SOIL EROSION ESTIMATION BY USING RUSLE MODEL

Dhirubhai Ambani Institute of Information and Communication Technology

SUBMITTED BY:

Mit Borda

(202319008)

Kaushal Kathiriya

(202319013)

Nidhi Gusai

(202319015)

Drashti Nayakpara

(202319019)

Divya prajapati

(202319024)

GUIDED BY

Dr. Ranendu Ghosh

INTRODUCTION

- Soil erosion is the natural process in which the topsoil of a field is carried away by physical sources such as wind and water.
- Soil erosion is major problem in many parts of the world including India.
- Soil erosion is natural process, but human activities have accelerated the rate of soil erosion.
- Various factors like deforestation, over grazing, industrialization, mining, climatic changes etc mainly affects soil erosion.
- Climate change is major cause of soil erosion and degradation. Heavy rainfall events can cause soil to become compacted and reducing its ability to absorb water.
- Floods can carry away large amounts of soil and vegetation.
- Soil erosion can have a significant impact on the ecosystem and can lead to habitat destruction, loss of biodiversity, and soil degradation. Soil erosion can reduce agricultural productivity by reducing soil fertility and decreasing water availability.
- Erosion and deposition ultimately results in soil loss, decreased soil depth, deterioration of the soil's structure, declining levels of organic matter and minerals, and decreases its fertility.

Reference:

https://www.sciencedirect.com/science/article/pii/S004896972101562X

https://www.researchgate.net/

DESCRIPTION OF STUDY AREA

We take the study area of Kheda district, which spans an area of 719400 ha (3953 sq.km) and it is located at coordinates 72.68’ E longitude and 22.75’ N latitude. The study area is at altitude of 20 to 25m from mean sea level.

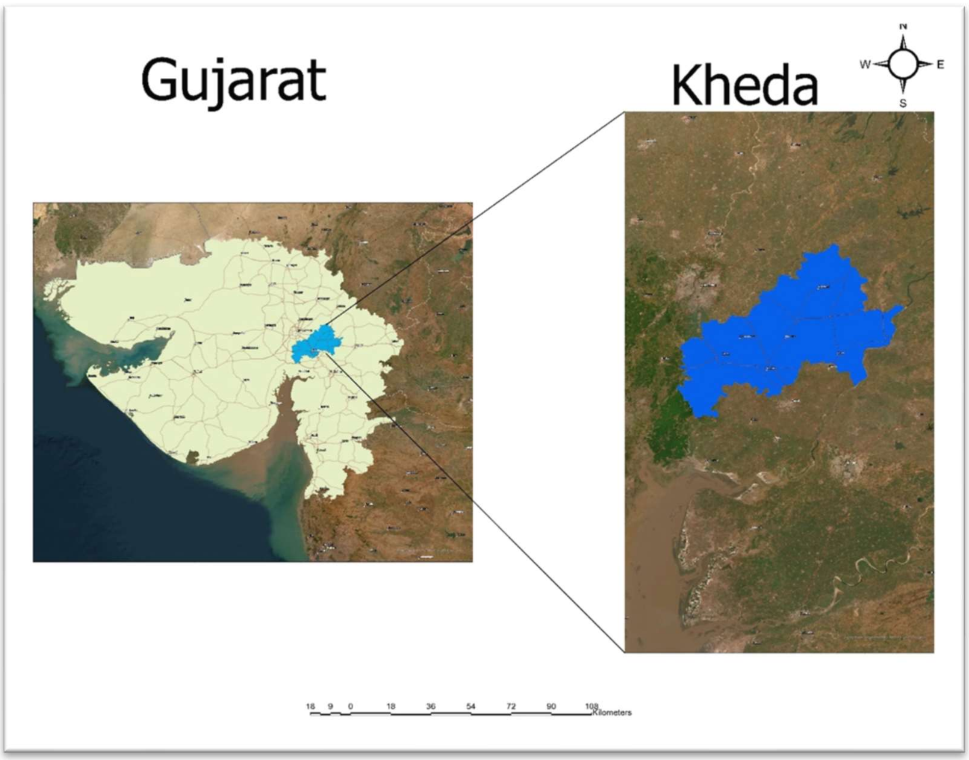


Image: Study Area

RUSLE model

The Revised Universal Soil Loss Equation used to estimate average annual soil erosion potential,

$A=R\times K\times L\times S\times C\times P$...(1)

Where,

A = computed average annual soil loss (tons/ha/year), R = rainfall-runoff erosivity factor, K soil erodibility factor, L = slope length factor, S= slope steepness factor, C = cover-management factor, P= conservation practice factor

PARAMETERS

Rainfall Factor

The R Factor in the Revised Universal Soil Loss Equation (RUSLE) model represents the Rainfall Erosivity Factor. RUSLE is a widely used empirical model for estimating soil erosion caused by water. The R Factor specifically quantifies the erosive force of rainfall on the soil. The formula for calculating the R Factor is:

$$\mathbf{R = \Sigma^{n}_{i=1}(EI_{30})_i}$$

where:

- (R) is the Rainfall Erosivity Factor,

- (n) is the number of storms in a year,

-$(EI_{30})_i$ is the Energy Index for each storm, representing the product of the rainfall kinetic energy (E) and the maximum 30-minute rainfall intensity (I_{30})for the (i)th storm.

Here's a breakdown of the components:

1. Rainfall Kinetic Energy (E):

- (E) is a measure of the energy available for detachment and transport of soil particles by raindrop impact.

- It is calculated using the formula (E = 0.039 \times I_{30}^{1.67}), where (I_{30}) is the maximum 30-minute rainfall intensity in millimeters per hour.

2. Maximum 30-Minute Rainfall Intensity (I_{30}):

- (I_{30}) represents the maximum intensity of rainfall during a 30-minute period in a storm event.

- It is usually derived from rainfall data and represents the erosive potential of the rainfall.

3. Energy Index $(EI_{30})_i$:

- $(EI_{30})_i$ for each storm is the product of (E) and (I_{30}) for that specific storm event.

4. Summation:

- The R Factor is obtained by summing up the $(EI_{30})_i$ values for all storm events in a year.

The R Factor provides a measure of the erosive power of rainfall for a specific location and time period. Higher R Factor values indicate greater potential for soil erosion due to more intense and erosive rainfall events. It is a critical parameter in soil erosion modeling, helping to assess and manage the risk of soil erosion in different regions.

Soil Erodibility Factor K

In RULSE, K is assumed to be constant throughout the year. Tables of K values are available in Soil Conservation Service Offices for most soils in the U. S. In the absence of published data, a widely used relationship for predicting erodibility is a nomograph by Wischmeier et al. (1971). Soil erodibility in the nomograph is predicted as a function of soil and soil profile parameters:

Percent silt (MS; 0.002-0.05 mm)

Percent very fine sand (VFS; 0.05-0.1 mm)

Percent sand (SA; 0.1-2 mm)

Percent organic matter (OM)

Structure code (s)

Permeability code (p)

The analytical relationship for the nomograph by Wischmeier et al. (1971) is given by following regression equation,

K= 2.1×10-4(12-OM)$M^{1.14}$ +3.25(s-2)+2.5(p-3) /759.4 ...(4)

Where,

K = soil erodibility (tons-yr/MJ-mm), OM = percentage organic matter, p = soil permeability code, s = soil structure code, M = a function of the primary particle size fraction given by:

M = (%silt + %very fine sand) × (100-%clay) ...(5)

Soil structure :

1. very fine granular
2. fine granular
3. medium or coarse granular
4. blocky, platy, prism like

Soil permeability :

1. rapid:(>150mm/hr)
2. moderate to rapid:(50-150 mm/hr)
3. moderate : (15-50 mm/hr)
4. slow to moderate:(5-15 mm/hr)
5. slow: (1-5 mm/hr)
6. very slow :(< 1mm/hr)

Conservation Practice Factor P

Support practice factor (P): P factor indicates erosion conservation practices and soil conservation measures on the annual soil loss from the watershed. It is the ratio of soil loss with contouring and/or strip cropping to that with straight row farming up-and-down slope. Its value generally lies between 0 to 1.

Cover-Management Factor C

The cover-management factor is the ratio of soil loss from an area with specified cover and management to that of an identical area in tilled continuous fallow. De Jong (1994) in his PhD thesis described the use of vegetation indices in order to extract vegetation parameters for erosion models. Based on his work following statement can be assumed to be valid in general: i) NDVI and RULSE C-factor are correlating; ii) There is a linear relation between NDVI and RULSE C-factor

Based on these assumptions the NDVI map can be analyzed to formulate the linear equation between NDVI and C factor. The NDVI values less than Zero (0) indicate the water and snow, so the negative values should not be considered in preparing the C factor equation. With this boundary conditions the regression equation for C factor can be developed.

$C_i = 0$ if NDVI ≤ 0

$C_i = -(l/(NDVI))\ (NDVI)+l$ if $0 <$ NDVI ...(6)

$$\mathbf{C=exp}[-\boldsymbol{\alpha}(\boldsymbol{NDVI}/\boldsymbol{\beta} - \boldsymbol{NDVI})]$$

Slope Length and Slope Steepness Factor LS

The effect of topography on erosion in RUSLE is accounted for by the LS factor. The slope length factor (L) is calculated using following equation,

$L= (\lambda/22.13)$ ...(7)

Where,

22.13 = the RUSLE unit plot length (m) and m = a variable slope-length exponent. Slope length is defined as the horizontal distance from the origin of overland flow to the point where either (1) the slope gradient decreases enough that deposition begins or (2) runoff becomes concentrated in a defined channel.

The slope-length exponent m is calculated as,

$m= \beta/(1+\beta)$ ...(8)

$\beta = (\sin\theta/0.0896)/[3.0(\sin\theta)^{0.8} +0.56]$ ...(9)

Where,

θ = slope angle.

The slope steepness factor (S) is evaluated from (McCool et al., 1987, 1993)

$S=10.8 \sin\theta +0.03 \quad S <9\%$ (i.e.tan $\theta <0.09$)

$S= (\sin\theta /\sin 5.143)^{0.6} \quad S\geq 9\%$ (i.e. tan $\theta \geq 0.09$) ...(10)

To calculate the Slope Length and Slope Steepness factor the slope map (degrees) and flow accumulation map can be derived from DEM using Hydrology tools available in Spatial Analyst tool box of ESRI Arc GIS.

Value 1 is assigned to all the grids having flow accumulation values equal to zero (since we are calculating the soil erosion at gird level). For further calculations the slope is converted in to radians (since all the trigonometric function in ESRI Arc-GIS are in radians).

The slope length factor map is derived by applying Eq. 7 on flow accumulation map and slope length exponent (m) map in raster calculator environment of Arc-GIS 9.1. Eq. 7 is converted in to form of grid equation as given below

L= (Flow Accumulation × Grid Size/22.13)^m ...(11)

Where,

Grid size = 1000 m, and slope length exponent m is taken from m map for respective grid.

The slope steepness factor (S) map is derived by applying Eq. 10 on slope map of India in Raster calculator of Arc-GIS. The Eq. 10 is converted in to grid equation as;

CON (Tan([slope deg_c]3.1428/180)<0.09,

10.8 Sin([slope deg_c]*3.1428/180)+0.03,

Pow((Sin([slope deg_cj*3.1428/180) / Sin (5.143*3.1428/180)),0.6)) ...(12)

METHODOLOGY

i. Data download from google earth engine .
ii. Calculate factors by using python.
iii. Creating map of all the factors of RUSLE model.
iv. Calculate soil erosion with RUSLE model by using python.
v. Calculate taluka wise mean soil erosion of Kheda district by using ArcGIS.

Satellite imageries used in the study:

No.	Sensor	Year of Acquisition	Spatial Resolution
1	**Sentinel -2**	2020	10m
2	**Landsat - 8**	2020	30m
3	**CHIRPS**	2020	5566m

Datasets used:

Chirps rainfall data from: https://developers.google.com/earth-engine/datasets/catalog/UCSB-CHG_CHIRPS_PENTAD

Soil erodibility data from: https://developers.google.com/earth-engine/datasets/catalog/OpenLandMap_SOL_SOL_TEXTURE-CLASS_USDA-TT_M_v02

LS factor data: https://developers.google.com/earth-engine/datasets/catalog/USGS_SRTMGL1_003

Land use land cover data: https://developers.google.com/earth-engine/datasets/catalog/MODIS_061_MCD12Q1

NDVI data: https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_S2

Method of data download:

We use Google earth engine to download datasets

Chirps rainfall data download method:

Input:

```
var dataset = ee.ImageCollection('UCSB-CHG/CHIRPS/DAILY')
                  .filter(ee.Filter.date('2018-05-01', '2018-05-03'));
var precipitation = dataset.select('precipitation');
var precipitationVis = {
  min: 1,
  max: 17,
  palette: ['001137', '0aab1e', 'e7eb05', 'ff4a2d', 'e90000'],
};
Map.setCenter(17.93, 7.71, 2);
Map.addLayer(precipitation, precipitationVis, 'Precipitation');
```

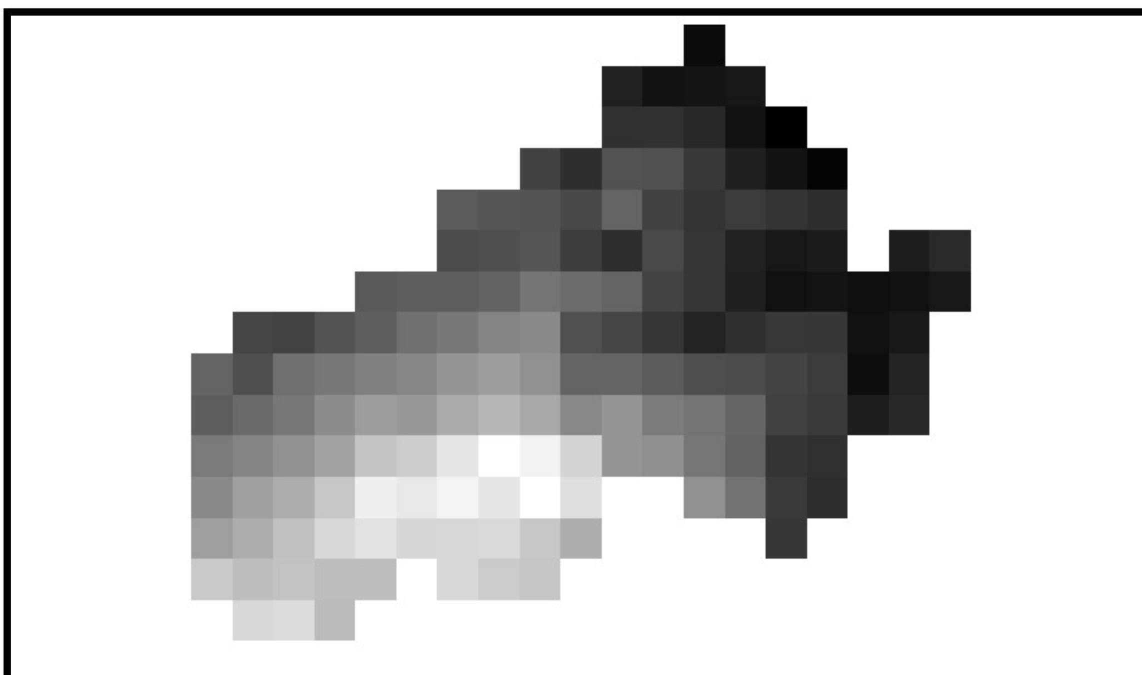
Output:

We get data for whole globe to extract data of over interest steps followed are:

Load raster image in arc gis

Use extract by mask tool

Input raster image and kheda shape file than run file and get output image.



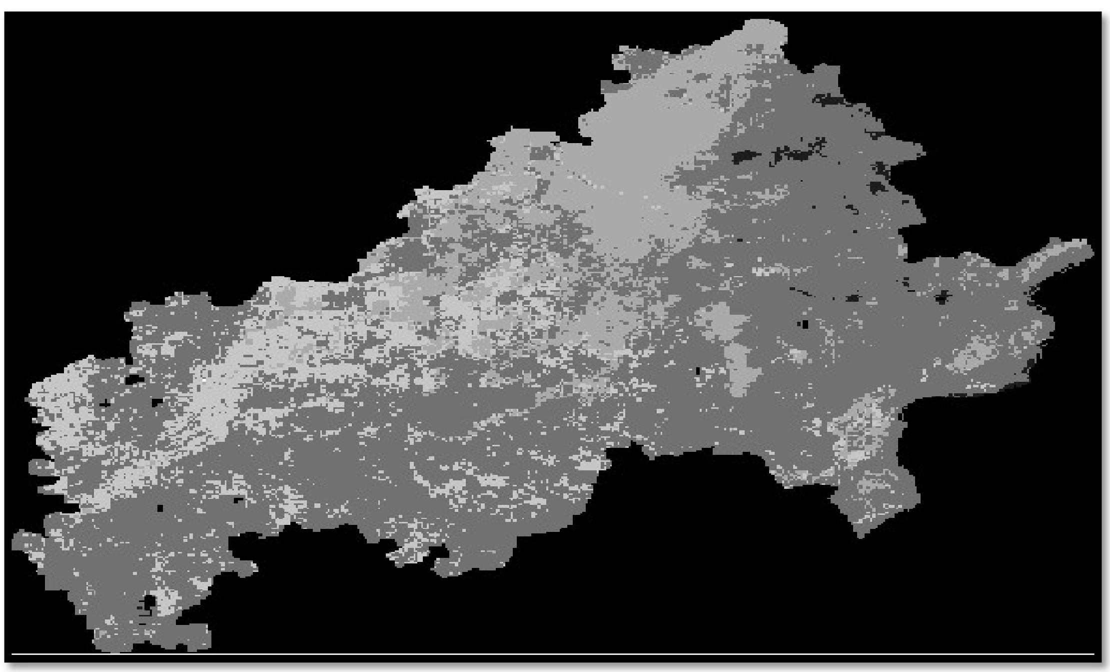
Chirps rainfall data

Soil erodibility data download method:

Input:

```
soil = soil.select('b0').clip(aoi).rename('soil')
Map.addLayer(soil, {min: 0, max: 100, palette:
['a52508','ff3818','fbff18','25cdff','2f35ff','0b2dab']}, 'Soil', 0);

var K = soil.expression(
    "(b('soil') > 11) ? 0.0053" +
      ": (b('soil') > 10) ? 0.0170" +
        ": (b('soil') > 9) ? 0.045" +
           ": (b('soil') > 8) ? 0.050" +
            ": (b('soil') > 7) ? 0.0499" +
            ": (b('soil') > 6) ? 0.0394" +
            ": (b('soil') > 5) ? 0.0264" +
            ": (b('soil') > 4) ? 0.0423" +
            ": (b('soil') > 3) ? 0.0394" +
            ": (b('soil') > 2) ? 0.036" +
            ": (b('soil') > 1) ? 0.0341" +
            ": (b('soil') > 0) ? 0.0288" +
             ": 0")
             .rename('K').clip(aoi);

Map.addLayer(K, {min: 0, max: 0.06, palette:
['a52508','ff3818','fbff18','25cdff','2f35ff','0b2dab']}, 'KFactor
Map', 0);
```

Output:



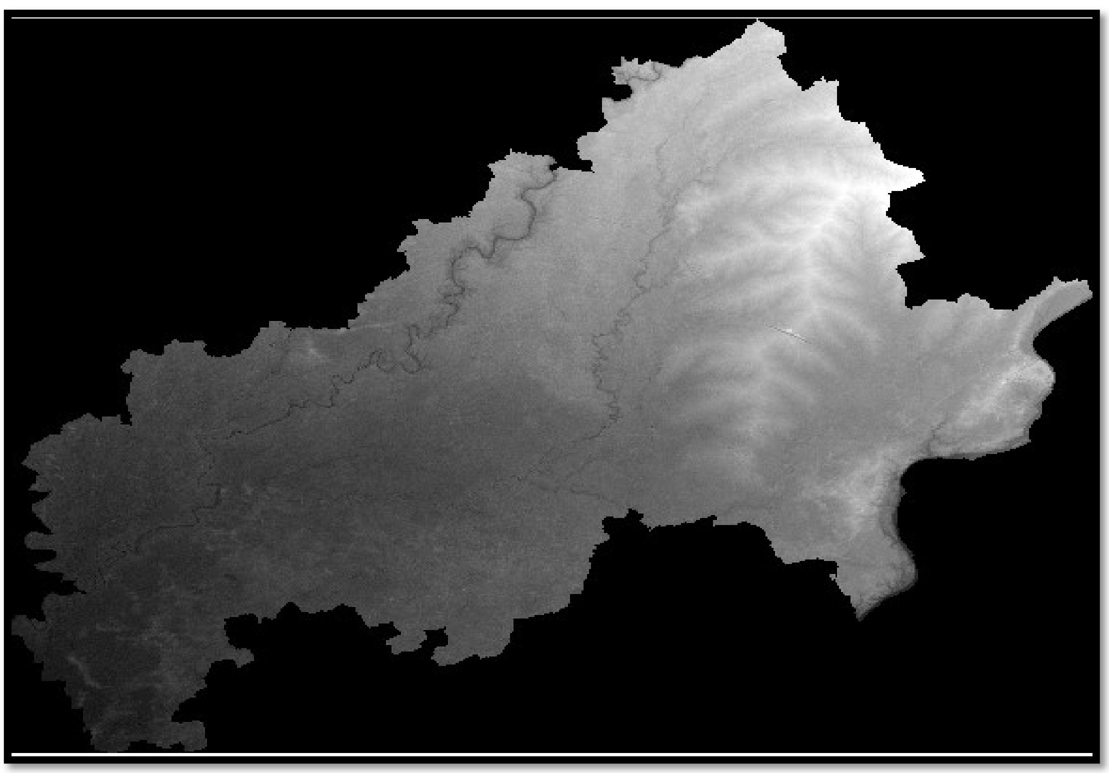
Soil erodibility data

LS factor data download method:

Input:

```
var elevation = DEM.select('elevation');
var slope1 = ee.Terrain.slope(elevation).clip(aoi);
    //Converting Slope from Degrees to %
var slope = slope1.divide(180).multiply(Math.PI).tan().multiply(100);
Map.addLayer(slope, {min: 0, max: 15, palette:
['a52508','ff3818','fbff18','25cdff','2f35ff','0b2dab']}, 'slope in %',
0);

var LS4 = Math.sqrt(500/100);
var LS3 = ee.Image(slope.multiply(0.53));
var LS2 = ee.Image(slope).multiply(ee.Image(slope).multiply(0.076));
var LS1 = ee.Image(LS3).add(LS2).add(0.76);
var LS = ee.Image(LS1).multiply(LS4).rename("LS");

Map.addLayer(LS, {min: 0, max: 90, palette:
['a52508','ff3818','fbff18','25cdff','2f35ff','0b2dab']}, 'LS Factor
Map', 0);
```

Output:

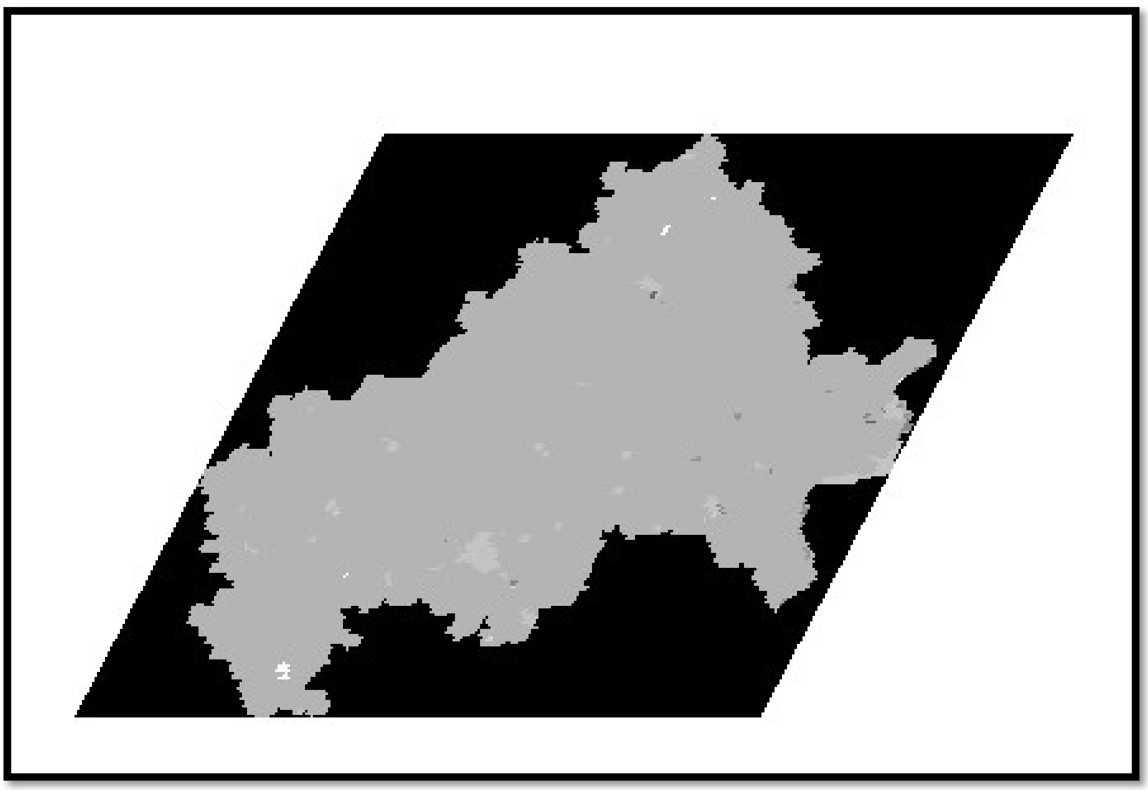


LS data

LULC data download method:

```
var lulc = modis.filterDate(date1, date2).select('LC_Type1')
        .first().clip(aoi).rename('lulc');
Map.addLayer (lulc, {}, 'lulc', 0)

// Combined LULC & slope in single image
var lulc_slope = lulc.addBands(slope)

// Create P Facor map using an expression
var P = lulc_slope.expression(

     "(b('lulc') < 11) ? 0.8" +
      ": (b('lulc') == 11) ? 1" +
      ": (b('lulc') == 13) ? 1" +
      ": (b('lulc') > 14) ? 1" +
      ": (b('slope') < 2) and((b('lulc')==12) or (b('lulc')==14)) ? 0.6" +
    ": (b('slope') < 5) and((b('lulc')==12) or (b('lulc')==14)) ? 0.5" +
    ": (b('slope') < 8) and((b('lulc')==12) or (b('lulc')==14)) ? 0.5" +
    ": (b('slope') < 12) and((b('lulc')==12) or (b('lulc')==14)) ? 0.6" +
    ": (b('slope') < 16) and((b('lulc')==12) or (b('lulc')==14)) ? 0.7" +
    ": (b('slope') < 20) and((b('lulc')==12) or (b('lulc')==14)) ? 0.8" +
    ": (b('slope') > 20) and((b('lulc')==12) or (b('lulc')==14)) ? 0.9" +
    ": 1"
).rename('P').clip(aoi);
Map.addLayer (P, {}, 'P Factor', 0)
```

Output:

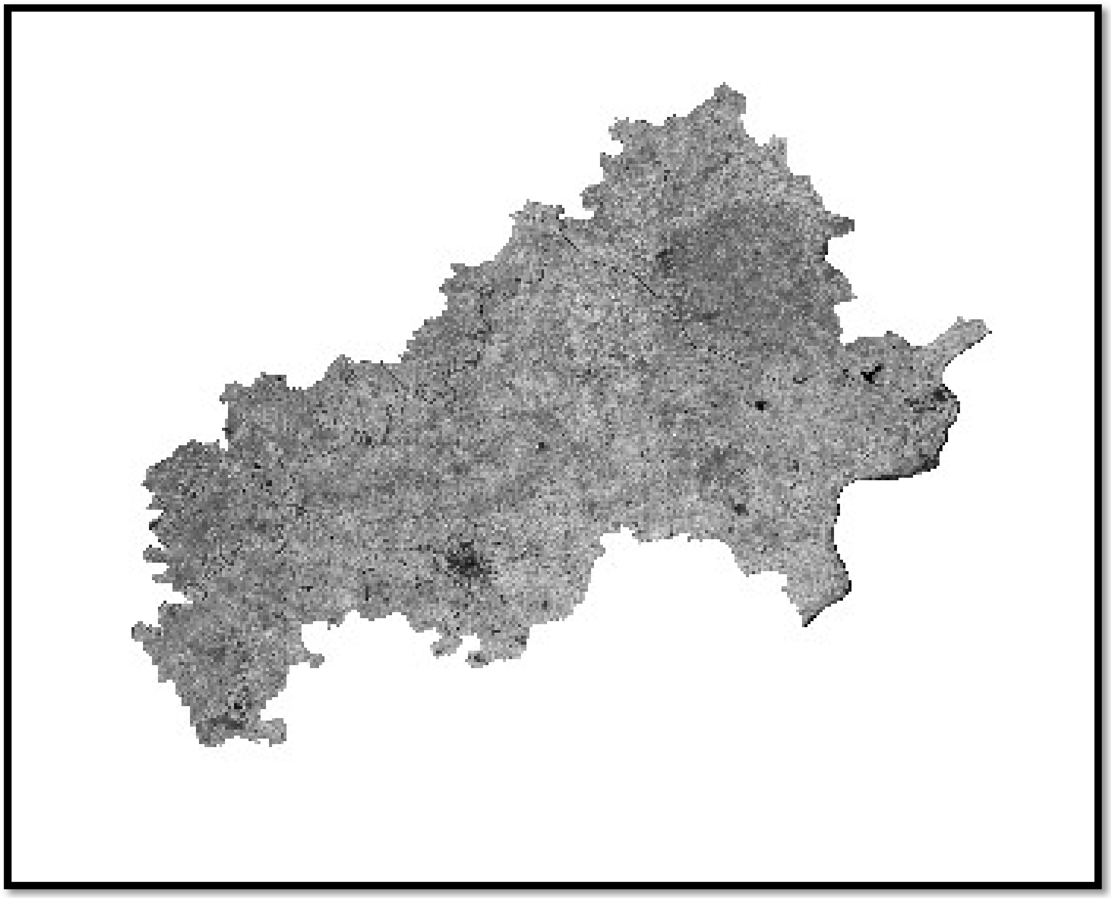


LULC data

NDVI data download method:

```
s2 = s2.filterDate(date1, date2).median().clip(aoi);
var image_ndvi =
s2.normalizedDifference(['B8','B4']).rename("NDVI");

Map.addLayer (image_ndvi, {min: 0, max: 0.85, palette:
['FFFFFF','CC9966','CC9900', '996600', '33CC00',
'009900','006600','000000']}, 'NDVI', 0);

var alpha = ee.Number(-2)
var beta = ee.Number (1)

var C1 = image_ndvi.multiply(alpha)
var oneImage = ee.Image(1).clip(aoi);
var C2 = oneImage.subtract(image_ndvi)
var C3 = C1.divide(C2).rename('C3')
var C4 = C3.exp()

var maxC4 = C4.reduceRegion({
  geometry: aoi,
  reducer: ee.Reducer.max(),
  scale: 3000,
  maxPixels: 475160679
})

var C5 = maxC4.toImage().clip(aoi)
var minC4 = C4.reduceRegion({

  geometry: aoi,
  reducer: ee.Reducer.min(),
  scale: 3000,
  maxPixels: 475160679
})

var C6 = minC4.toImage().clip(aoi)
var C7 = C4.subtract(C6)
var C8 = C5.subtract(C6)

var C = C7.divide(C8).rename('C')

Map.addLayer (C, {min: 0, max: 1, palette:
['FFFFFF','CC9966','CC9900', '996600', '33CC00',
'009900','006600','000000']}, 'C Map',0);
```

Output:



NDVI data

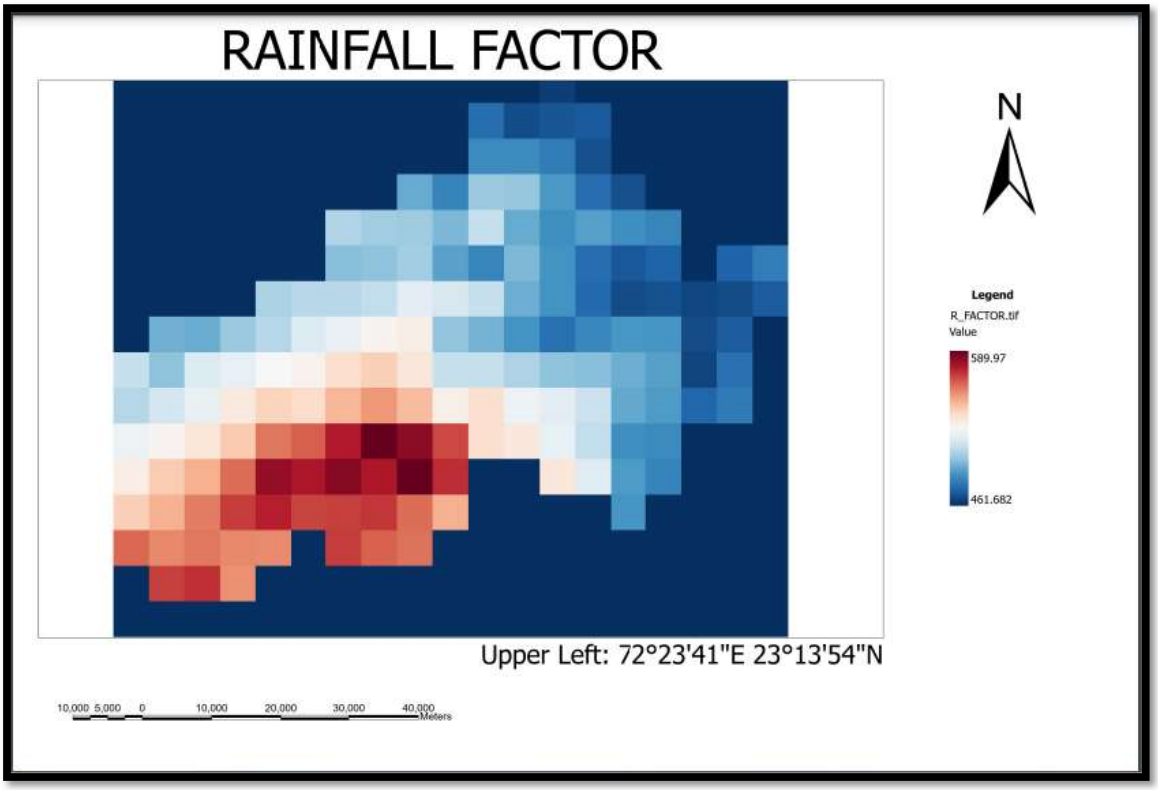
Step 1: Calculate and create R factor raster using python then created the map of output image using ArcGIS.

Input:

```
from osgeo import gdal

import numpy as np

#1. R FActor FROM RAINFALL

# Define the function to calculate the R factor from rainfall

# This is a placeholder function, and you will need to replace it with the actual formula

def calculate_r_factor(rainfall_array):

    # Apply the R factor formula here

    # For example, R = (E * I30) / A, where E is the total kinetic energy of the storm,

    # I30 is the maximum 30-minute intensity, and A is the area (if needed).

    # This is just a conceptual representation and will differ based on your specific formula.

    r_factor_array = (rainfall_array * 0.363)+79

    return r_factor_array

    print(r_factor_array.shape())


# Function to read a raster and convert it to an array

def raster_to_array(raster_path):

    raster = gdal.Open(raster_path)

    band = raster.GetRasterBand(1)

    array = band.ReadAsArray()

    return array
```

```
# Function to write an array to a raster, using a reference raster for
#georeferencing
def array_to_raster(array, reference_raster_path, output_raster_path):
    reference_raster = gdal.Open(reference_raster_path)
    driver = gdal.GetDriverByName('GTiff')
    out_raster = driver.Create(output_raster_path, reference_raster.RasterXSize,
reference_raster.RasterYSize, 1, gdal.GDT_Float32)
    out_raster.SetGeoTransform(reference_raster.GetGeoTransform())
    out_raster.SetProjection(reference_raster.GetProjection())
    out_band = out_raster.GetRasterBand(1)
    out_band.WriteArray(array)
    out_band.FlushCache()
    out_raster = None

# Load your CHIRPS rainfall raster image
rainfall_raster_path = r'/content/gdrive/MyDrive/Rusle/Rusle
project/All_factor/Input/Chirps_Rainfall_Data.tif'
rainfall_array = raster_to_array(rainfall_raster_path)

# Calculate the R factor from the rainfall data
r_factor_array = calculate_r_factor(rainfall_array)

# Save the R factor as a new raster image
output_raster_path = r'/content/gdrive/MyDrive/rusle1/R_factor1.tif'
array_to_raster(r_factor_array, rainfall_raster_path, output_raster_path)
print("R factor raster created successfully.")
```

Then by use of R_factor1.tif we have created the map of R_factor in ArcGIS



Step 2: Calculate and create K factor raster using python then created the map of output image using ArcGIS.

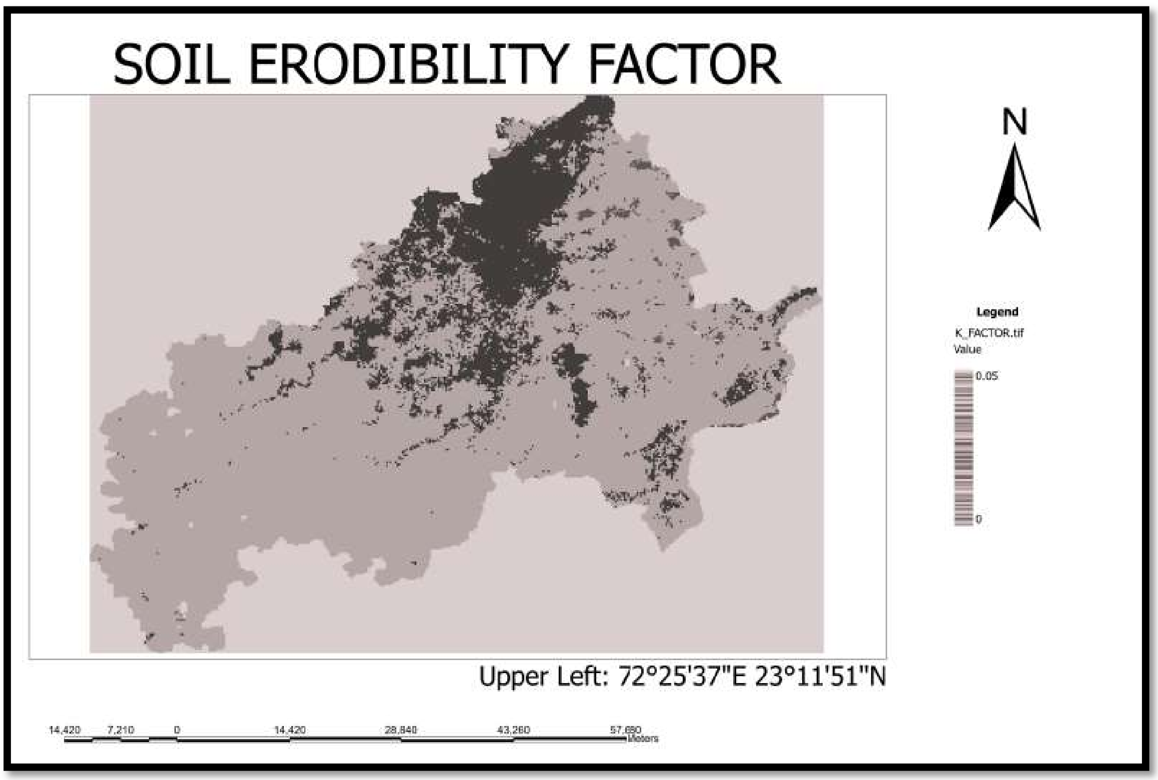
Input:

```
from osgeo import gdal

import numpy as np

# 2. K FACTOR FROM SOIL DATA PROVIDED BY USGS

# Function to read a raster and convert it to an array

def raster_to_array(raster_path):

    raster = gdal.Open(raster_path)

    band = raster.GetRasterBand(1)

    array = band.ReadAsArray()

    return array


# Function to write an array to a raster, using a reference raster for
georeferencing

def array_to_raster(array, reference_raster_path, output_raster_path):

    reference_raster = gdal.Open(reference_raster_path)

    driver = gdal.GetDriverByName('GTiff')

    out_raster = driver.Create(output_raster_path, reference_raster.RasterXSize,
reference_raster.RasterYSize, 1, gdal.GDT_Float32)

    out_raster.SetGeoTransform(reference_raster.GetGeoTransform())

    out_raster.SetProjection(reference_raster.GetProjection())

    out_band = out_raster.GetRasterBand(1)

    out_band.WriteArray(array)

    out_band.FlushCache()

    out_raster = None
```

```
# Calculate K factor based on the soil data and the provided conditions
def calculate_k_factor(soil_array):
    k_factor_array = np.zeros_like(soil_array, dtype=float)

    # Nested conditions are used to assign K factor values based on soil classes
    k_factor_array[soil_array > 11] = 0.0053
    k_factor_array[(soil_array > 10) & (soil_array <= 11)] = 0.0170
    k_factor_array[(soil_array > 9) & (soil_array <= 10)] = 0.045
    k_factor_array[(soil_array > 8) & (soil_array <= 9)] = 0.050
    k_factor_array[(soil_array > 7) & (soil_array <= 8)] = 0.0499
    k_factor_array[(soil_array > 6) & (soil_array <= 7)] = 0.0394
    k_factor_array[(soil_array > 5) & (soil_array <= 6)] = 0.0264
    k_factor_array[(soil_array > 4) & (soil_array <= 5)] = 0.0423
    k_factor_array[(soil_array > 3) & (soil_array <= 4)] = 0.0394
    k_factor_array[(soil_array > 2) & (soil_array <= 3)] = 0.036
    k_factor_array[(soil_array > 1) & (soil_array <= 2)] = 0.0341
    k_factor_array[(soil_array > 0) & (soil_array <= 1)] = 0.0288
    k_factor_array[soil_array <= 0] = 0  # This assumes that a value of 0 means
no soil data or not applicable

    return k_factor_array

# Load your soil raster image
soil_raster_path = r'/content/gdrive/MyDrive/Rusle/Rusle
project/All_factor/Input/SOIl_DATA_USDA.tif'
soil_array = raster_to_array(soil_raster_path)
```

```
# Calculate the K factor from the soil data
k_factor_array = calculate_k_factor(soil_array)

# Save the K factor as a new raster image
output_raster_path = r'/content/gdrive/MyDrive/rusle1/K_factor1.tif'
array_to_raster(k_factor_array, soil_raster_path, output_raster_path)
print("K factor raster created successfully.")
```

Then by use of K_factor1.tif we have created the map of K_factor in ArcGIS



Step 3: Calculate and create LS factor raster using python then created the map of output image using ArcGIS.

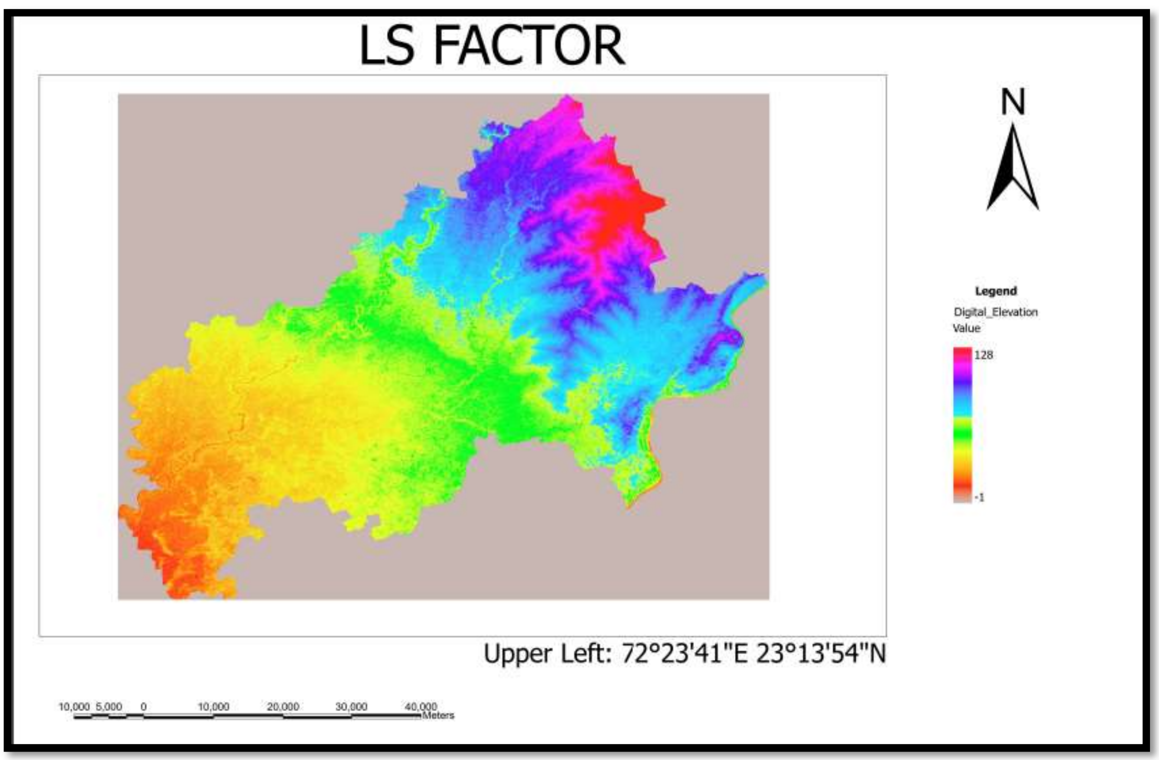
Input:

```
#3.LS FACTOR
import numpy as np
from osgeo import gdal
# Function to read a raster and convert it to an array
def raster_to_array(raster_path):
    raster = gdal.Open(raster_path)
    band = raster.GetRasterBand(1)
    array = band.ReadAsArray()
    return array
# Function to write an array to a raster, using a reference raster for georeferencing
def array_to_raster(array, reference_raster_path, output_raster_path):
    reference_raster = gdal.Open(reference_raster_path)
    driver = gdal.GetDriverByName('GTiff')
    out_raster =driver.Create(output_raster_path, reference_raster.RasterXSize, reference_raster.RasterYSize, 1, gdal.GDT_Float32)
    out_raster.SetGeoTransform(reference_raster.GetGeoTransform())
    out_raster.SetProjection(reference_raster.GetProjection())
    out_band = out_raster.GetRasterBand(1)
    out_band.WriteArray(array)
    out_band.FlushCache()
    out_raster = None
```

```
# Calculate the slope in percentage from the elevation data
def calculate_slope_percentage(elevation_array, cell_size):
    # Calculate slope in radians
    x, y = np.gradient(elevation_array, cell_size)
    slope_radians = np.arctan(np.sqrt(x**2 + y**2))
    # Convert to slope percentage
    slope_percentage = np.tan(slope_radians) * 100
    return slope_percentage

# Calculate the LS factor from the slope percentage
def calculate_ls_factor(slope_percentage):
    LS4 = np.sqrt(500 / 100)
    LS3 = slope_percentage * 0.53
    LS2 = slope_percentage * (slope_percentage * 0.076)
    LS1 = LS3 + LS2 + 0.76
    LS = LS1 * LS4
    return LS

# Load your SRTM elevation raster image
elevation_raster_path = r'/content/gdrive/MyDrive/Rusle/Rusle project/All_factor/Input/Digital_Elevation_Model_Data.tif'
elevation_array = raster_to_array(elevation_raster_path)

# The cell size (resolution) of your SRTM data, in meters
cell_size = 100 # SRTM data is commonly available at 30m resolution
```

```
# Calculate the slope percentage from the elevation data
slope_percentage_array = calculate_slope_percentage(elevation_array,
cell_size)

# Calculate the LS factor from the slope percentage
ls_factor_array = calculate_ls_factor(slope_percentage_array)

# Save the LS factor as a new raster image
output_raster_path = r'/content/gdrive/MyDrive/rusle1/LS_factor1.tif'
array_to_raster(ls_factor_array, elevation_raster_path, output_raster_path)

print("LS factor raster created successfully.")
```

Then by use of LS_factor1.tif we have created the map of LS_factor in ArcGIS.



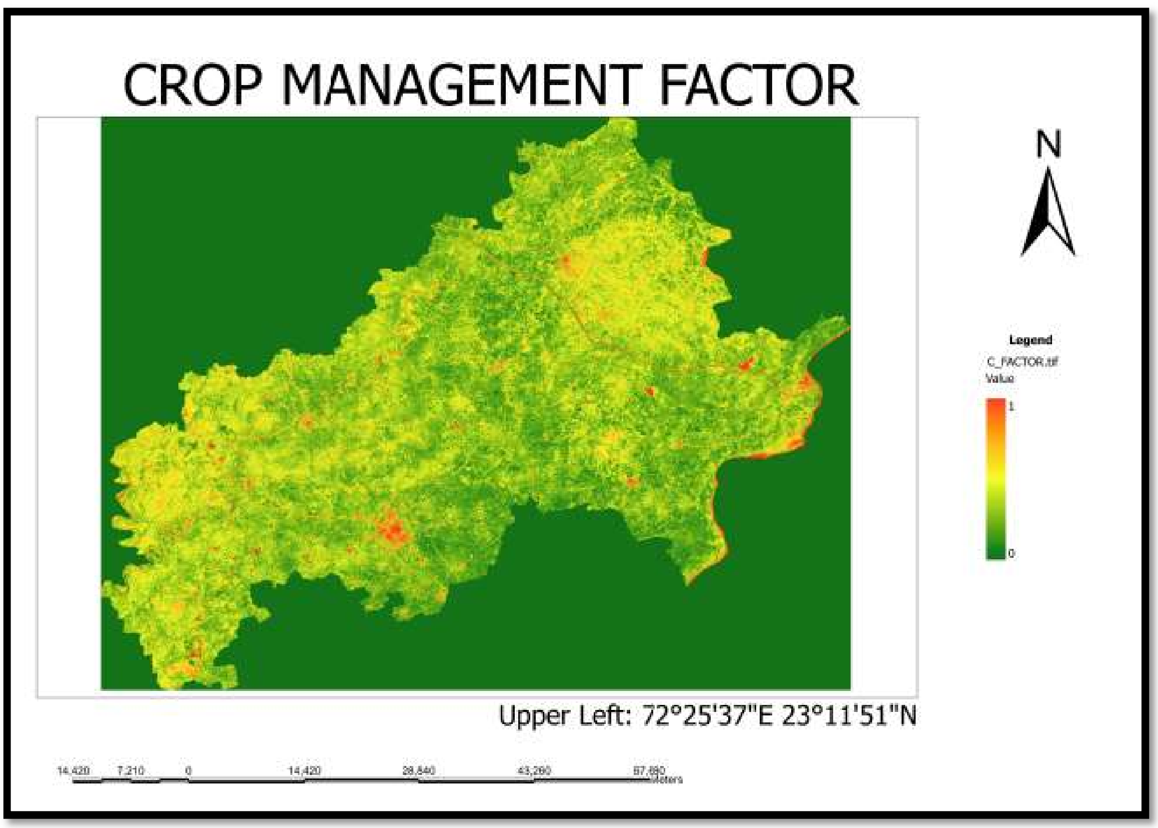
Step 4: Calculate and create C factor raster using python then created the map of output image using ArcGIS.

```
#4. C FACTOR

import numpy as np

from osgeo import gdal

# Function to read a raster and convert it to an array

def raster_to_array(raster_path):

    raster = gdal.Open(raster_path)

    band = raster.GetRasterBand(1)

    array = band.ReadAsArray()

    return array

# Function to write an array to a raster, using a reference raster for
georeferencing

def array_to_raster(array, reference_raster_path, output_raster_path):

    reference_raster = gdal.Open(reference_raster_path)

    driver = gdal.GetDriverByName('GTiff')

    out_raster = driver.Create(output_raster_path, reference_raster.RasterXSize,
reference_raster.RasterYSize, 1, gdal.GDT_Float32)

    out_raster.SetGeoTransform(reference_raster.GetGeoTransform())

    out_raster.SetProjection(reference_raster.GetProjection())

    out_band = out_raster.GetRasterBand(1)

    out_band.WriteArray(array)

    out_band.FlushCache()

    out_raster = None
```

```
# Calculate the C factor from the NDVI data
def calculate_c_factor(ndvi_array):
    # Define alpha and beta values
    alpha = -2
    beta = 1
    # Apply the equation to calculate C3
    C1 = alpha * ndvi_array
    one_array = np.ones_like(ndvi_array)
    C2 = one_array - ndvi_array
    C3 = C1 / C2

    # Calculate C4
    C4 = np.exp(C3)

    # Handle NaN values
    C4[np.isnan(C4)] = 0

    min_x = np.min(C4)
    max_x = np.max(C4)

    # Perform min-max normalization only on valid values
    C_factor = np.zeros_like(C4)
    valid_mask = C4 != 0
    C_factor[valid_mask] = (C4[valid_mask] - min_x) / (max_x - min_x)
    return C_factor
```

```
# Load your NDVI raster image

ndvi_raster_path = r'/content/gdrive/MyDrive/Rusle/Rusle project/All_factor/Input/NDVI_Data.tif'

ndvi_array = raster_to_array(ndvi_raster_path)

# Calculate the C factor from the NDVI data

c_factor_array = calculate_c_factor(ndvi_array)

# Save the C factor as a new raster image

output_raster_path = r'/content/gdrive/MyDrive/rusle1/C_factor1.tif'

array_to_raster(c_factor_array, ndvi_raster_path, output_raster_path)

print("C factor raster created successfully.")
```

Then by use of C_factor1.tif we have created the map of C_factor in ArcGIS



Step 5: Calculate and create P factor raster using python then created the map of output image using ArcGIS.

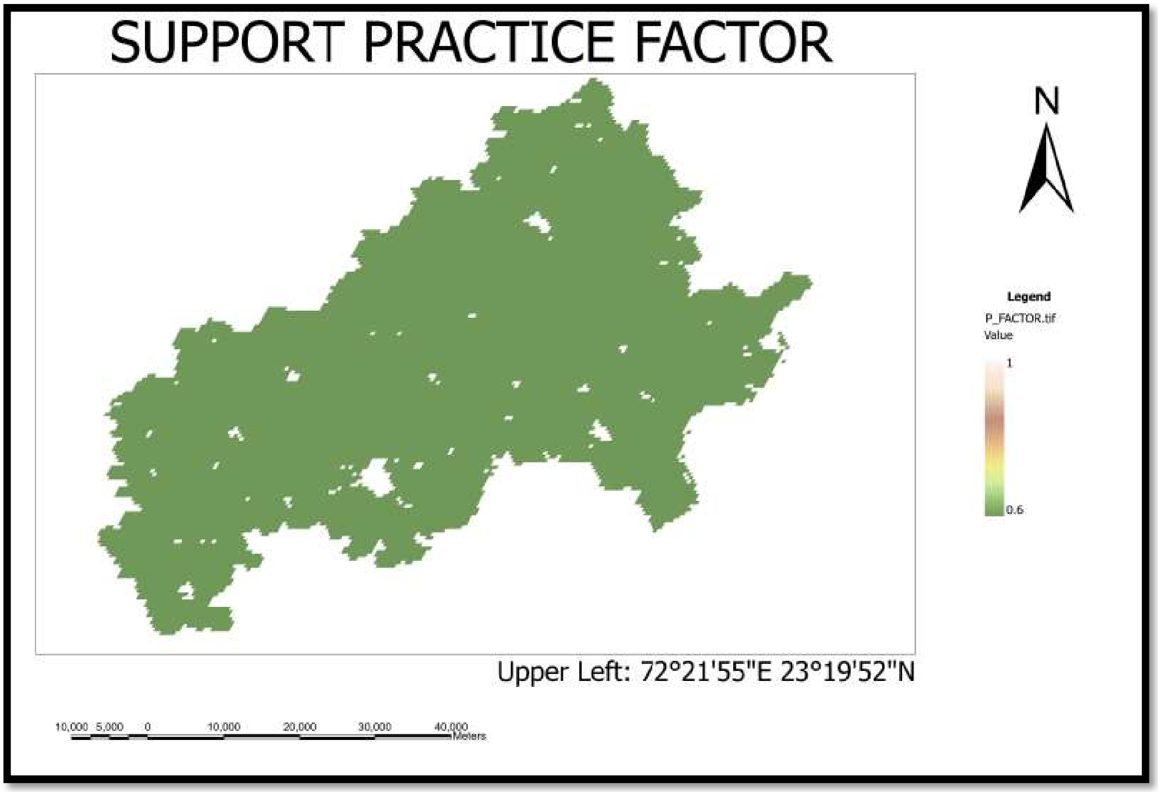
Input:

```
#5.P_FACTOR

import numpy as np

from osgeo import gdal


# Function to read a raster and convert it to an array

def raster_to_array(raster_path):

    raster = gdal.Open(raster_path)

    band = raster.GetRasterBand(1)

    array = band.ReadAsArray()

    return array


# Load LULC raster image

lulc_raster_path = r'/content/gdrive/MyDrive/Rusle/Rusle project/All_factor/Input/Land_Cover_Type_Data_LULC.tif'  # Replace with your LULC raster path

lulc_array = raster_to_array(lulc_raster_path)

lulc_array = np.resize(lulc_array, np.shape(lulc_array))



# Load slope raster image

slope_raster_path = r'/content/gdrive/MyDrive/rusle1/SLOPE_In_PERCENTAGE.tif'  # Replace with your slope raster path

slope_array = raster_to_array(slope_raster_path)

slope_array = np.resize(slope_array,np.shape(lulc_array))
```

```
# Create P Factor map using an expression
P_factor = np.where(
    (lulc_array < 11), 0.8,
    np.where(lulc_array == 11, 1,
        np.where(lulc_array == 13, 1,
            np.where(lulc_array > 14, 1,
                np.where((slope_array < 2) & ((lulc_array == 12) | (lulc_array ==
14)), 0.6,
                    np.where((slope_array < 5) & ((lulc_array == 12) | (lulc_array ==
14)), 0.5,
                        np.where((slope_array < 8) & ((lulc_array == 12) | (lulc_array
== 14)), 0.5,
                            np.where((slope_array < 12) & ((lulc_array == 12) |
(lulc_array == 14)), 0.6,
                                np.where((slope_array < 16) & ((lulc_array == 12) |
(lulc_array == 14)), 0.7,
                                    np.where((slope_array < 20) & ((lulc_array == 12) |
(lulc_array == 14)), 0.8,
                                        np.where((slope_array > 20) & ((lulc_array == 12) |
(lulc_array == 14)), 0.9, 1)
                                    )
                                )
                            )
                        )
                    )
                )
            )
        )
    )
)
```

```
# Save the P factor as a new raster image
output_raster_path = r'/content/gdrive/MyDrive/rusle1/P_FACTOR.tif'

# Use one of the rasters as a reference for georeferencing (in this case, using LULC raster)
reference_raster_path = lulc_raster_path

# Load reference raster for georeferencing
reference_raster = gdal.Open(reference_raster_path)
driver = gdal.GetDriverByName('GTiff')
out_raster = driver.Create(output_raster_path, reference_raster.RasterXSize, reference_raster.RasterYSize, 1, gdal.GDT_Float32)
out_raster.SetGeoTransform(reference_raster.GetGeoTransform())
out_raster.SetProjection(reference_raster.GetProjection())

# Write P factor array to the raster
out_band = out_raster.GetRasterBand(1)
out_band.WriteArray(P_factor)
out_band.FlushCache()
out_raster = None

print("P factor raster created successfully.")
```

Then by use of P_factor1.tif we have created the map of P_factor in ArcGIS.



Step6: All the raster images have different maximum and minimum values therefore we have to normalize it before final RUSLE multiplication.

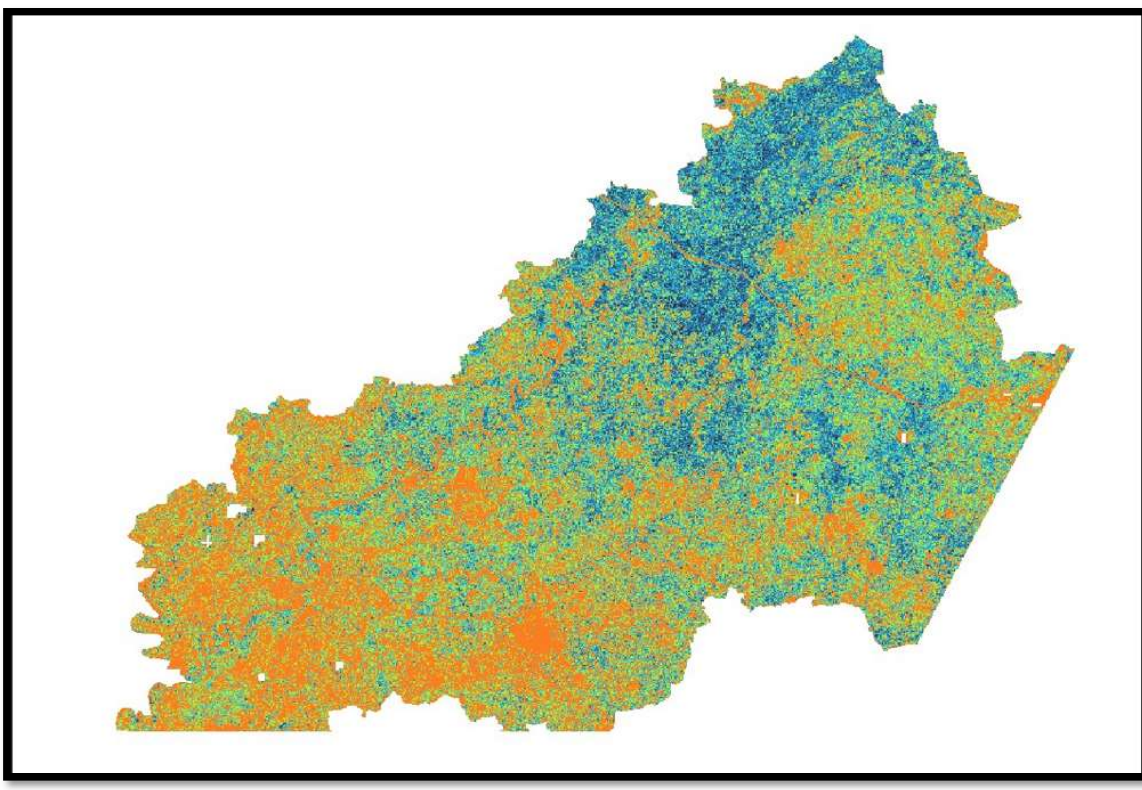
Step 7: Then calculate RUSLE by multiply R_FACTOR_RES.tif, K_FACTOR_RES.tif, LS_FACTOR_RES.tif, C_FACTOR_RES.tif, P_FACTOR_RES.tif in python

Input:

```
# FINAL RUSLE
import rasterio
from rasterio.plot import show
import numpy as np
import matplotlib.pyplot as plt
def read_tiff(file_path):
    with rasterio.open(file_path) as src:
        array = src.read(1)  # Assuming single-band TIFFs
    return array
def write_tiff(file_path, data_array, template_tiff):
    with rasterio.open(template_tiff) as template_src:
        profile = template_src.profile.copy()
    with rasterio.open(file_path, 'w', **profile) as dst:
        dst.write(data_array, 1)
# Replace these paths with your actual file paths
path_r = r'/content/gdrive/MyDrive/rusle1/R_FACTOR_RES.tif'
path_k = r'/content/gdrive/MyDrive/rusle1/K_FACTOR_RES.tif'
path_ls = r'/content/gdrive/MyDrive/rusle1/LS_FACTOR_RES.tif'
path_c = r'/content/gdrive/MyDrive/rusle1/C_FACTOR_RES.tif'
path_p = r'/content/gdrive/MyDrive/rusle1/P_FACTOR_RES.tif'
```

```
# Read TIFF files
array_r = read_tiff(path_r)
array_k = read_tiff(path_k)
array_ls = read_tiff(path_ls)
array_c = read_tiff(path_c)
array_p = read_tiff(path_p)
# Perform multiplication
result_array = array_r * array_k * array_ls * array_c * array_p
# Write result to a new TIFF file
output_tiff_path = r'/content/gdrive/MyDrive/rusle1/RUSLE.tif'
template_tiff_path = r'/content/gdrive/MyDrive/rusle1/LS_FACTOR_RES.tif'
write_tiff(output_tiff_path, result_array, template_tiff_path)
```

Output:



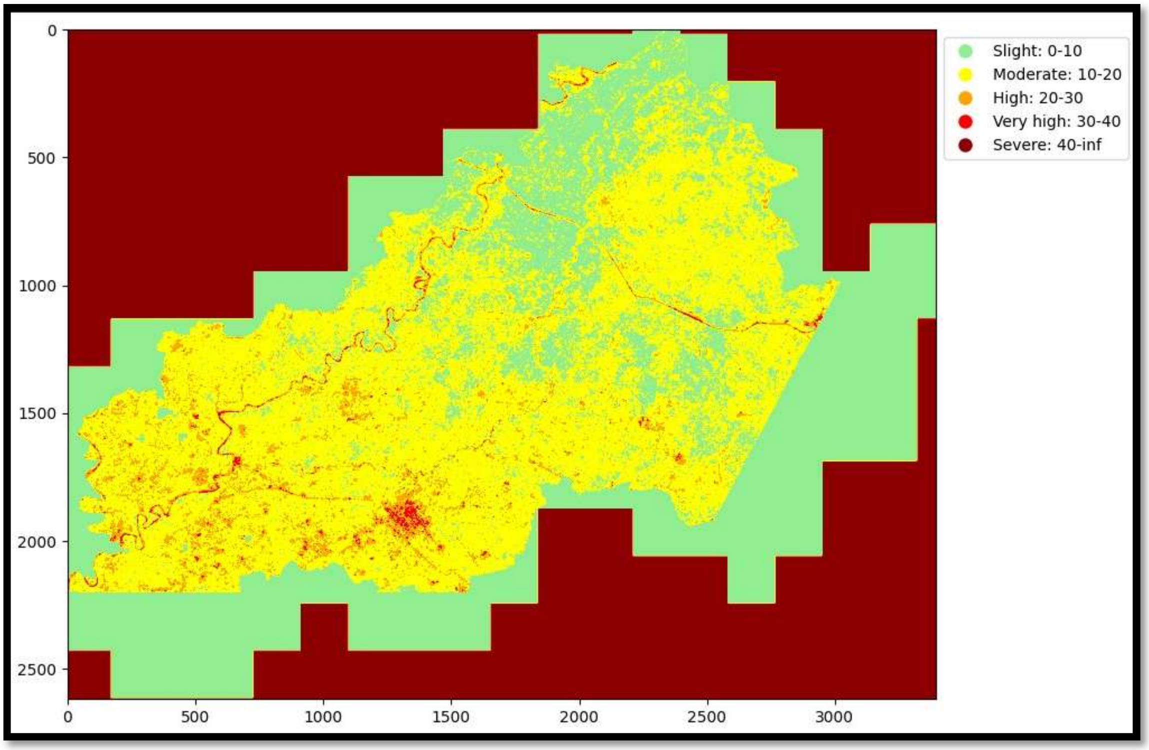
Step 8: Generated final calculated RUSLE.tif file.

Input:

```
import numpy as np
import matplotlib.pyplot as plt
import rasterio
from matplotlib.colors import ListedColormap

# Specify the path to your input raster file
input_raster_path = r'/content/gdrive/MyDrive/rusle1/RUSLE.tif'

# Open the raster file
with rasterio.open(input_raster_path) as src:
    # Read the raster data
    data = src.read(1, masked=True)  # Assuming the data is in the first band

    # Define the categories and corresponding color map
    categories = ['Slight', 'Moderate', 'High', 'Very high', 'Severe']
    color_values = [0, 10, 20, 30, 40, np.inf]
    colors = ['lightgreen', 'yellow', 'orange', 'red', 'darkred']

    # Create a colormap
    cmap = ListedColormap(colors)

    # Create a legend
    legend_labels = [f'{categories[i]}: {color_values[i]}-{color_values[i+1]}' for
i in range(len(categories))]
```

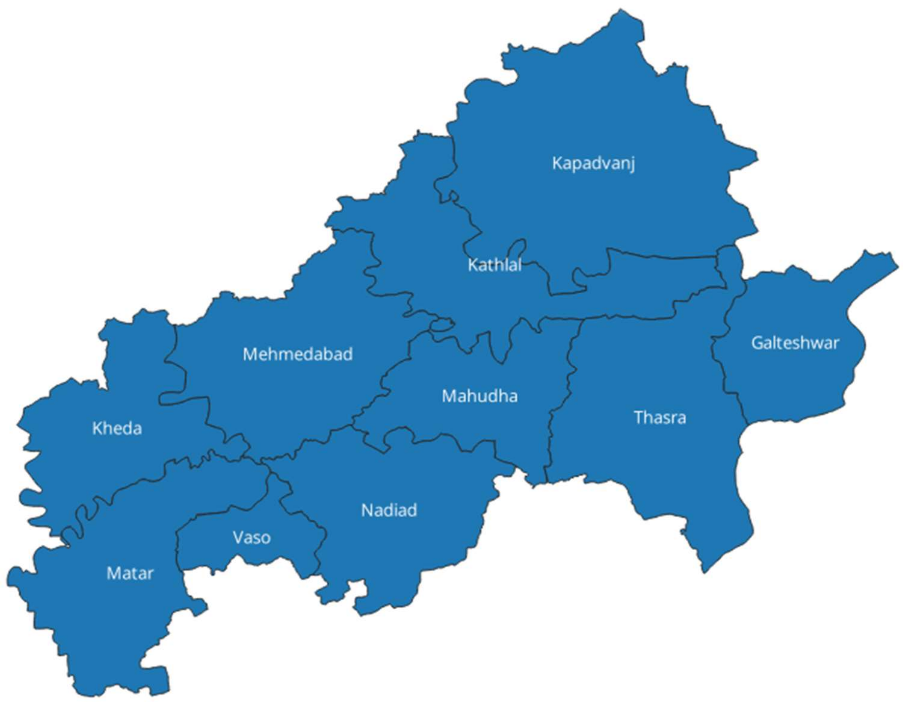
```
    legend = [plt.Line2D([0], [0], marker='o', color='w',
markerfacecolor=colors[i], markersize=10, label=legend_labels[i]) for i in
range(len(colors))]

    # Categorize the data
    category_data = np.digitize(data, bins=color_values, right=True)

    # Create a figure and axis
    fig, ax = plt.subplots(figsize=(10, 10))

    # Plot the categorized data
    im = ax.imshow(category_data, cmap=cmap)

    # Add the legend
    ax.legend(handles=legend, loc='upper left', bbox_to_anchor=(1, 1))

    # Show the plot
    plt.show()
    output_image_path = r'/content/gdrive/MyDrive/rusle1/final_rusle.tif'
    plt.savefig(output_image_path, bbox_inches='tight', pad_inches=0.1)
```

Output:


Slight: 0-10
Moderate: 10-20
High: 20-30
Very high: 30-40
Severe: 40-inf
0
500
1000
1500
2000
2500
3000

Talukawise mean calculation of soil erosion

We calculated the mean value of soil erosion for each talukas of kheda district.

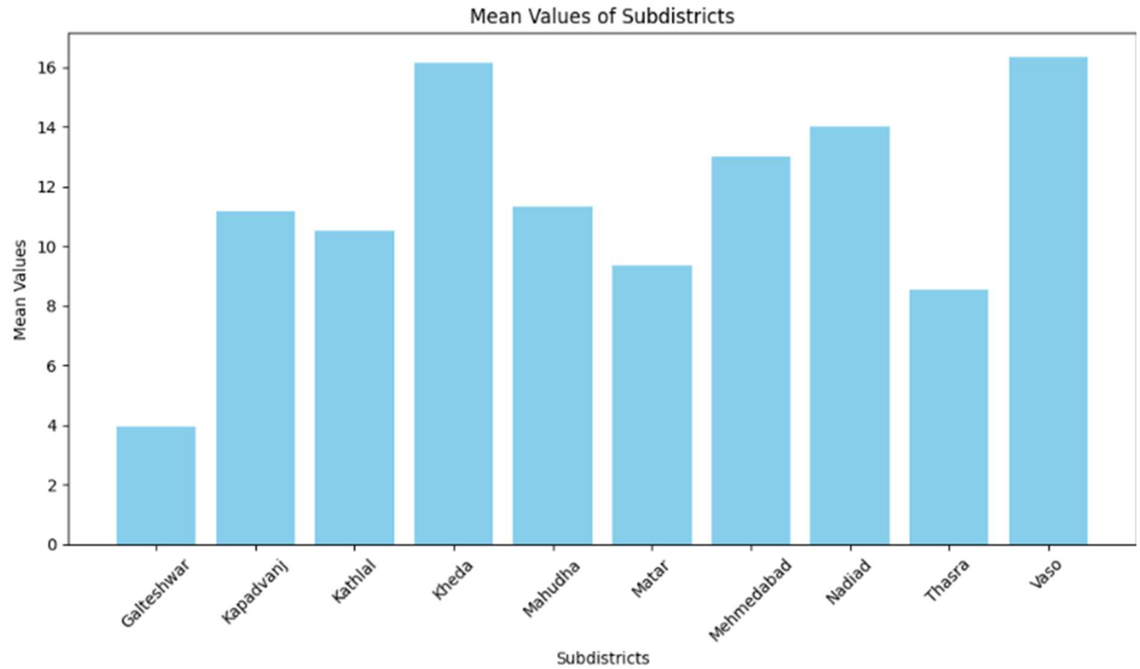


subdistric	_mean
Galteshwar	3.965207394
Kapadvanj	11.16750369
Kathlal	10.53399319
Kheda	16.14962733
Mahudha	11.31283056
Matar	9.362292869
Mehmedabad	13.02221896
Nadiad	14.02985216
Thasra	8.532709776
Vaso	16.33221787

Graph of Mean Values of Subdistricts:

Input:

```
import matplotlib.pyplot as plt

# Data

subdistricts = ['Galteshwar', 'Kapadvanj', 'Kathlal', 'Kheda', 'Mahudha',
'Matar', 'Mehmedabad', 'Nadiad', 'Thasra', 'Vaso']

mean_values = [3.965207394, 11.16750369, 10.53399319, 16.14962733,
11.31283056, 9.362292869, 13.02221896, 14.02985216, 8.532709776,
16.33221787]


# Plotting the bar graph

plt.figure(figsize=(10, 6))  # Adjust the figure size if needed

plt.bar(subdistricts, mean_values, color='skyblue')

plt.xlabel('Subdistricts')

plt.ylabel('Mean Values')

plt.title('Mean Values of Subdistricts')

plt.xticks(rotation=45)  # Rotate x-axis labels for better readability

plt.tight_layout()  # Adjust layout to prevent clipping of labels

plt.show()
```



Conclusion: Highest soil erosion is noticed in kheda and vaso while Galteshwar have lowest value of soil erosion.