.01
            new_lon = lon + j * 0.01
            nearby_points.append((new_lat, new_lon))

    # Fetch elevations for the nearby points and store them in a dictionary
    elevations_dict = {}
    for point in nearby_points:
        lat, lon = point
        url = f'https://api.open-elevation.com/api/v1/lookup?locations={lat},{lon}'
        response = requests.get(url)
        if response.status_code == 200:
            data = response.json()
            elevation = data['results'][0]['elevation']
            elevations_dict[point] = elevation

    # Calculate the variance of elevations for nearby points
    elevations = list(elevations_dict.values())
    variance = calculate_variance(elevations)

    # Calculate the slope using nearby points
    central_point = (lat, lon)
    slope = 0.0
    for point in nearby_points:
        if point != central_point:
            point_slope = calculate_slope(central_point, point)
            slope += point_slope
    slope /= len(nearby_points) - 1  # Calculate the average slope

    # Store results in the city_results dictionary
    city_results[city] = {
        'variance': variance,
        'slope': slope,
    }
```

[10]

Run Testcases CodeTogether ⊗ 2 ⚠ 22 ⑂ 0 Live Share

[10]

```
City: Belgaum
Variance of Elevations: 3851.1111111111113
Slope: 45.0259757018606 degrees
City: Bellary
Variance of Elevations: 246.22222222222223
Slope: 67.42181164463624 degrees
City: Bidar
Variance of Elevations: 15.358024691358025
Slope: 56.064149861014194 degrees
City: Bijapur
Variance of Elevations: 38.00000000000001
Slope: 22.606478133784393 degrees
City: Chikmagalur
Variance of Elevations: 1151.5555555555557
Slope: -89.80723235107253 degrees
City: Dakshin Kannad
Variance of Elevations: 217.06172839506172
Slope: 89.96650251078262 degrees
City: Dharwad
Variance of Elevations: 32.24691358024692
Slope: 89.86522216547199 degrees
City: Gulbarga
Variance of Elevations: 85.35802469135803
Slope: -67.42636495544444 degrees
City: Hassan
...
Slope: -78.63907954791058 degrees
City: Udupi
Variance of Elevations: 119.55555555555556
Slope: -67.40869172724405 degrees
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

https://open-elevation.com/

Closest Highway For Each District

1	city	closest highway
2	Belgaum	2742.63155180908
3	Bellary	15836.5024695069
4	Bidar	10935.4066511948
5	Bijapur	20000
6	Chikmagalur	8370.72520451014
7	Dakshin Kannad	1344.49071135982
8	Dharwad	2094.2775597468
9	Gulbarga	8728.50997655084
10	Hassan	1551.90453176854
11	Kodagu	12514.0863811858
12	Kolar	846.715469342277
13	Mandya	11720.7864968229
14	Mysore	657.083203494251
15	Raichur	20000
16	Shimoga	11152.1113748654
17	Tumkur	6789.65338818251
18	Uttar Kannand	12183.2292393991
19	Bagalkot	3833.67005387741
20	Bangalore Rural	529.243692715744
21	Bangalore Urban	184.984489873634
22	Chamrajnagar	20000
23	Chitradurga	8097.97097371858
24	Davanagere	8861.90757015773
25	Gadag	3393.24034542422
26	Haveri	4099.92459017976
27	Koppal	20000
28	Udupi	5745.11796713281

```python
import overpy
from geopy.distance import geodesic


def get_distance_to_closest_highway(latitude, longitude):
    api = overpy.Overpass()

    query = f"""
    [out:json];
    (
      node["highway"](around:20000,{latitude},{longitude});
      way["highway"](around:20000,{latitude},{longitude});
      rel["highway"](around:20000,{latitude},{longitude});
    );
    out center;
    """

    result = api.query(query)

    closest_distance = float('inf')
    for item in result.nodes + result.ways + result.relations:
        if isinstance(item, overpy.Node) and hasattr(item, 'lat') and hasattr(item, 'lon'):
            coord = (item.lat, item.lon)
            distance = geodesic((latitude, longitude), coord).meters
            closest_distance = min(closest_distance, distance)

    return closest_distance

# Initialize an empty dictionary to store the distance to the closest highway for each city
closest_highway_dict = {}

# Iterate through each city in city_dict
for city, (lat, lon) in city_dict.items():
    distance_to_highway = get_distance_to_closest_highway(lat, lon)
    closest_highway_dict[city] = distance_to_highway

# Print the distance to the closest highway for each city
for city, distance in closest_highway_dict.items():
    print(f"Distance to the closest highway from {city}: {distance} meters")
```

[26]

Solar Radiation

Radiation Data

```
df1 = pd.read_csv('/home/pranav/Desktop/GIS/merged1.csv')
df1['Solar_Radiation']=df1['Solar_Radiation']/1e6

# Group the data by 'City' and calculate the mean of 'Shortwave_radiation_sum'
average_radiation_by_city = df1.groupby('District')['Solar_Radiation'].median()

# Print the result
print(average_radiation_by_city)
```

[8]

```
District
Bagalkot          16.386963
Bangalore Rural   16.005395
Bangalore Urban   16.129122
Belgaum           16.353020
Bellary           16.088431
Bidar             16.191795
Bijapur           16.535413
Chamrajnagar      16.029236
Chikmagalur       15.778635
Chitradurga       15.915955
Dakshin Kannad    15.715203
Davanagere        15.942206
Dharwad           16.128209
Gadag             16.230383
Gulbarga          16.117295
Hassan            15.633436
Haveri            15.885848
Kodagu            15.667182
Kolar             15.933400
Koppal            15.958163
Mandya            15.880734
Mysore            15.626788
Raichur           16.011917
Shimoga           16.062303
Tumkur            15.912711
Udupi             16.335249
Uttar Kannand     16.240204
Name: Solar_Radiation, dtype: float64
```

	District	Percentage High Radiation Days
0	Bagalkot	60.144386
1	Bangalore Rural	54.368932
2	Bangalore Urban	54.294249
3	Belgaum	57.455813
4	Bellary	55.339806
5	Bidar	59.049042
6	Bijapur	63.355738
7	Chamrajnagar	55.240229
8	Chikmagalur	52.750809
9	Chitradurga	53.945731
10	Dakshin Kannad	52.601444
11	Davanagere	53.871048
12	Dharwad	56.285785
13	Gadag	56.957929
14	Gulbarga	59.173513
15	Hassan	52.103560
16	Haveri	53.572318
17	Kodagu	52.651232
18	Kolar	54.045307
19	Koppal	54.991287
20	Mandya	53.671894
21	Mysore	51.779935
22	Raichur	57.231765
23	Shimoga	54.344038
24	Tumkur	53.696789
25	Udupi	57.679861
26	Uttar Kannand	55.763007

Land Use and Land Cover Patterns

B	C	D	E	F	G	H	I	J
District	Bare/sparse vegetation	Built-up	Cropland	Grassland	Shrubland	Total_Area_km2	Tree cover	Water bodies
Belgaum	67.4546843574803	244.043122391702	7790.99719806537	1253.20805603947	1240.08651932604	13380.8415102258	2566.03007800592	170.910187486151
Bellary	88.4115213283118	173.304188816539	5624.50014240878	220.029495830189	1836.83428410047	8455.4441547441	355.265972120614	87.9219404086925
Bidar	14.361831660671	114.94265037755	4426.91243119724	281.605657782971	337.1470483903	5450.92546338607	180.533992657565	77.9588562792236
Bijapur	118.225456711799	177.222220631067	8399.83492961747	973.235908276449	290.767900160806	10519.6534805948	325.699046829387	193.228723123881
Chikmagalur	10.2029182672745	81.3579794522831	1246.54780817153	932.300368532806	227.025565116598	7215.43278571646	4594.63679623303	96.2002436815345
Dakshin Kannad	5.65471596258659	99.8104036756633	34.4053271273409	149.610708810542	0.025712743950728	4579.82397796899	4223.15640515947	47.0156395489201
Dharwad	12.7483802176081	118.453856242881	3235.90645609202	142.822308140214	272.336688834833	4276.06779197312	464.308191184499	14.2293465829345
Gulbarga	71.2888243965505	273.981713781016	13643.8491096657	560.521947581233	961.101353443681	16280.0146098288	513.193742585302	201.27076864711
Hassan	10.7239387829587	152.040411213628	2696.38440262283	919.084725319106	241.624447071549	6816.68067378734	2667.43606626586	102.666225577589
Kodagu	0.838203908589112	25.4495299013571	121.226306172857	347.313589528623	0.003709761234687	4111.78155983967	3586.88590985584	14.1331310311959
Kolar	63.9597051440251	222.484514800824	5287.77927091563	313.097658988593	1733.74982959911	8231.81131746279	533.651869787506	45.6956020721574
Mandya	11.6208728992534	178.723275022457	1534.30683480624	1209.2515187254	207.625447707226	4944.58993061095	1673.22567416785	110.60718311561
Mysore	5.71317099162245	238.581756120427	2873.85213318271	1136.2371632254	24.2893523156217	6321.91434689691	1894.54028736429	123.737494728804
Raichur	55.1570303695142	136.766978233645	6986.54686614983	173.295008198434	826.696512063102	8465.29777528759	177.859386929924	79.2474511678625
Shimoga	9.42296834246183	74.9806220438131	1484.60791375156	1002.15118172677	25.8529097386647	7762.31110772206	4856.89893173675	279.54501493927
Tumkur	49.1793336904924	255.15891303613	5061.52262169033	481.930350888273	2452.20084052283	10631.7047950317	2204.6119073992	85.1625627062104
Uttar Kannand	13.3216455780744	40.8185933264154	687.819083332513	570.522256678476	75.0158149948177	10317.214493706	8614.13046816314	254.765444023995
Bagalkot	36.9056292813456	123.761586977	4269.69815072031	368.844275459774	1122.38197070564	6575.10098598859	419.692505136852	186.557198319413
Bangalore Rural	36.9177275751379	248.371916206455	2350.20343463004	890.062773932963	867.724305169656	5832.73740194214	1378.10544249085	38.3777568808424
Bangalore Urban	27.8512230087278	642.310282826103	644.908651562562	190.458074404253	278.363177058093	2187.31234525566	374.441766584205	20.0805109381484
Chamrajnagar	6.18758330115826	79.6243047338505	1539.30275511217	1333.4126594846	319.505768195538	5675.74800070516	2363.79146465722	11.7925294451319
Chitradurga	57.3013542048214	132.187845646757	4683.7779749787	373.959823391122	2201.66009232893	8459.8372372231	899.70778719942	80.1000163280511
Davanagere	15.4143406593362	147.515082377629	3804.40268128261	266.074209451076	726.948839104373	6632.83118751818	1584.65963101536	61.4677887450186
Gadag	18.1909817124799	83.9843246781084	3664.09630099375	300.703511038907	454.540586506055	4647.27333166064	88.4824378857027	20.5089547015293
Haveri	17.5287739887531	108.414362859694	3667.63812675115	175.742127285544	337.015338040674	4811.45614459711	451.604009557715	35.8620983569429
Koppal	52.3731314012573	102.397545714188	4402.72222984512	196.816493473575	612.48442237372	5575.47827677929	88.537901833682	61.8011405779815
Udupi	6.75622200554813	59.4806983089955	182.231069139138	294.327503554857	0.060550299959071	3905.24650000157	3296.57861266864	45.7761993069366

Land Use and Land Cover Patterns

Criteria	Suitability Levels and Scores					
	Highly Suitable	Suitable	Moderately Suitable	Less Suitable	Not Suitable	Reference
	5	4	3	2	1	
Wind speed (m/sn)	>6	5–6	4–5	3–4	<3	(Saraswat et al., 2021, Nagababu et al., 2022)
Wind power density (W/m^2)	>300	250–300	150–250	100–150	<100	(Ayodele et al., 2018)
Rainfall (mm)	<300	300–500	500–600	600–700	>700	(Al-Shabeeb et al. 2016)
Elevation (m)	<500	500–1000	1000–1500	1500–2000	>2000	(Saraswat et al., 2021, Nagababu et al., 2022)
Slope (degree)	0–6	6–9	9–12	12–15	>15	(Saraswat et al., 2021, Nagababu et al., 2022)
Aspect	E, SE, Flat	NE	N, S	NW, SW	W	(Gigovic et al., 2017)
Land use	Natural and semi-natural areas		Agriculture	Forest	Settlement, artificial zones, water bodies	(Al-Shabeeb et al., 2016, Ali et al., 2019)
Population density (p/km^2)	<10	10–50	50–90	90–110	>110	(Gigovic et al., 2017)

	District	Land_use_patterns
0	Belgaum	1.056359
1	Bellary	0.881657
2	Bidar	0.468981
3	Bijapur	0.559861
4	Chikmagalur	1.890443
5	Dakshin Kannad	1.979869
6	Dharwad	0.553759
7	Gulbarga	0.395389
8	Hassan	1.434566
9	Kodagu	2.083377
10	Kolar	0.944723
11	Mandya	1.790405
12	Mysore	1.333417
13	Raichur	0.442940
14	Shimoga	1.782672
15	Tumkur	1.306495
16	Uttar Kannand	1.918026
17	Bagalkot	0.886607
18	Bangalore Rural	1.554554
19	Bangalore Urban	1.123392
20	Chamrajnagar	1.945911
21	Chitradurga	1.197356
22	Davanagere	0.976373
23	Gadag	0.605982
24	Haveri	0.558529
25	Koppal	0.540095
26	Udupi	1.996717

CR and CI for Wind and Solar

```
windValues = finalWindValues
windComparison = dict(zip(windPairs, windValues))
print(windComparison)
```

```
{('windspeed_10m', 'LULC'): 1, ('windspeed_10m', 'slope'): 4, ('windspeed_10m', 'highways'): 6, ('LULC', 'slope'): 0.25, ('LULC', 'highways'): 2, ('slope', 'highways')
```

```
wind  = ahpy.Compare(name='Wind', comparisons=windComparison, precision=3, random_index='saaty')

print(wind.target_weights)

weightsWind = wind.target_weights

print(wind.consistency_ratio)
```

[119]

```
{'windspeed_10m': 0.465, 'slope': 0.27, 'LULC': 0.184, 'highways': 0.081}
0.015999999999999996
```

```
solarValues = finalSolarValues
solarComparison = dict(zip(solarPairs, solarValues))
print(solarComparison)
```

[78] Python

```
{('Temperature_2m', 'Solar_Radiation'): 0.5, ('Temperature_2m', 'windspeed_10m'): 4, ('Temperature_2m', 'Evaporation'): 3, ('Temperature_2m', 'LULC'): 4, ('Temperature_2m', 'highways'): 2, ('S
```

```
solar  = ahpy.Compare(name='Solar', comparisons=solarComparison, precision=3, random_index='saaty')

print(solar.target_weights)

weightsSolar = solar.target_weights

print(solar.consistency_ratio)
```

[79] Python

```
{'Solar_Radiation': 0.445, 'Temperature_2m': 0.22, 'highways': 0.147, 'Evaporation': 0.067, 'windspeed_10m': 0.064, 'LULC': 0.057}
0.073
```

Comparison Matrix: Wind and Solar

```
import pandas as pd
from itertools import combinations


# Extract city names and corresponding temperature values
city_names = avg['District']
temperature_values = avg['windspeed_10m']

# Initialize an empty matrix to store comparisons
matrix_size = len(city_names)
comparison_matrix = [[0] * matrix_size for _ in range(matrix_size)]

for i, j in combinations(range(matrix_size), 2):
    comparison_matrix[i][j] = temperature_values[j] / temperature_values[i]
    comparison_matrix[j][i] = 1/comparison_matrix[i][j]

for i in range(matrix_size):
  comparison_matrix[i][i] = 1

flattened_upper_triangular = [comparison_matrix[i][j] for i in range(len(comparison_matrix)) for j in range(i + 1, len(comparison_matrix[i]))]

print(flattened_upper_triangular)
print(len(flattened_upper_triangular))

wind_city_Pairs = list(itertools.combinations(city_names, 2))
print(wind_city_Pairs)
len_windspeed_10m_max=len(wind_city_Pairs)
print(len_windspeed_10m_max)

wind_windspeed_10m_max = dict(zip(wind_city_Pairs, flattened_upper_triangular))

windspeed_10m_max  = ahpy.Compare(name='windspeed_10m_max', comparisons=wind_windspeed_10m_max, precision=3, random_index='dd')

print(windspeed_10m_max.target_weights)

weightswindspeed_10m_max = windspeed_10m_max.target_weights

print(windspeed_10m_max.consistency_ratio)
```

Comparison Matrix: Wind and Solar

```python
# Initialize a dictionary to store the weighted sums
weighted_sums = {}

# Iterate through each city name
for city in city_names:
    weighted_sum = (
        weightstemperature_2m_mean[city] *weightsSolar ['Temperature_2m'] + weightsshortwave_radiation_sum[city]* weightsSolar ['Solar_Radiation'] +
        weightset0_fao_evapotranspiration[city]* weightsSolar ['Evaporation'] + highways_max.target_weights[city]*weightsSolar['highways']+
        LULC_max.target_weights[city]*weightsSolar['LULC'] + weightswindspeed_10m_max[city]*weightsSolar ['windspeed_10m']

    )
    weighted_sums[city] = weighted_sum


SolarRanking = pd.DataFrame(list(weighted_sums.items()), columns=['City', 'Value'])

# Sort the DataFrame by 'roi' in descending order
sorted_avg_solar = SolarRanking.sort_values(by='Value', ascending=False)

# Print the sorted DataFrame
sorted_avg_solar
```

[118]

Solar

```python
# Initialize a dictionary to store the weighted sums
weighted_sums = {}

# Iterate through each city name
for city in city_names:
    weighted_sum = (
        weightswindspeed_10m_max[city]*weightsWind ['windspeed_10m'] +slope_max.target_weights[city]*weightsWind['slope']+
        highways_max.target_weights[city]*weightsWind['highways']+LULC_max.target_weights[city]*weightsWind['LULC']
    )
    weighted_sums[city] = weighted_sum


WindRanking = pd.DataFrame(list(weighted_sums.items()), columns=['City', 'Value'])

# Sort the DataFrame by 'roi' in descending order
sorted_avg_wind = WindRanking.sort_values(by='Value', ascending=False)

# Print the sorted DataFrame
# print(sorted_avg_wind)
sorted_avg_wind
```

[117]

Wind

Results

[117]

	Unnamed: 0	City	Value
0	21	Gadag	0.262705
1	1	Bangalore Rural	0.066988
2	10	Dakshin Kannad	0.051434
3	2	Belgaum	0.049218
4	25	Udupi	0.036650
5	17	Kodagu	0.033190
6	14	Gulbarga	0.030238
7	5	Bidar	0.029360
8	22	Raichur	0.027247
9	12	Dharwad	0.026917
10	8	Chikmagalur	0.026745
11	18	Kolar	0.026486
12	26	Uttar Kannand	0.025469
13	15	Hassan	0.025466
14	16	Haveri	0.025437
15	7	Chamrajnagar	0.024761
16	13	Mysore	0.023365
17	19	Koppal	0.023276
18	23	Shimoga	0.022582
19	3	Bangalore Urban	0.021863
20	6	Bijapur	0.021513
21	0	Bagalkot	0.020540
22	24	Tumkur	0.020474
23	11	Davanagere	0.019924
24	20	Mandya	0.019792
25	4	Bellary	0.019406
26	9	Chitradurga	0.018901

Wind

	Unnamed: 0	City	Value
0	2	Raichur	0.085720
1	1	Kolar	0.049515
2	21	Mandya	0.046548
3	18	Belgaum	0.043129
4	10	Bidar	0.041831
5	15	Gadag	0.037687
6	12	Koppal	0.037405
7	16	Haveri	0.034862
8	13	Hassan	0.034429
9	3	Bangalore Rural	0.034259
10	14	Gulbarga	0.034188
11	25	Udupi	0.033900
12	17	Kodagu	0.033696
13	5	Dakshin Kannad	0.033438
14	22	Bangalore Urban	0.033391
15	8	Chikmagalur	0.033259
16	0	Bagalkot	0.033077
17	19	Dharwad	0.032621
18	24	Tumkur	0.032237
19	26	Uttar Kannand	0.032120
20	11	Davanagere	0.032057
21	9	Chitradurga	0.031996
22	23	Shimoga	0.031628
23	4	Bellary	0.031464
24	6	Bijapur	0.031339
25	7	Chamrajnagar	0.031299
26	20	Mysore	0.031244

Solar

Website

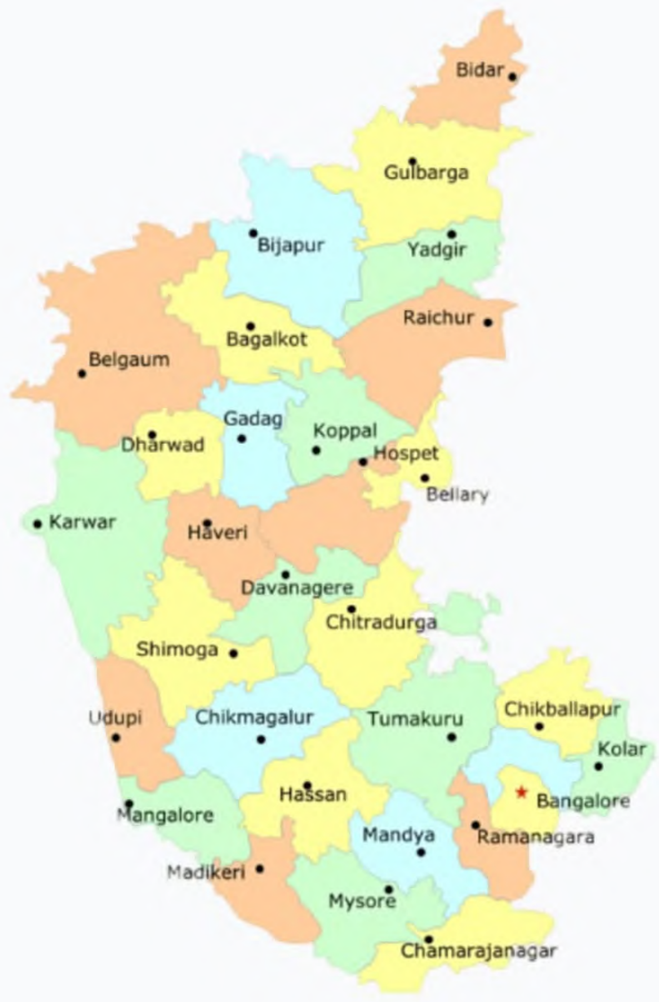
localhost:5173

Renewable Energy Site Selection

About Our Project

Our solution leverages data-driven insights to guide renewable energy site selection in Karnataka. By combining advanced geospatial analysis and strategic feature engineering, we provide comprehensive recommendations that facilitate efficient power generation and attractive returns on investment.

Go to Wind Map | Go to Solar Map



References

Solar PV power plant site selection using a GIS-AHP based approach with application in Saudi Arabia Hassan Z. Al Garni* , Anjali Awasthi

Wind farm site selection using geographic information system and fuzzy decision making model Gülay Demir , Muhammad Riaz , Muhammet Deveci

Optimal wind-solar site selection using a GIS-AHP based approach: A case of Tunisia Sassi Rekik , Souheil El Alimi

Multi-Criteria Decision-Making System for Wind Farm Site-Selection Using Geographic Information System (GIS): Case Study of Semnan Province, Iran

A Regional GIS-Assisted Multi-Criteria Evaluation of Site-Suitability for the Development of Solar Farms

Thank you