```
!pip install --upgrade numpy==1.23.5 joblib
```

```
Collecting numpy==1.23.5
  Using cached numpy-1.23.5.tar.gz (10.7 MB)
  Installing build dependencies ... done
  error: subprocess-exited-with-error

  × Getting requirements to build wheel did not run successfully.
  │ exit code: 1
  ╰─> See above for output.

  note: This error originates from a subprocess, and is likely not a
  Getting requirements to build wheel ... error
error: subprocess-exited-with-error

× Getting requirements to build wheel did not run successfully.
│ exit code: 1
╰─> See above for output.

note: This error originates from a subprocess, and is likely not a p
```

```
import joblib
model = joblib.load("solar_model.pkl")
print("Model loaded successfully!")
```

```
Model loaded successfully!
```

```
!pip install requests pandas matplotlib scikit-learn folium
```

```
Requirement already satisfied: requests in /usr/local/lib/python3.12
Requirement already satisfied: pandas in /usr/local/lib/python3.12/d
Requirement already satisfied: matplotlib in /usr/local/lib/python3.
Requirement already satisfied: scikit-learn in /usr/local/lib/pythor
Requirement already satisfied: folium in /usr/local/lib/python3.12/d
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/loca
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/pythor
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/
Requirement already satisfied: numpy>=1.26.0 in /usr/local/lib/pytho
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/pythor
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/pyth
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/py
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/pythor
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/p
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/p
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/pyt
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.1
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/py
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/pythor
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/pytho
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/li
Requirement already satisfied: branca>=0.6.0 in /usr/local/lib/pytho
Requirement already satisfied: jinja2>=2.9 in /usr/local/lib/python3
Requirement already satisfied: xyzservices in /usr/local/lib/python3
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/pyt
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12
```

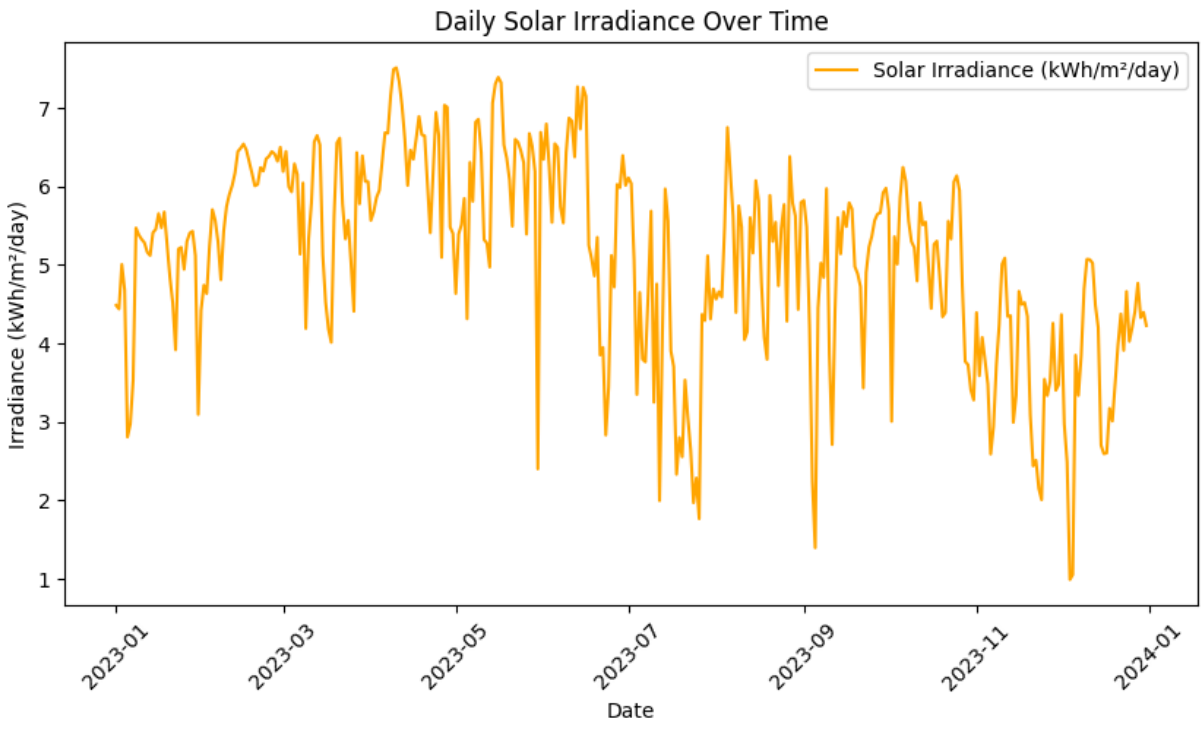
```
import requests
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import folium
```

```
# Example: latitude and longitude range (Andhra Pradesh)
latitude = 15.9
longitude = 79.7

url = f"https://power.larc.nasa.gov/api/temporal/daily/point?parame

response = requests.get(url)
data = response.json()

# Convert to DataFrame
df = pd.DataFrame(data['properties']['parameter'])
# The initial DataFrame from data['properties']['parameter'] alreac
# We just need to convert the index to datetime objects and rename
df.index = pd.to_datetime(df.index, format='%Y%m%d')
df = df.reset_index().rename(columns={'index': 'Date'})
df.head()
```

```
         Date  ALLSKY_SFC_SW_DWN    T2M   RH2M
0  2023-01-01             4.4837  24.83  77.67
1  2023-01-02             4.4366  24.79  75.41
2  2023-01-03             5.0050  24.58  74.73
3  2023-01-04             4.6798  24.47  76.75
4  2023-01-05             2.8044  24.33  77.81
```

Next steps: Generate code with df New interactive sheet

```
plt.figure(figsize=(10,5))
plt.plot(df['Date'], df['ALLSKY_SFC_SW_DWN'], label='Solar Irradiar
plt.title("Daily Solar Irradiance Over Time")
plt.xlabel("Date")
plt.ylabel("Irradiance (kWh/m²/day)")
plt.legend()
plt.xticks(rotation=45)
plt.show()
```



```
# Rename columns for clarity
df = df.rename(columns={
    'ALLSKY_SFC_SW_DWN': 'Solar_Irradiance',
    'T2M': 'Temperature',
    'RH2M': 'Humidity'
})

# Convert Date to datetime for plotting
df['Date'] = pd.to_datetime(df['Date'])

# Select features and target
X = df[['Temperature', 'Humidity']]
y = df['Solar_Irradiance']

# Split into training and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
```

```
model = RandomForestRegressor(n_estimators=200, random_state=42)
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)
```

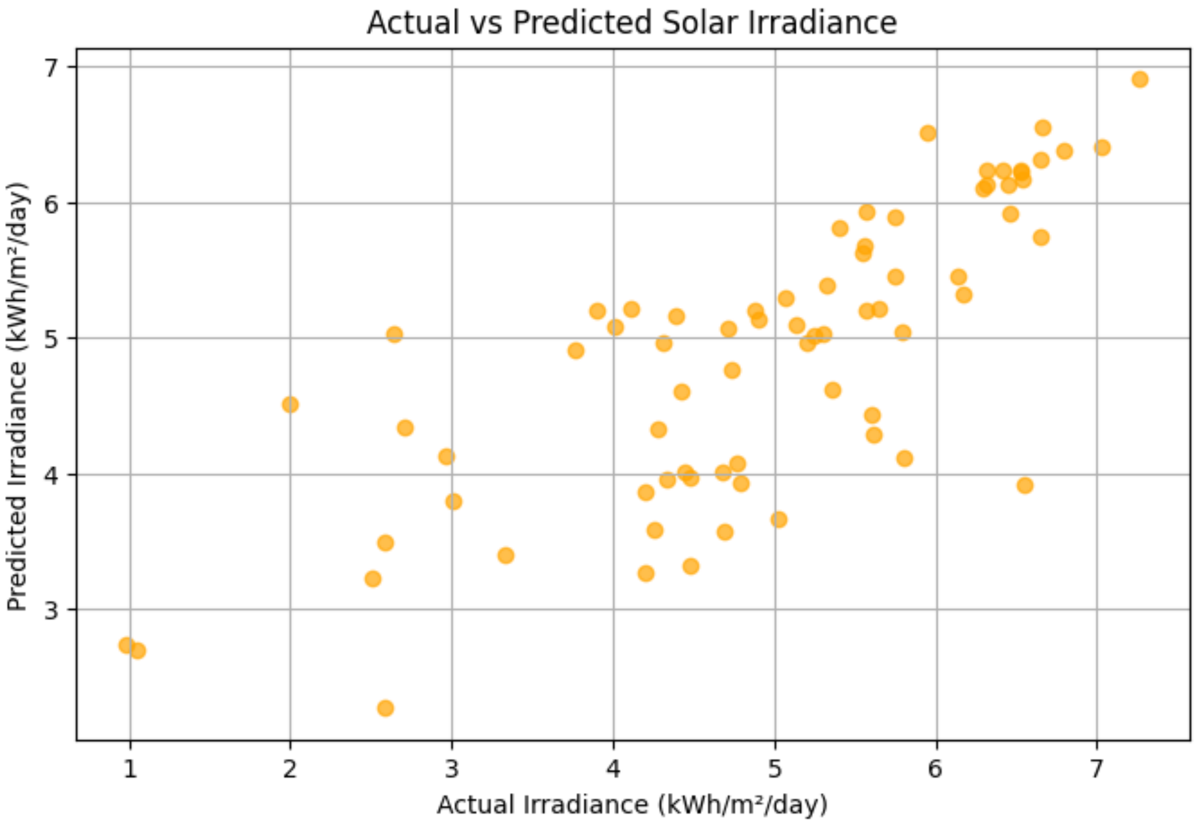
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Model Performance:")
print(f"Mean Squared Error: {mse:.3f}")
print(f"R² Score: {r2:.3f}")
```

```
Model Performance:
Mean Squared Error: 0.790
R² Score: 0.589
```

```
import joblib
joblib.dump(model, "solar_model.pkl")
```

```
['solar_model.pkl']
```

```
plt.figure(figsize=(8,5))
plt.scatter(y_test, y_pred, alpha=0.7, color='orange')
plt.title("Actual vs Predicted Solar Irradiance")
plt.xlabel("Actual Irradiance (kWh/m²/day)")
plt.ylabel("Predicted Irradiance (kWh/m²/day)")
plt.grid(True)
plt.show()
```



```
import numpy as np

# Sample coordinates (latitude, longitude)
locations = [
    (17.3850, 78.4867, "Hyderabad"),
    (16.3067, 80.4365, "Guntur"),
    (15.9129, 79.7400, "Ongole"),
    (14.4426, 79.9865, "Nellore"),
    (13.0827, 80.2707, "Chennai")
]

# Simulated temperature and humidity data (you can replace with rea
temp = [32, 33, 31, 30, 34]
humidity = [55, 50, 60, 65, 58]

test_df = pd.DataFrame({
    'Latitude': [loc[0] for loc in locations],
    'Longitude': [loc[1] for loc in locations],
    'Place': [loc[2] for loc in locations],
    'Temperature': temp,
    'Humidity': humidity
})

test_df
```

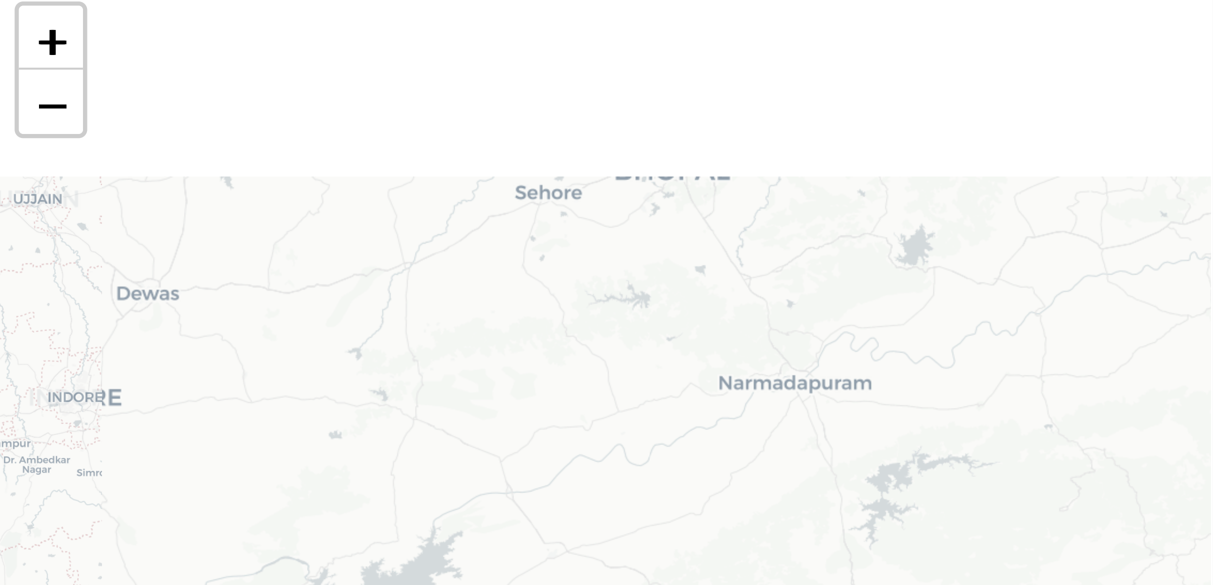
	Latitude	Longitude	Place	Temperature	Humidity
0	17.3850	78.4867	Hyderabad	32	55
1	16.3067	80.4365	Guntur	33	50
2	15.9129	79.7400	Ongole	31	60
3	14.4426	79.9865	Nellore	30	65
4	13.0827	80.2707	Chennai	34	58

Next steps: Generate code with `test_df` New interactive sheet

```
# Predict using your trained model
test_df['Predicted_Solar_Potential'] = model.predict(test_df[['Temp

# Display results
print(test_df[['Place', 'Predicted_Solar_Potential']])
```

```
       Place  Predicted_Solar_Potential
0  Hyderabad                   6.238898
1     Guntur                   6.253023
2     Ongole                   5.567684
3    Nellore                   5.497132
4    Chennai                   6.200803
```

```
# Create a base map
solar_map = folium.Map(location=[16.5, 79.5], zoom_start=6, tiles='

# Add markers
for _, row in test_df.iterrows():
    popup_text = f"{row['Place']}<br>Solar Potential: {row['Predict
    folium.CircleMarker(
        location=[row['Latitude'], row['Longitude']],
        radius=8,
        color='orange',
        fill=True,
        fill_color='orange',
        fill_opacity=0.7,
        popup=popup_text
    ).add_to(solar_map)

solar_map
```



ADVANCED SOLAR POTENTIAL MAPPING

20–50 coordinates covering Andhra Pradesh

```
import requests, pandas as pd, time

coords = [
    (17.3850,78.4867), (16.3067,80.4365), (15.9129,79.7400),
    (14.4426,79.9865), (13.0827,80.2707), (18.1124,83.3956),
    (15.8281,78.0373), (19.0760,72.8777), (12.9716,77.5946)
]

records = []

for lat, lon in coords:
    url = f"https://power.larc.nasa.gov/api/temporal/daily/point?pa
    r = requests.get(url)
    data = r.json()['properties']['parameter']
    # Correctly parse the data into a DataFrame
    df = pd.DataFrame(data)
    df.index = pd.to_datetime(df.index, format='%Y%m%d')
    df = df.reset_index().rename(columns={'index': 'Date'})
    df['Latitude'], df['Longitude'] = lat, lon
    records.append(df)
    time.sleep(1)   # be polite to the API

big_df = pd.concat(records, ignore_index=True)
big_df.head()
```

	Date	ALLSKY_SFC_SW_DWN	T2M	RH2M	Latitude	Longitude
0	2023-01-01	3.8635	22.65	69.13	17.385	78.4867
1	2023-01-02	3.8614	22.19	74.37	17.385	78.4867
2	2023-01-03	3.8964	21.85	74.11	17.385	78.4867
	2023-01-					

Next steps: Generate code with big_df New interactive sheet

Clearness Index (Kt) and Monthly Averages.

```
import numpy as np

# Extraterrestrial Radiation (simplified constant for now)
# ~1367 W/m² × 24 h × 3600 s ÷ 1e6 = 49.2 MJ/m²/day ≈ 13.7 kWh/m²/c
E0 = 13.7

big_df['Solar_Irradiance'] = big_df['ALLSKY_SFC_SW_DWN']
big_df['Temperature'] = big_df['T2M']
big_df['Humidity'] = big_df['RH2M']
big_df['Clearness_Index'] = big_df['Solar_Irradiance'] / E0

# Monthly aggregation
big_df['Date'] = pd.to_datetime(big_df['Date'])
monthly = (big_df.groupby([pd.Grouper(key='Date', freq='ME'),
                           'Latitude','Longitude'])
           .agg({'Solar_Irradiance':'mean',
                 'Temperature':'mean',
                 'Humidity':'mean',
                 'Clearness_Index':'mean'})
           .reset_index())
monthly.head()
```

	Date	Latitude	Longitude	Solar_Irradiance	Temperature	Humidity
0	2023-01-31	12.9716	77.5946	5.534855	18.820000	74.017097
1	2023-01-31	13.0827	80.2707	4.915871	23.923548	76.627419
2	2023-01-31	14.4426	79.9865	5.003552	23.873871	77.255806

Next steps: Generate code with monthly New interactive sheet

Train ML Model on Aggregated Data

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

X = monthly[['Temperature','Humidity','Clearness_Index']]
y = monthly['Solar_Irradiance']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size

model = RandomForestRegressor(n_estimators=300, random_state=42)
model.fit(X_train, y_train)

preds = model.predict(X_test)
print("R²:", r2_score(y_test, preds))
```

```
R²: 0.9911067871473368
```

Regional Predictions

```
import numpy as np

lat_range = np.arange(12,20,0.5)
lon_range = np.arange(76,84,0.5)

grid = [(lat,lon) for lat in lat_range for lon in lon_range]
grid_df = pd.DataFrame(grid, columns=['Latitude','Longitude'])

# Assume average temp/humidity (or use regional interpolation)
grid_df['Temperature'] = 32
grid_df['Humidity'] = 55
grid_df['Clearness_Index'] = 0.75

grid_df['Predicted_Irradiance'] = model.predict(grid_df[['Temperatu
```

SOLAR HEATMAP

```
import folium
from folium.plugins import HeatMap

m = folium.Map(location=[16,79], zoom_start=6, tiles='CartoDB posit

heat_data = [[row['Latitude'], row['Longitude'], row['Predicted_Irr
             for _, row in grid_df.iterrows()]
```

```
HeatMap(heat_data, radius=12, blur=15, max_zoom=6).add_to(m)
m
```

```
# Step: Save a compatible version of the model
import joblib

# Assuming cell 10 already loaded your model as 'model'
joblib.dump(model, "solar_model_compatible.pkl")
```

```
['solar_model_compatible.pkl']
```

```
grid_df.to_csv("grid_data.csv", index=False)
```