**Code and Tools availability statement**

Open-source tools were only used for the manuscript. The data extraction was carried out using Google Earth Engine (<https://code.earthengine.google.com/>). The SVR modelling was carried out using Anaconda which can be downloaded from- <https://www.anaconda.com/download>, using the **Jupyter** module. The Maps were prepared in QGIS which is an open-source software and can be downloaded for free from- <https://qgis.org/download/>.

**I. GOOGLE EARTH ENGINE SCRIPTS**

**1. CHIRPS Precipitation Data Extraction**

var chirps = ee.ImageCollection("UCSB-CHG/CHIRPS/DAILY")

.filterDate('2021-05-01', '2023-05-31')

.select('precipitation')

.mean()

.clip(regionOfInterest);

Export.image.toDrive({

image: chirps,

description: 'CHIRPS\_Mean\_2021\_2023',

scale: 5000,

region: regionOfInterest

});

**2. MODIS ET (MOD16A2) Data Extraction**

var modisET = ee.ImageCollection("MODIS/006/MOD16A2")

.filterDate('2021-05-01', '2023-05-31')

.select('ET')

.mean()

.clip(regionOfInterest);

Export.image.toDrive({

image: modisET,

description: 'MODIS\_ET\_Mean\_2021\_2023',

scale: 500,

region: regionOfInterest

});

**3. SMAP Soil Moisture Data Extraction**

var smap = ee.ImageCollection("NASA\_USDA/HSL/SMAP10KM\_soil\_moisture")

.filterDate('2021-05-01', '2023-05-31')

.select('ssm')

.mean()

.clip(regionOfInterest);

Export.image.toDrive({

image: smap,

description: 'SMAP\_Mean\_2021\_2023',

scale: 10000,

region: regionOfInterest

});

**II. PYTHON CODE (FOR SVR MODELING & EVALUATION)**

**1. Data Preprocessing and Resampling**

import rasterio

from rasterio.enums import Resampling

import numpy as np

def resample\_raster(src\_path, dst\_path, scale\_factor):

with rasterio.open(src\_path) as src:

data = src.read(

out\_shape=(

src.count,

int(src.height \* scale\_factor),

int(src.width \* scale\_factor)

),

resampling=Resampling.nearest

)

profile = src.profile

profile.update({

"height": data.shape[1],

"width": data.shape[2],

"transform": src.transform \* src.transform.scale(

(src.width / data.shape[-1]),

(src.height / data.shape[-2])

)

})

with rasterio.open(dst\_path, "w", \*\*profile) as dst:

dst.write(data)

**2. Support Vector Regression (SVR) with Scikit-learn**

from sklearn.svm import SVR

from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error

from sklearn.model\_selection import train\_test\_split

import numpy as np

# X: CHIRPS - MODIS ET difference, y: SMAP soil moisture

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

svr\_model = SVR(kernel='linear', C=1.8, epsilon=0.005)

svr\_model.fit(X\_train, y\_train)

y\_pred = svr\_model.predict(X\_test)

print("R2 Score:", r2\_score(y\_test, y\_pred))

print("MAE:", mean\_absolute\_error(y\_test, y\_pred))

print("RMSE:", mean\_squared\_error(y\_test, y\_pred, squared=False))

**Notes:**

* The manuscript used **linear, polynomial, RBF, and sigmoid kernels** — these can be toggled using the kernel argument in the SVR instantiation.
* Downscaling and reprojection to a 500 m resolution was done using **nearest neighbor resampling**.
* The workflow involved **GEE for data preparation** and **Python (scikit-learn, rasterio)** for modeling and evaluation.