中国科学院空天信息创新研究院
Aerospace Information Research Institute (AIR)
Chinese Academy of Sciences (CAS)

INTERNATIONAL RESEARCH CENTER OF BIG DATA FOR SUSTAINABLE DEVELOPMENT GOALS
可持续发展大数据国际研究中心

Practice of Remote Sensing Modeling for Disaster Risk Reduction

Yu Bo

4 September 2025

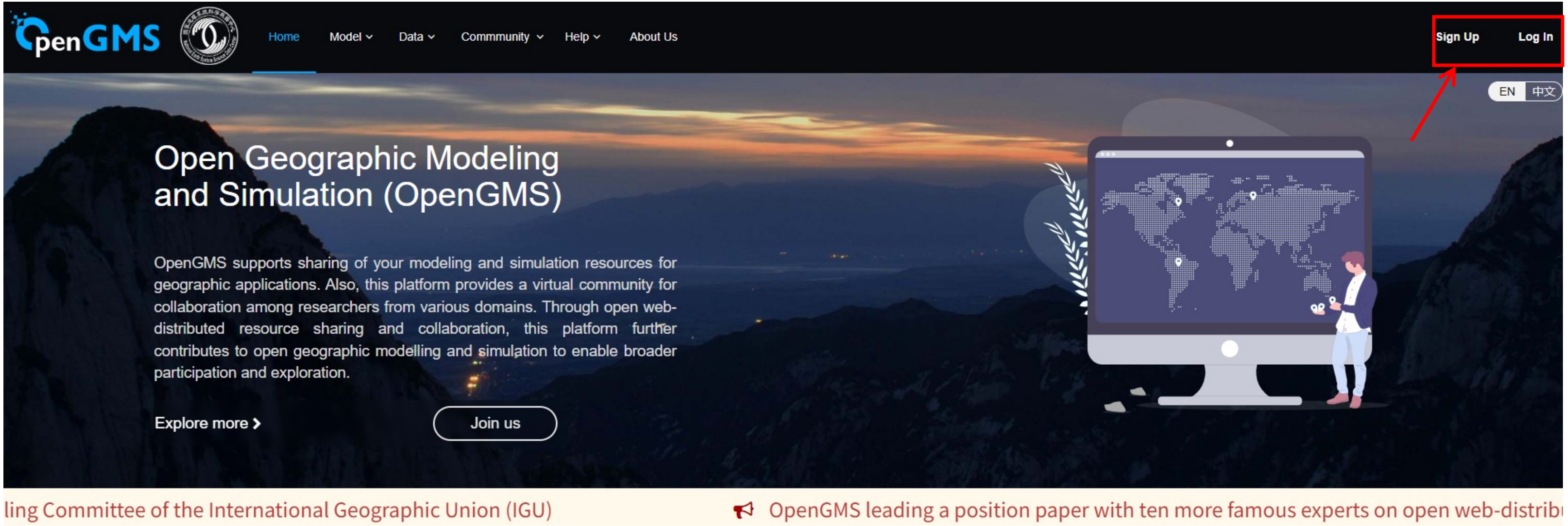
Outline

1. Model Invocation via OpenGMS Portal
2. Annual Image Composite
3. Landslide Sample Production
4. Random Forest Model Training
5. Random Forest Model Testing

1. Model Invocation via OpenGMS Portal

1.1 Instructions for Model Invocation via OpenGMS Portal

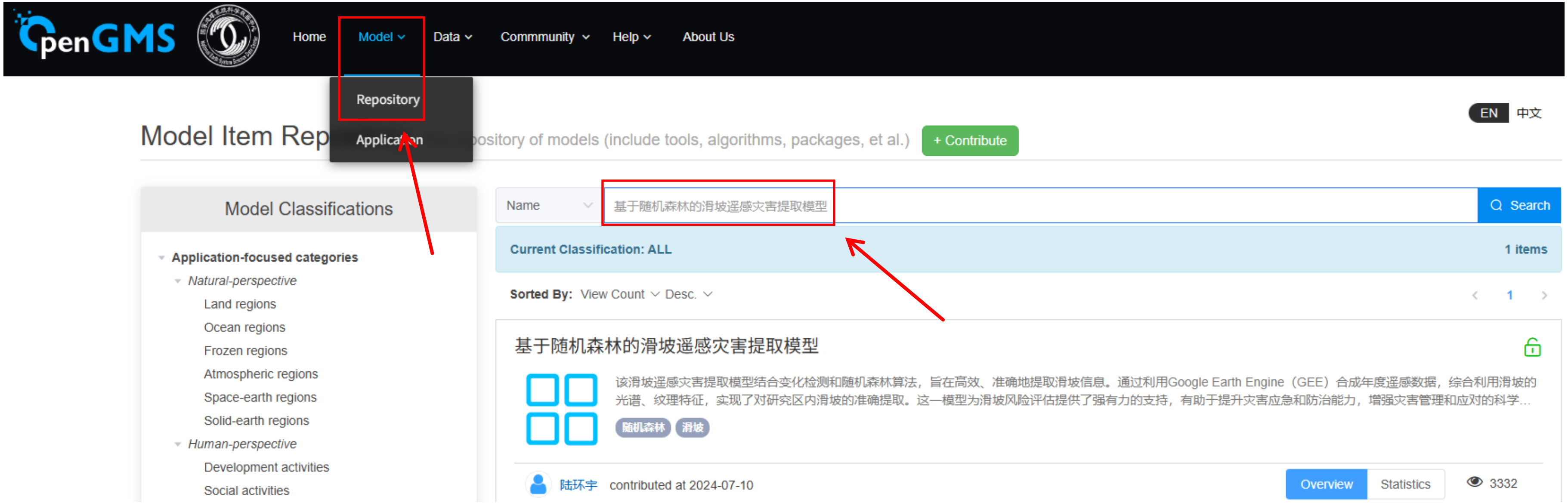
1. Access the OpenGMS portal and log in or register an account. (https://geomodeling.njnu.edu.cn/)



1. Model Invocation via OpenGMS Portal

1.1 Instructions for Model Invocation via OpenGMS Portal

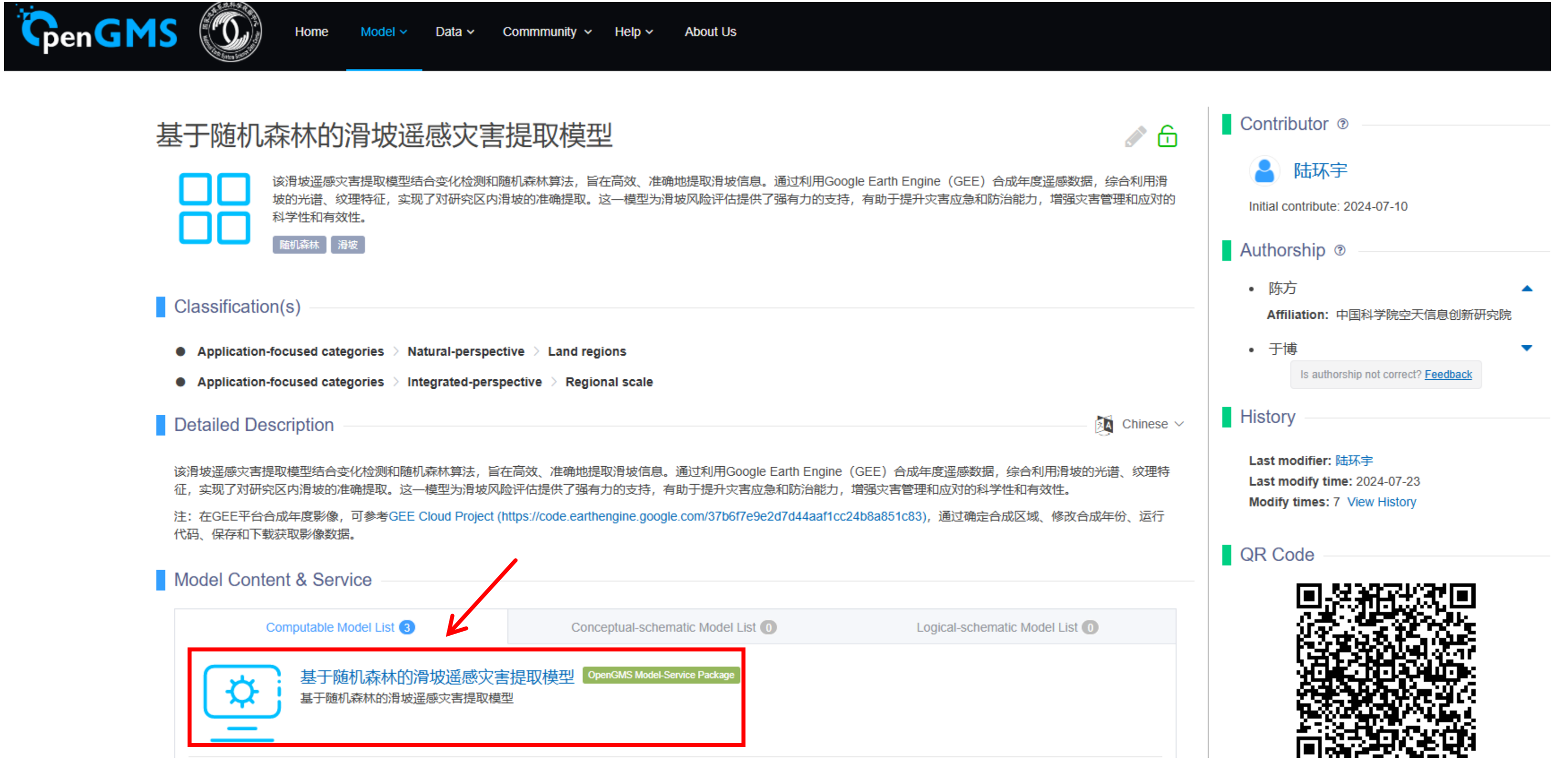
2. Enter the Model/Repository page and search for the model item:
   “基于随机森林的滑坡遥感灾害提取".



1. Model Invocation via OpenGMS Portal

1.1 Instructions for Model Invocation via OpenGMS Portal

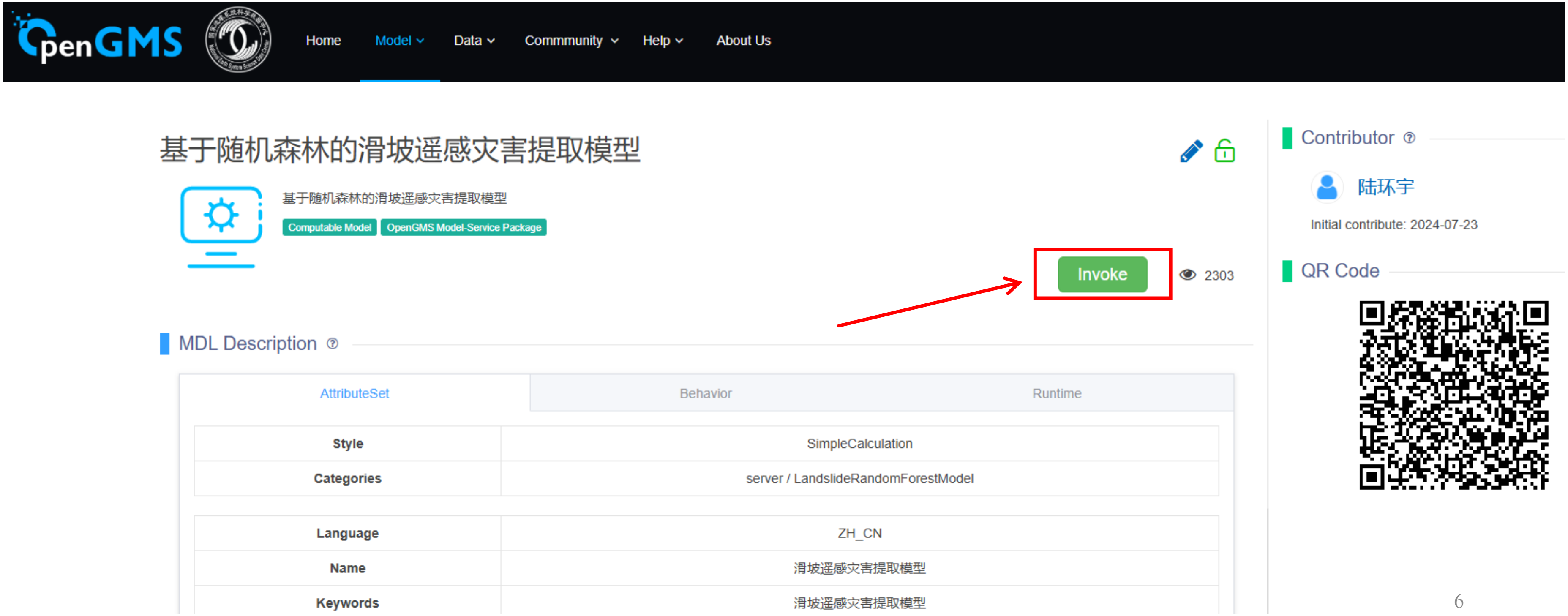
3. Select the model item to view its basic information.



1. Model Invocation via OpenGMS Portal

1.1 Instructions for Model Invocation via OpenGMS Portal

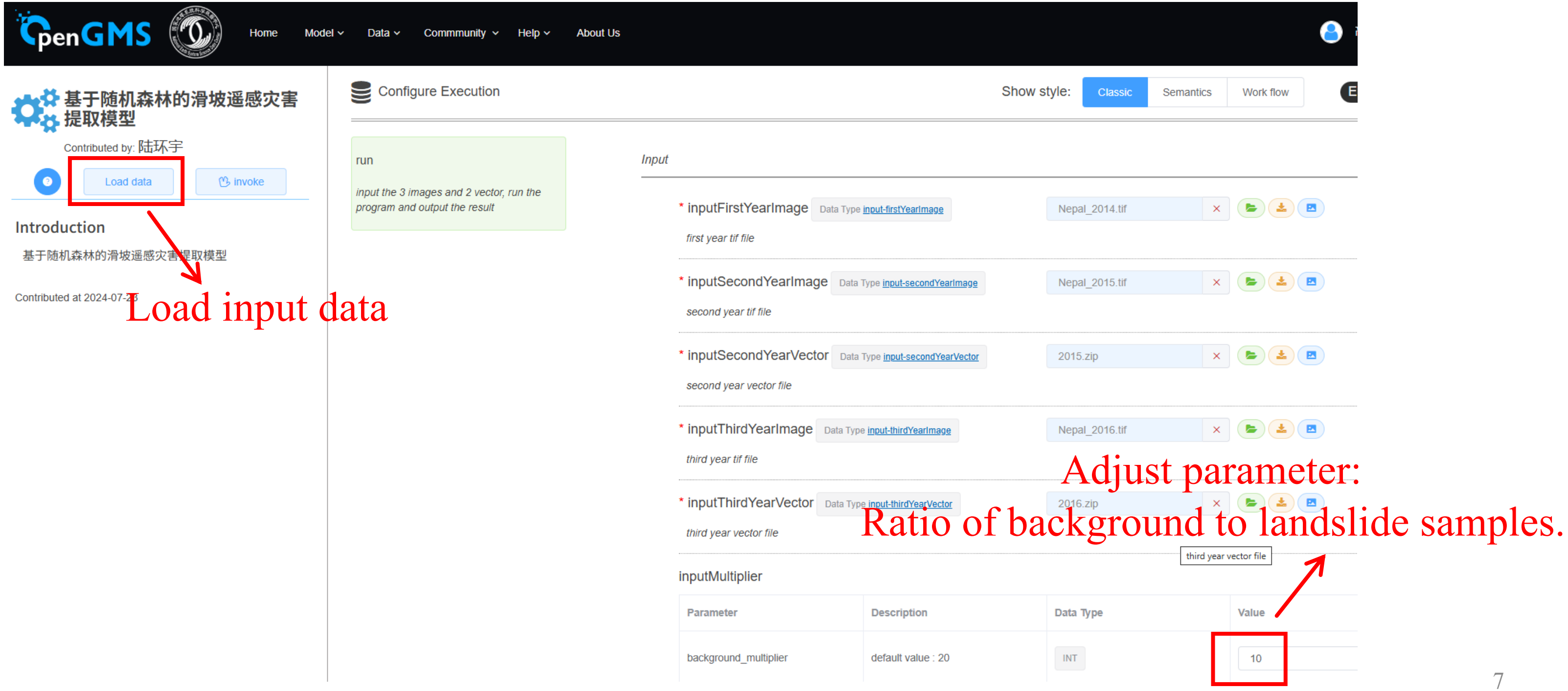
4. From the model item page, select the "Invoke" to open the details page.



1. Model Invocation via OpenGMS Portal

1.1 Instructions for Model Invocation via OpenGMS Portal

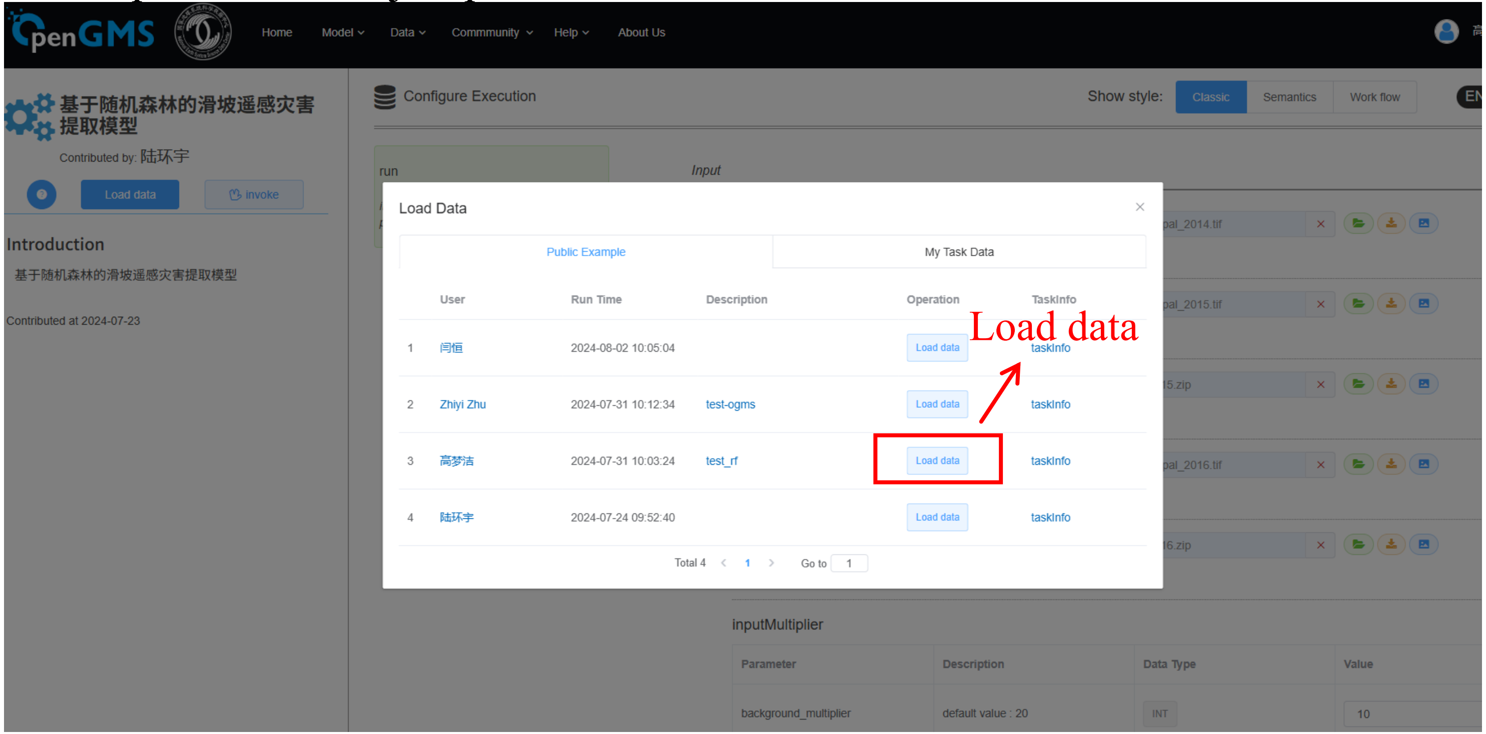
6. Load input data and adjust parameters.



1. Model Invocation via OpenGMS Portal

1.1 Instructions for Model Invocation via OpenGMS Portal

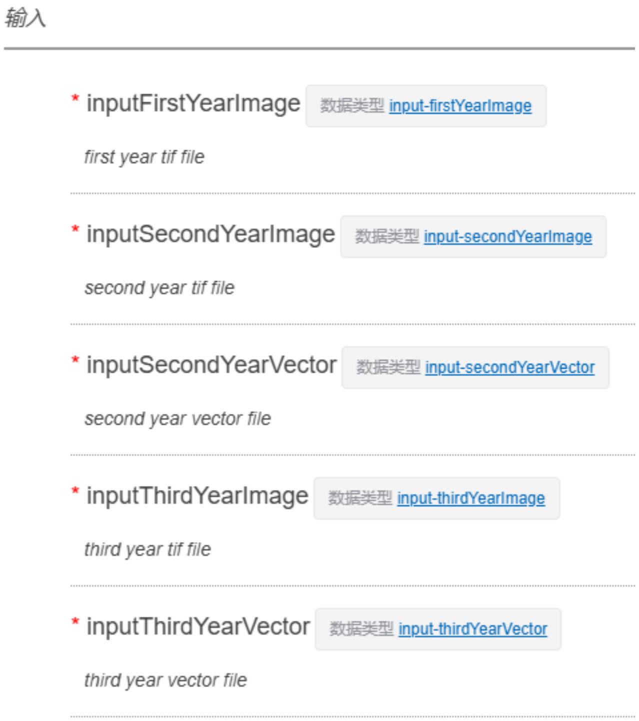
6. Load input data and adjust parameters.



1. Model Invocation via OpenGMS Portal

1.1 Instructions for Model Invocation via OpenGMS Portal

7. Model Input Data File Format Specification



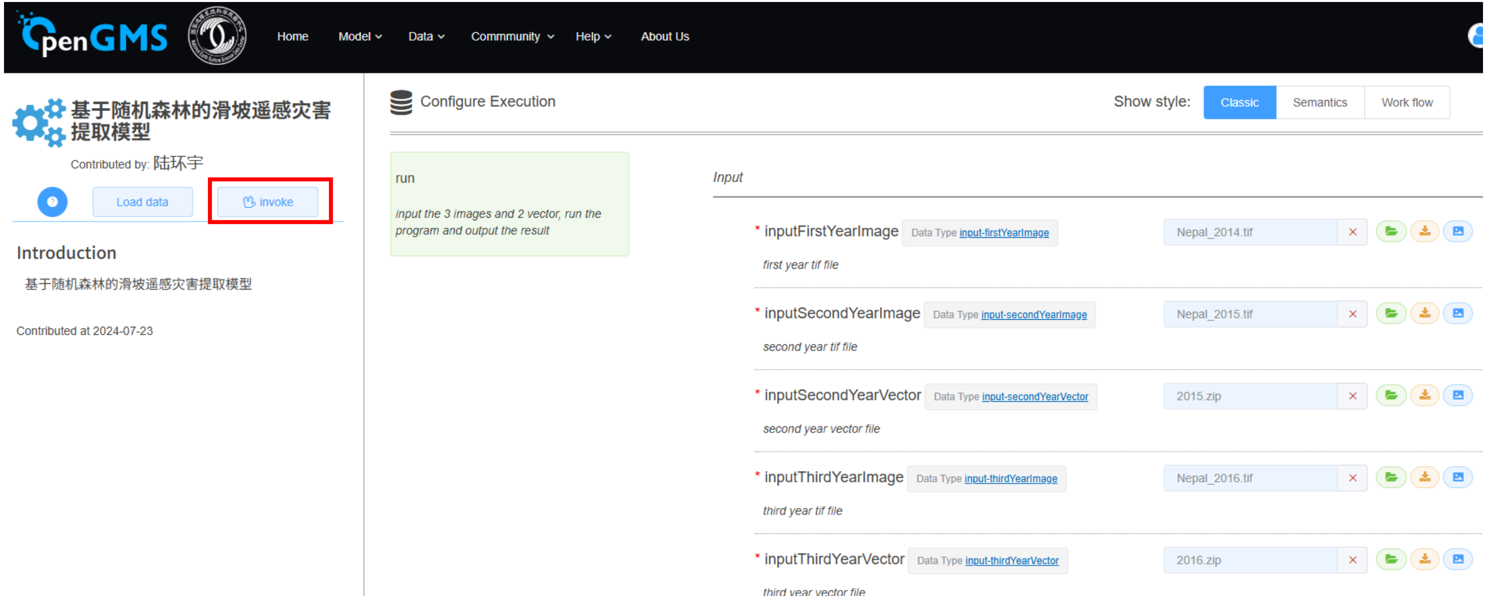
1. **First Year Image:** Pre-processed .tif image of the landslide study area for the first year, generated from the Google Earth Engine (GEE) platform.
2. **Second Year Image:** Pre-processed .tif image for the second year.
3. **Second Year Vector:** Delineated landslide vector data for the second year (zip format).
4. **Third Year Image:** Pre-processed .tif image for the third year.
5. **Third Year Vector:** Delineated landslide vector data for the third year (zip format).

Note: The landslide vector file (ESRI Shapefile) must be packaged into a .zip archive before uploading. Do not include nested folders within the zip file.

1. Model Invocation via OpenGMS Portal

1.1 Instructions for Model Invocation via OpenGMS Portal

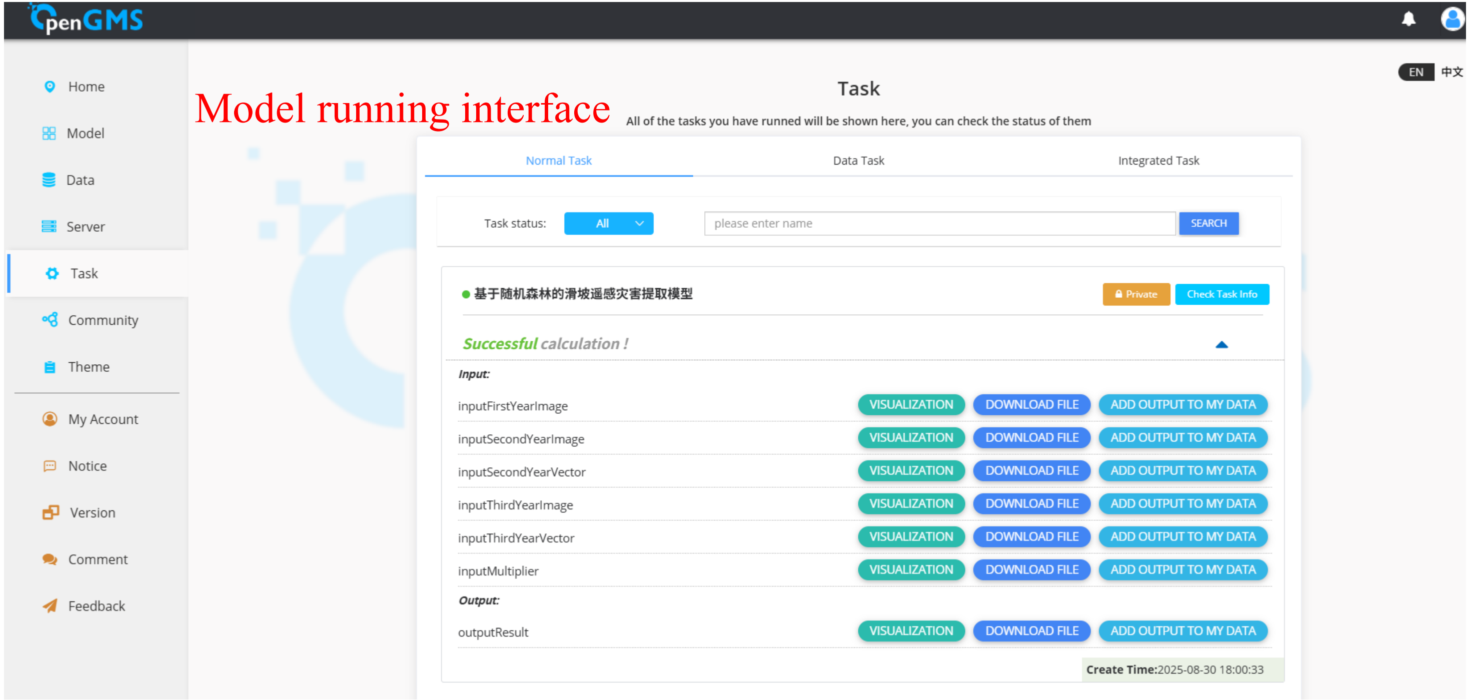
8. Click "Invoke" to initiate the model invocation request. The backend will automatically complete the relevant scheduling strategies and run the model.



1. Model Invocation via OpenGMS Portal

1.1 Instructions for Model Invocation via OpenGMS Portal

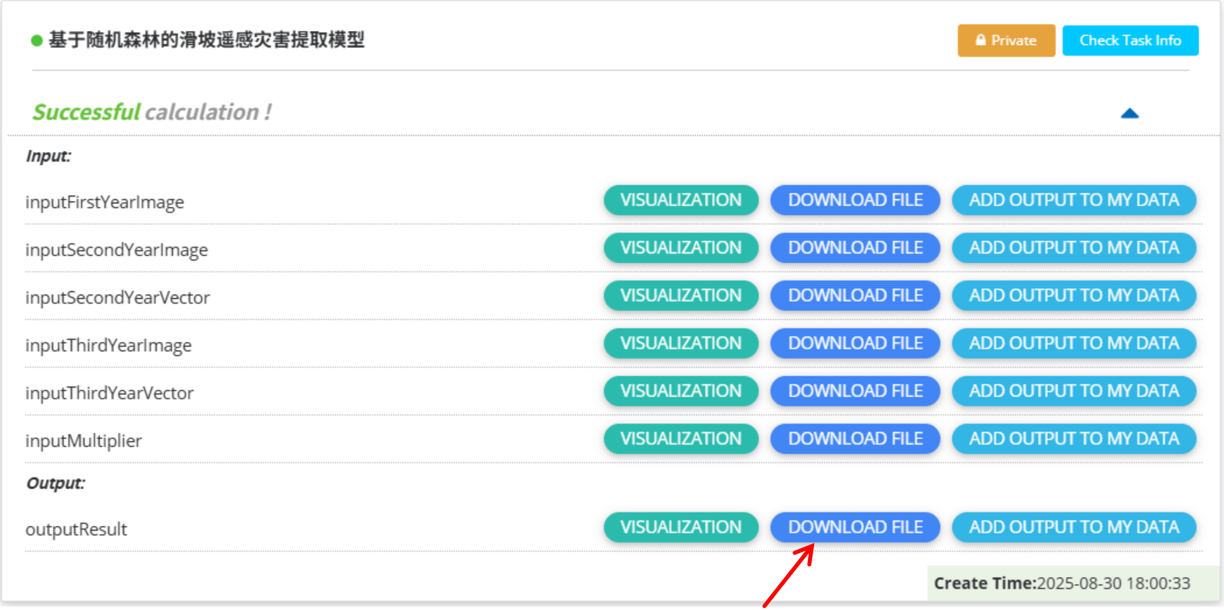
9. The page redirects to the user's task center. Await model completion to download the result file.



1. Model Invocation via OpenGMS Portal

1.1 Instructions for Model Invocation via OpenGMS Portal

10. Upon successful model computation, the output files can be downloaded.



The model's output is packaged into a single zip file, containing intermediate data, training results, and testing results.

1. Model Invocation via OpenGMS Portal

1.1 Instructions for Model Invocation via OpenGMS Portal

11. Model Execution Logic

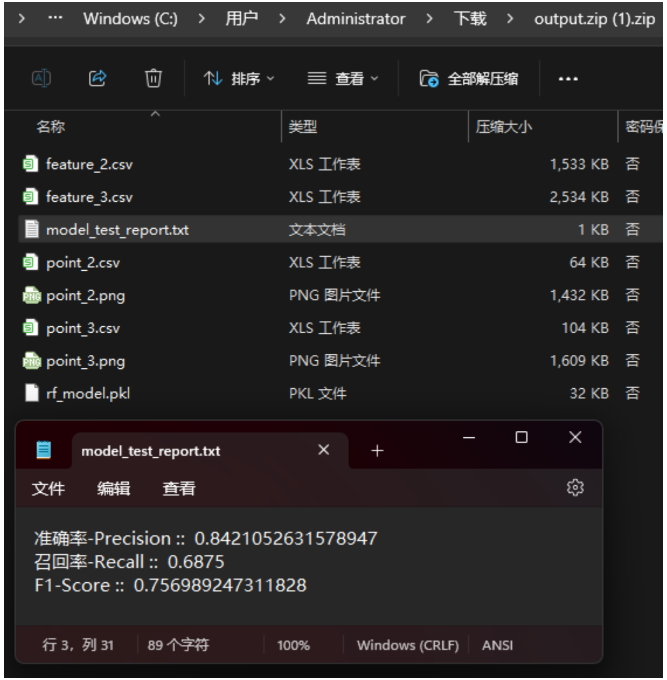
(1) Generate and visualize sample points (Set 2) based on the second-year image and vector data.
(2) Generate and visualize sample points (Set 3) based on the third-year image and vector data.
(3) Generate Feature File 2 based on the first-year image, second-year image, and sample points (Set 2).
(4) Generate Feature File 3 based on the second-year image, third-year image, and sample points (Set 3).
(5) Train the Random Forest model using Feature File 2.
(6) Test the Random Forest model using Feature File 3.

The execution time is considerable (approx. **45+ minutes**) due to the Random Forest training process and limited backend computing resources.

1. Model Invocation via OpenGMS Portal

1.1 Instructions for Model Invocation via OpenGMS Portal

12.1 Model Output Data



12.2 Model Results and File Description

File Name	Description of Output Files
point_2.csv point_3.csv	Sample point files
point_2.png point_3.png	Sample point visualization files
feature_2.csv feature_3.csv	Feature files
rf_model.pkl	Trained Random Forest model file
model_test_report.txt	Model test report file

Outline

1. Model Invocation via OpenGMS Portal
2. Annual Image Composite
3. Landslide Sample Production
4. Random Forest Model Training
5. Random Forest Model Testing

2. Annual Image Composite

2.1 Introduction to GEE Platform

GEE(Google Earth Engine)：

Developed by Google, Carnegie Mellon University, and the US Geological Survey (USGS), GEE is a **cloud-based platform** for processing satellite remote sensing and other Earth observation data.

It integrates Google's powerful servers and large-scale **cloud computing resources**.The platform provides access to massive Earth observation datasets, including **Sentinel, MODIS, Landsat**, as well as vegetation, land surface temperature, and socio-economic datasets. The database is updated daily.

GEE offers **Python and JavaScript APIs**, with a web-based code editor for fast and interactive algorithm development.

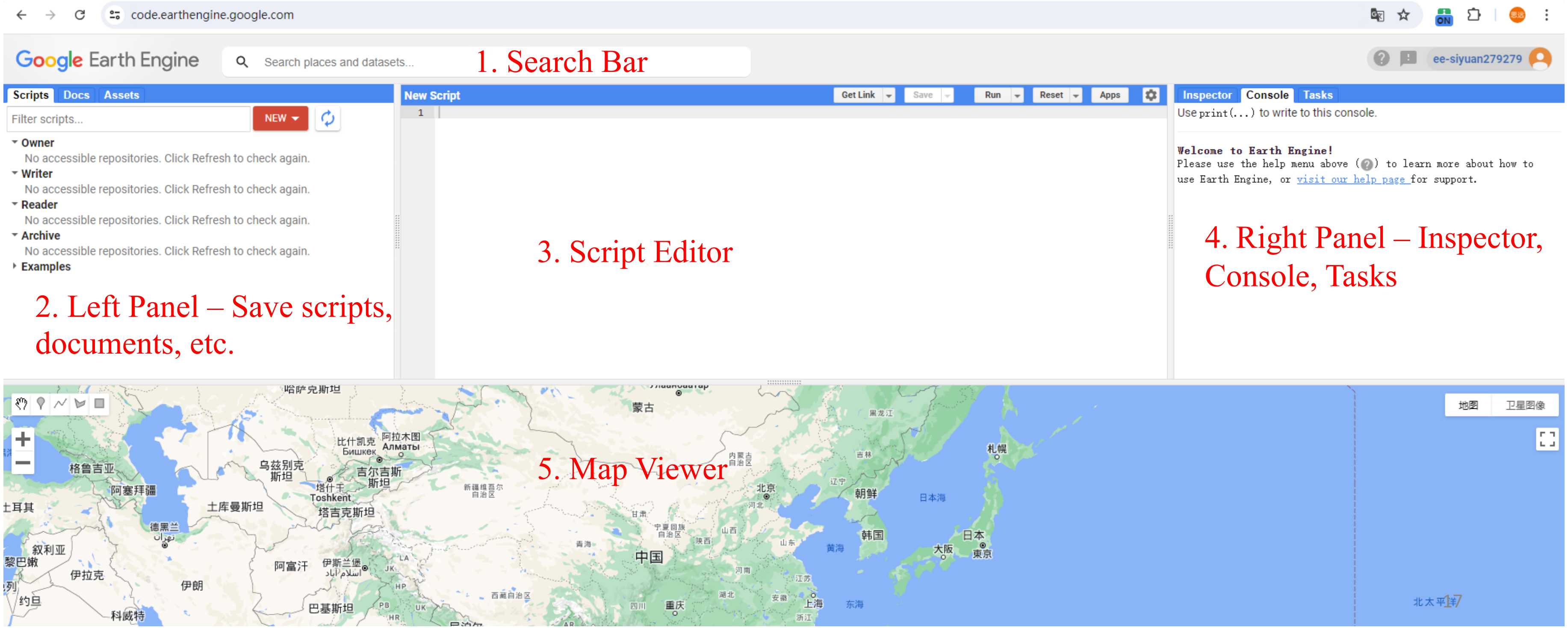
Advantages of GEE：

- Built-in multiple datasets, no need for downloading and preprocessing huge amounts of data.
- Strong computing power supported by Google

2. Annual Image Composite

2.1 Introduction to GEE Platform

https://code.earthengine.google.com



2. Annual Image Composite

2.2 Purpose of Annual Image Composite

Annual image Composite in GEE processes and combines a full year of satellite imagery to overcome limitations of single images, such as cloud cover, shadows, and other interferences, providing clearer and more continuous land surface information.

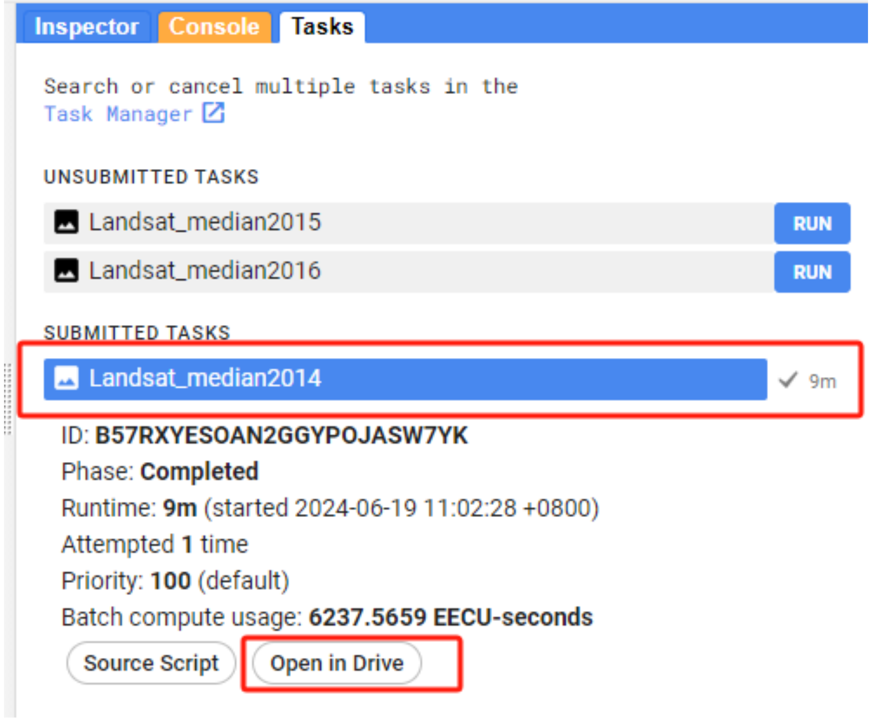
Main Purposes:

- **Reduce cloud cover and noise:** Single satellite images are often affected by clouds, shadows, and atmospheric conditions. Annual composites combine multiple images within a year to remove these effects and generate clearer surface images.
- **Improve temporal resolution :** By compositing images across a whole year, the average surface conditions can be captured. This reduces errors from seasonal variations and provides a more stable representation of the surface.
- **Enhance data quality and consistency:** Combining multiple images helps eliminate problems in individual scenes (such as sensor errors or missing data), resulting in more consistent and higher-quality datasets.

2. Annual Image Composite

2.3 Method of Annual Image Composite

Check and download the image from **Google Drive**.



2. Annual Image Composite

2.4 Analysis of the Annual Image Composite Code

```
// 导入 Landsat 影像集合
var Landsat8 = ee.ImageCollection("LANDSAT/LC08/C02/T1_L2");
var Landsat7 = ee.ImageCollection("LANDSAT/LE07/C02/T1_L2");
var Landsat5 = ee.ImageCollection("LANDSAT/LT05/C02/T1_L2");
var Landsat4 = ee.ImageCollection("LANDSAT/LT04/C02/T1_L2");
```

1. Import Landsat image collections

```
// 定义矩形的边界，使用指定的经纬度
var rectBounds = ee.Geometry.Rectangle([
  84.4442418216, 27.3900821704, 85.7243411014, 28.2538123161
]);
```

2. Define the rectangle boundary using specified longitude and latitude

2. Annual Image Composite

2.4 Analysis of the Annual Image Composite Code

```
// 打印矩形边界信息
print('Rectangle bounds:', rectBounds);

// 在地图上添加矩形图层
Map.addLayer(rectBounds, {color: 'red'}, 'Rectangle');
```

3. Print the rectangle boundary information and add the rectangle layer to the map

```
// 设置地图中心和缩放级别
Map.setCenter(
  (84.4442418216 + 85.7243411014) / 2,
  (27.3900821704 + 28.2538123161) / 2,
  10
); // 中心位置的经纬度和缩放级别
```

4. Set the map center and zoom level

2. Annual Image Composite

2.4 Analysis of the Annual Image Composite Code

```
// 云掩膜函数（适用于 Landsat 8），参考 https://landsat.usgs.gov/landsat-surface-
var cloudMaskL8 = function(image) {
  var qa = image.select('QA_PIXEL');
  var cloud = qa.bitwiseAnd(1 << 2)
                .or(qa.bitwiseAnd(1 << 5))
                .or(qa.bitwiseAnd(1 << 7))
                .or(qa.bitwiseAnd(1 << 4))
                .or(qa.bitwiseAnd(1 << 3))
                .or(qa.bitwiseAnd(1 << 8).and(qa.bitwiseAnd(1 << 9)))
                .or(qa.bitwiseAnd(1 << 10).and(qa.bitwiseAnd(1 << 11)))
                .or(qa.bitwiseAnd(1 << 12).and(qa.bitwiseAnd(1 << 13)))
                .or(qa.bitwiseAnd(1 << 14).and(qa.bitwiseAnd(1 << 15)));
  return image.select([
    'SR_B2','SR_B3','SR_B4','SR_B5','SR_B6','SR_B7'
  ], [
    'blue', 'green', 'red', 'nir', 'swir1', 'swir2'
  ]).updateMask(cloud.not());
};
```

5. Cloud mask function (for Landsat 8)

2. Annual Image Composite

2.4 Analysis of the Annual Image Composite Code

```
// 云掩膜函数（适用于 Landsat 4-7）
var cloudMaskL457 = function(image) {
  var qa = image.select('QA_PIXEL');
  var cloud = qa.bitwiseAnd(1 << 5)
                .or(qa.bitwiseAnd(1 << 7))
                .or(qa.bitwiseAnd(1 << 3))
                .or(qa.bitwiseAnd(1 << 4))
                .or(qa.bitwiseAnd(1 << 8).and(qa.bitwiseAnd(1 << 9)))
                .or(qa.bitwiseAnd(1 << 10).and(qa.bitwiseAnd(1 << 11)))
                .or(qa.bitwiseAnd(1 << 12).and(qa.bitwiseAnd(1 << 13)))
                .or(qa.bitwiseAnd(1 << 14).and(qa.bitwiseAnd(1 << 15)));
  return image.select([
    'SR_B1', 'SR_B2', 'SR_B3', 'SR_B4', 'SR_B5', 'SR_B7'
  ], [
    'blue', 'green', 'red', 'nir', 'swir1', 'swir2'
  ]).updateMask(cloud.not());
};
```

5. Cloud mask function (for Landsat 457)

2. Annual Image Composite

2.4 Analysis of the Annual Image Composite Code

6. Composite annual imagery and export

```
// 合成年度影像并导出
var exportImageOfCertainYear = function(year1) {
  var startDate = year1.toString() + '-01-01';
  var endDate = year1.toString() + '-12-31';

  // 获取各年份的影像集合
  var Landsat8_1 = Landsat8
    .filterDate(startDate, endDate)
    .filterBounds(rectBounds)
    .map(cloudMaskL8);

  var Landsat7_1 = Landsat7
    .filterDate(startDate, endDate)
    .filterBounds(rectBounds);

  var Landsat5_1 = Landsat5
    .filterDate(startDate, endDate)
    .filterBounds(rectBounds);
```

```
  var Landsat4_1 = Landsat4
    .filterDate(startDate, endDate)
    .filterBounds(rectBounds);

  var Landsat457 = Landsat7_1
    .merge(Landsat5_1)
    .merge(Landsat4_1)
    .map(cloudMaskL457);

  var Landsat = Landsat457
    .merge(Landsat8_1)
    .sort('SENSING_TIME');

  var Landsat_median = Landsat.map(function(image) {
    return image.clip(rectBounds);
  }).median();
```

2. Annual Image Composite

2.4 Analysis of the Annual Image Composite Code

```
  Map.addLayer(Landsat_median, {
    bands: ['red', 'green', 'blue'],
    min: 0, max: 3000
  }, 'Landsat_median' + year1.toString());

  Export.image.toDrive({
    image: Landsat_median,
    description: 'Landsat_median' + year1.toString(),
    folder: 'Landsat_median',
    fileNamePrefix: 'scale_along_median_' + year1.toString(),
    region: rectBounds,
    scale: 30,
    maxPixels: 1e13,
    fileFormat: 'GeoTIFF'
  });
  return 0;
};
```

7. Add the annual image to the map view for visualization

2. Annual Image Composite

2.4 Analysis of the Annual Image Composite Code

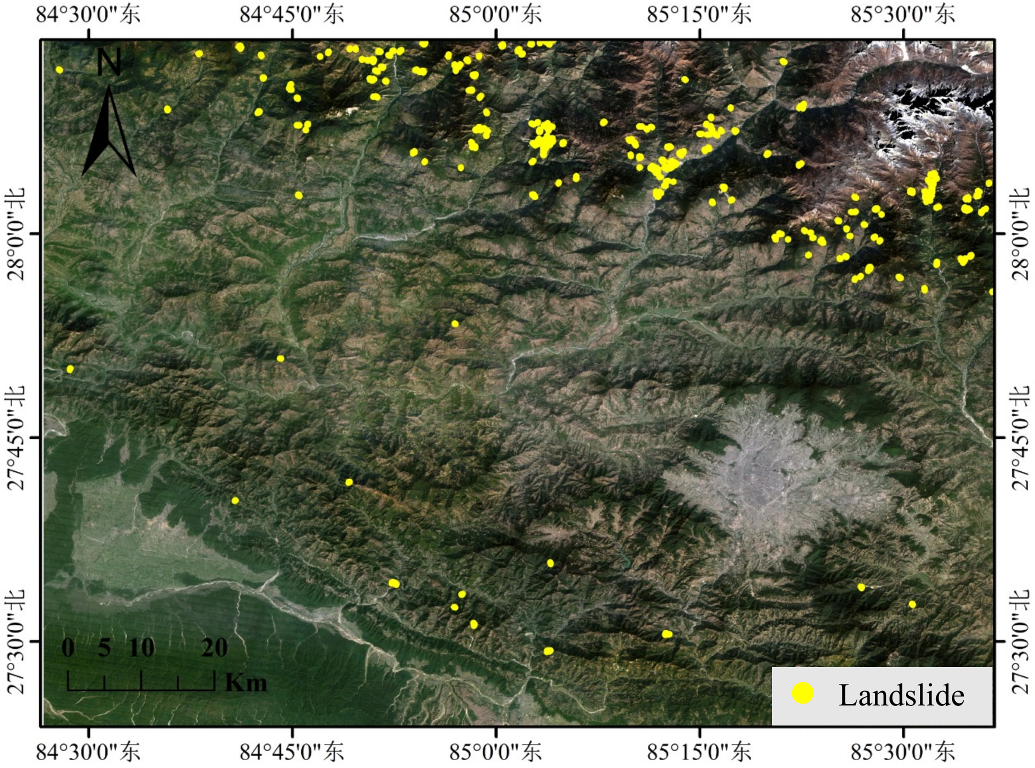
```
// 指定合成的年份范围
var years = [];
for (var i = 2014; i < 2017; i++) {  // 修改合成年份范围
  years.push(i);
}

// 执行年度影像合成并导出
var res = years.map(exportImageOfCertainYear);
```

8. Specify the year range for Composite and export

2. Annual Image Composite

2.5 Demonstration of Annual Image Composites



2016 Annual Data for Nepal

Outline

1. Model Invocation via OpenGMS Portal

2. Annual Image Composite

3. Landslide Sample Production

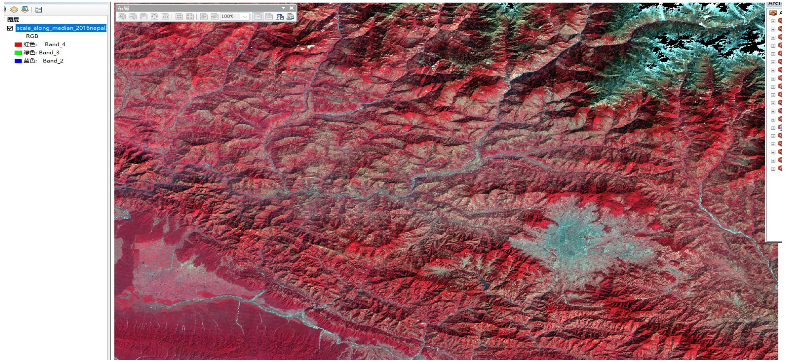
4. Random Forest Model Training

5. Random Forest Model Testing

3. Landslide Sample Production

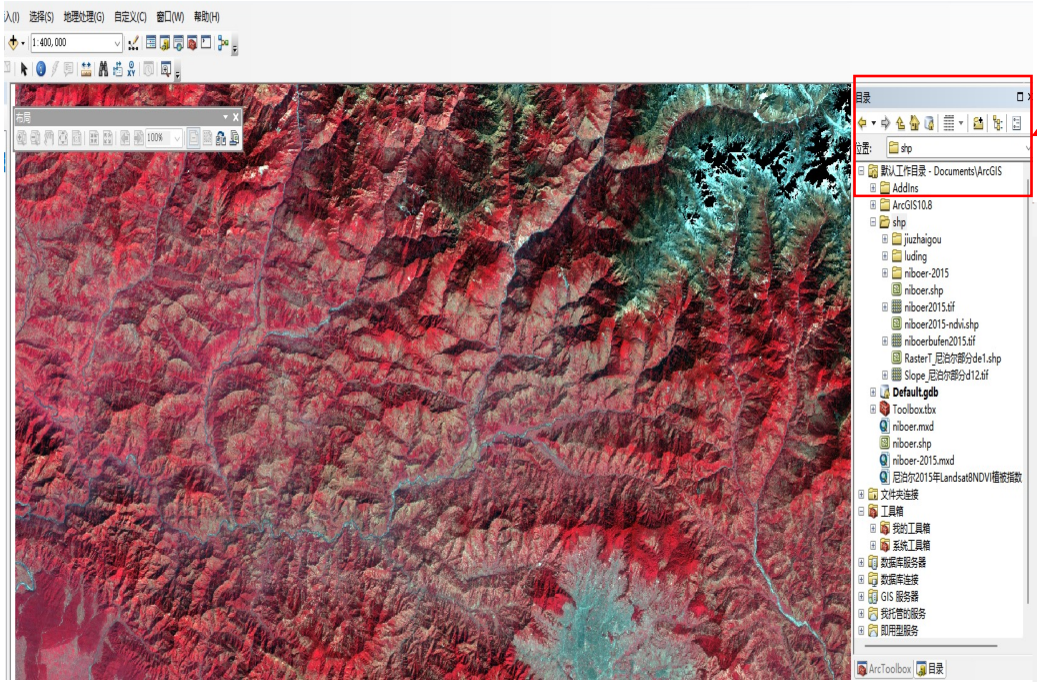
3.1 Landslide Data Preparation

1. Import Nepal remote sensing imagery into ArcGIS
2. Set the image band combination to 4, 3, 2 for a false-color display, which makes it easier to distinguish between landslides and vegetation

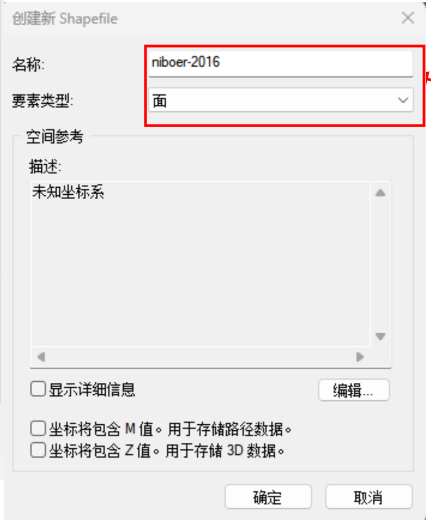


3. Landslide Sample Production

3.2 Creating the Landslide Labeling File



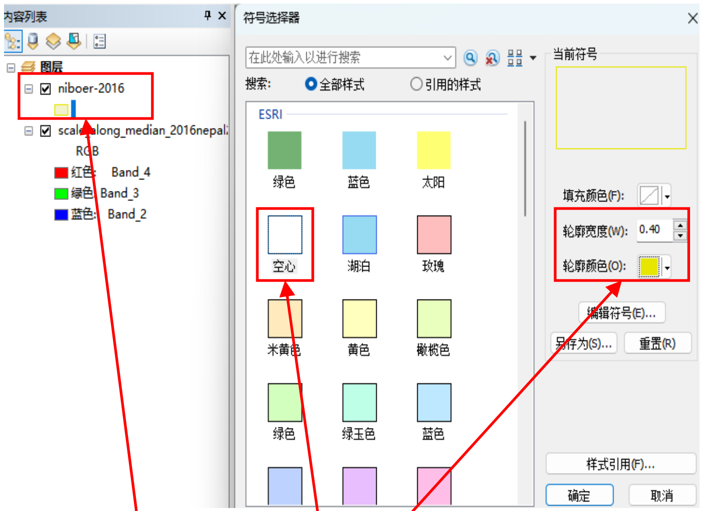
1. Open the Catalog in ArcGIS, create a folder in the default workspace to store the shapefile, and then create a new shapefile for landslide labeling within that folder.



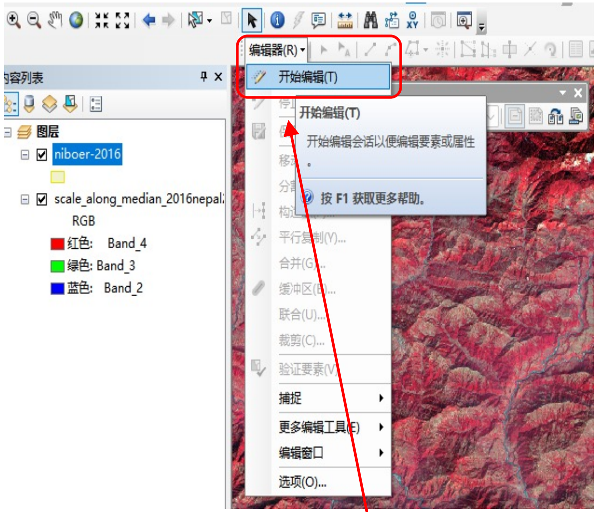
2. Set the name to the study area (Nepal), select "Polygon" as the feature type, and then click OK.

3. Landslide Sample Production

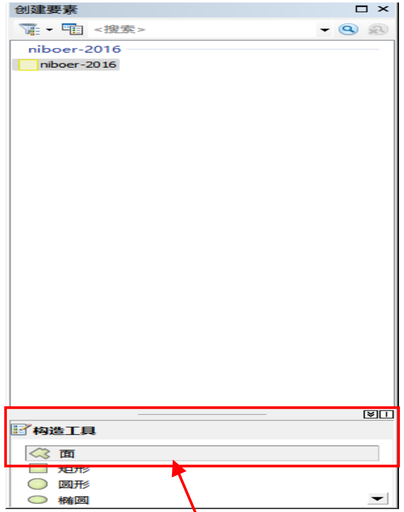
3.3 Preparation for Landslide Labeling



1. Adjust the created shapefile by setting its fill to hollow.



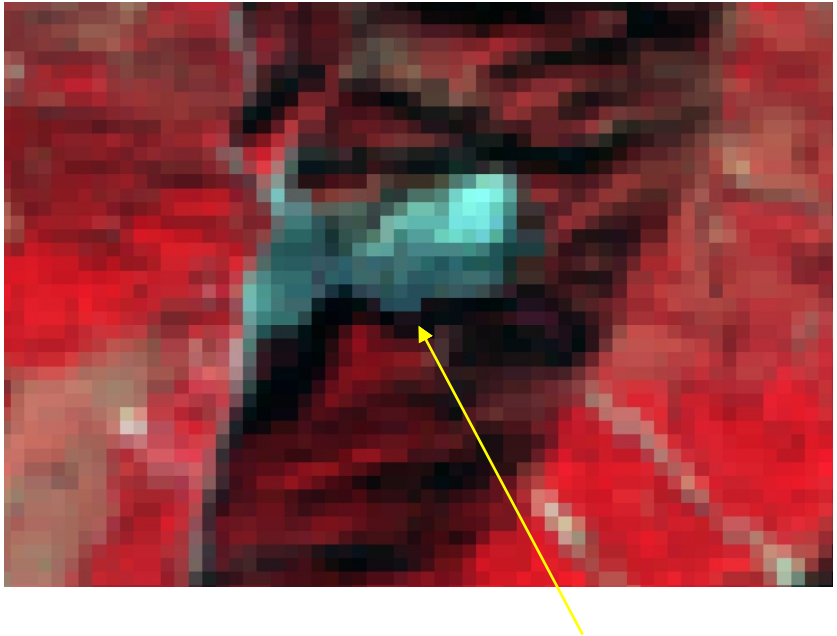
2. Click on the Editor and start an editing session for the shapefile.



3. In the construction tools, select the Polygon tool.

3. Landslide Sample Production

3.4 Landslide Labeling

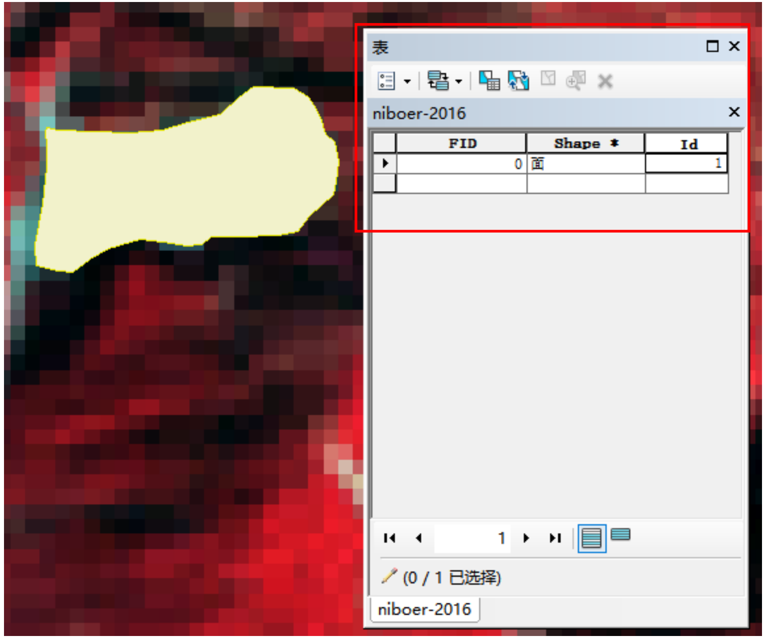


1. Identify the landslide areas (distinguished by their gray appearance, which contrasts with surrounding vegetation, and by their characteristic shape).

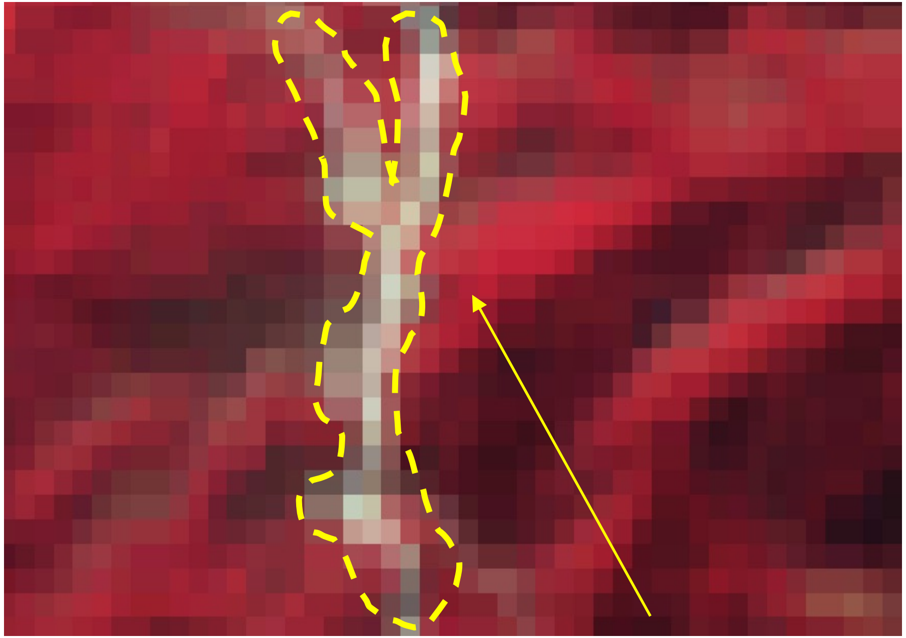
2. Label the landslide areas.

3. Landslide Sample Production

3.5 Reviewing Labeled Data and Identifying Interferences



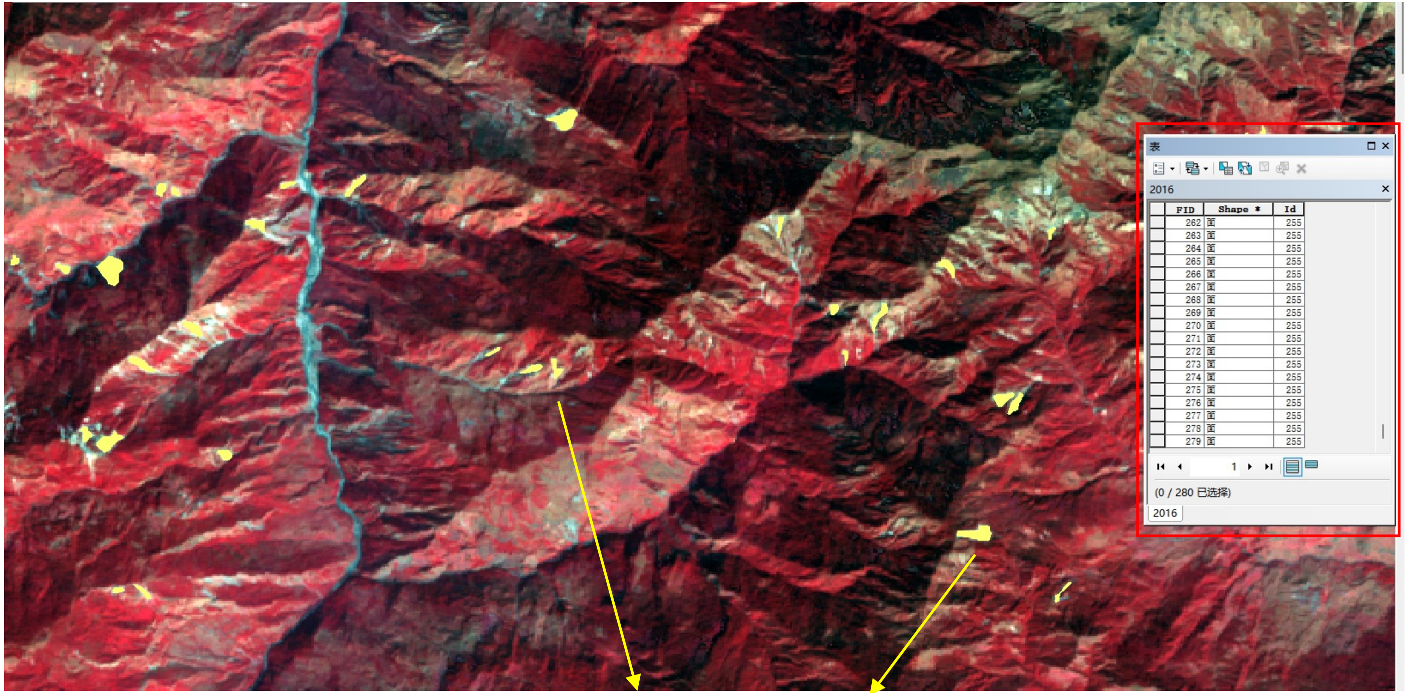
1. Open the shapefile’s attribute table. the labeled landslides are now stored as records within it.



2. When labeling the landslide file, be mindful of interfering features, for example, **roads** in the image.

3. Landslide Sample Production

3.6 Display of Completed Landslide Labeling



The figure displays the locations of extensive labeling, and the total number of labels can be viewed in the attribute table on the right.

3. Landslide Sample Production

3.7 Randomly generate sample points

Main Process of Landslide Extraction Based on Random Forest

Randomly generate sample points

↓

Visualize random sample points

↓

Generate features

↓

Train random forest model

↓

Model evaluation

3. Landslide Sample Production

3.7 Randomly generate sample points

1. Import Packages

```
import pandas as pd
import geopandas as gpd
import random
from shapely.geometry import Point
import rasterio
```

pandas: Provides data structures and operations for manipulating data in tabular form (DataFrames).
geopandas: Extends pandas to allow spatial data handling (e.g., shapefiles).
random: Used for generating random numbers (e.g., for selecting random background points).
shapely.geometry: Provides geometric objects (e.g., Point, Polygon) and geometric operations.
rasterio: Used to read/write geospatial raster data (e.g., TIFF images).

3. Landslide Sample Production

3.7 Randomly generate sample points

2. Prepare images and landslide annotation files for Nepal in 2016

2016.cpg	2024/6/17 17:41	CPG 文件	1 KB
2016.dbf	2024/6/17 17:41	DBF 文件	2 KB
2016.prj	2024/6/11 21:04	PRJ 文件	1 KB
2016.sbn	2024/6/17 17:41	SBN 文件	3 KB
2016.sbx	2024/6/17 17:41	SBX 文件	1 KB
2016.shp	2024/6/17 17:41	SHP 文件	74 KB
2016.shx	2024/6/17 17:41	SHX 文件	3 KB

scale_along_median_2014nepal2.tif	2024/6/7 15:03	TIF 文件	254,963 KB
scale_along_median_2014nepal2.tif.ovr	2024/6/7 15:03	OVR 文件	72,974 KB
scale_along_median_2015nepal2.tif	2024/6/7 15:01	TIF 文件	256,148 KB
scale_along_median_2015nepal2.tif.ovr	2024/6/7 15:01	OVR 文件	73,188 KB
scale_along_median_2016nepal2.tif	2024/6/7 15:02	TIF 文件	255,304 KB
scale_along_median_2016nepal2.tif.ovr	2024/6/7 15:02	OVR 文件	73,572 KB

3. Load all the landslide points and generate background points randomly, with landslide to background point ratio set at 1:5

```
# Parameters
use_all_landslide_points = True  # True means selecting all landslide points; False means selecting a subset
num_landslide_points = 100  # Effective when 'use_all_landslide_points' is set to False
background_multiplier = 20  # Background points are generated at a multiple of landslide points
```

3. Landslide Sample Production

3.7 Randomly generate sample points

4. Read images and shapefile for landslide annotations

```
# File paths for input TIFF and shapefile
tiff_path = "image_path.tif"
shp_path = "image_label.shp"

# Output CSV file path (stores geographic location and pixel row/column numbers)
csv_path = 'point.csv'
```

```
# Load the TIFF file using rasterio
with rasterio.open(tiff_path) as dataset:
    tiff_array = dataset.read(1)  # Read the first band (assumed to be grayscale)
    transform = dataset.transform  # Affine transformation matrix (maps pixel to geographic coordinates)
    width = dataset.width  # Image width in pixels
    height = dataset.height  # Image height in pixels

# Load the shapefile using geopandas
shapefile = gpd.read_file(shp_path)
```

Used to read/write geospatial TIFF images).

3. Landslide Sample Production

3.7 Randomly generate sample points

5. Check the number of landslide points in the shapefile

```
# Check the number of landslide points in the shapefile
total_landslide_points = len(shapefile)
print(f"Total landslide points in shapefile: {total_landslide_points}")
```

len() is used to get the number of items in an object

6. Select the number of landslide points based on the settings

```
# Check if we are using all landslide points or sampling a subset
if use_all_landslide_points or num_landslide_points >= total_landslide_points:
    sampled_landslide_points = shapefile  # Use all points
    print(f"Using all landslide points: {len(sampled_landslide_points)}")
else:
    # Randomly sample a subset of landslide points
    sampled_landslide_points = shapefile.sample(n=num_landslide_points, random_state=1)
    print(f"Using sampled landslide points: {len(sampled_landslide_points)}")
    print("Sampled landslide points coordinates:")
    print(sampled_landslide_points[['geometry']])
```

Return a random sample of items from an object

Control the seed of the random number generator, ensuring that the random operations yield reproducible results

3. Landslide Sample Production

3.7 Randomly generate sample points

7. Perform coordinate system conversion and geometry processing

```
# Reproject the landslide points to a projected CRS (coordinate reference system)
projected_crs = "EPSG:32644"  # UTM zone 44N (Universal Transverse Mercator)
sampled_landslide_points = sampled_landslide_points.to_crs(projected_crs)

# If the landslide points are polygons, convert them to centroids
if sampled_landslide_points.geometry.iloc[0].geom_type == 'Polygon':
    sampled_landslide_points = sampled_landslide_points.copy()
    sampled_landslide_points['geometry'] = sampled_landslide_points['geometry'].centroid

# Convert the points back to the original geographic CRS
sampled_landslide_points = sampled_landslide_points.to_crs(shapefile.crs)
```

Check if points are polygons

Convert polygons to centroids (center points)

3. Landslide Sample Production

3.7 Randomly generate sample points

8. Add pixel coordinates of landslide points

```
# Add pixel_x and pixel_y columns based on the landslide points' coordinates
sampled_landslide_points['pixel_x'] = sampled_landslide_points['geometry'].apply(
    lambda geom: int((geom.x - transform[2]) / transform[0]) if geom is not None else None)
sampled_landslide_points['pixel_y'] = sampled_landslide_points['geometry'].apply(
    lambda geom: int((geom.y - transform[5]) / transform[4]) if geom is not None else None)
```

Create a new column **pixel_x** in the GeoDataFram to store the x-coordinate

- **geom.x** and **geom.y**: Geographic coordinates (longitude and latitude) of the landslide point.
- **transform[2]:** Geographic x-offset for the top-left corner.
- **transform[5]:** Geographic y-offset for the top-left corner.
- **transform[0]:** Pixel width in geographic units.
- **transform[4]:** Pixel height in geographic units.

- **(geom.x - transform[2]) / transform[0] :** converts the geographic x-coordinate to a pixel x-coordinate
- **(geom.y - transform[5]) / transform[4]:** converts the geographic y-coordinate to a pixel y-coordinate.

3. Landslide Sample Production

3.7 Randomly generate sample points

9. Randomly generate background points

```
# Function to generate random background points (not in landslide areas)
1 usage
def generate_background_points(num_points, width, height, landslide_points, transform):
    points = []
    while len(points) < num_points:
        x = random.randint( a: 0, width - 1)  # Random x-coordinate
        y = random.randint( a: 0, height - 1)  # Random y-coordinate
        coord = pixel2coord(x, y, transform)  # Convert pixel to geographic coordinates
        point = Point(coord)
        if not landslide_points.geometry.contains(point).any():  # Ensure point is not in landslide area
            points.append((x, y, point))
    return points

# Function to convert pixel coordinates to geographic coordinates
1 usage
def pixel2coord(x, y, transform):
    """Convert pixel coordinates to geographic coordinates."""
    px, py = rasterio.transform.xy(transform, y, x)  # Convert pixel to longitude/latitude
    return px, py
```

- num_points: Number of background points to generate.
- width: Width of the image, used to generate the x-coordinate.
- height: Height of the image, used to generate the y-coordinat.
- landslide_points: Landslide data used to ensure no overlap between landslide and background points.
- transform: Affine transformation matrix used to convert pixel coordinates to geographic coordinates.

3. Landslide Sample Production

3.7 Randomly generate sample points

10. Create GeoDataFrame for background points and add labels

```
# Generate background points
num_background_points = len(sampled_landslide_points) * background_multiplier
background_points = generate_background_points(num_background_points, width, height, sampled_landslide_points, transform)

# Create a GeoDataFrame for background points
background_coords = [(pt[2].x, pt[2].y) for pt in background_points]
background_pixel_coords = [(pt[0], pt[1]) for pt in background_points]
background_gdf = gpd.GeoDataFrame( data: {'geometry': [pt[2] for pt in background_points],
                                   'pixel_x': [pt[0] for pt in background_pixel_coords],
                                   'pixel_y': [pt[1] for pt in background_pixel_coords]}, crs=shapefile.crs)

# Assign labels to the points
sampled_landslide_points['label'] = 1  # Landslide points labeled as 1
background_gdf['label'] = 0  # Background points labeled as 0

# Concatenate landslide and background points into a single dataset
samples = pd.concat([sampled_landslide_points, background_gdf])
```

pt[0]: the pixel X coordinate of the point
pt[1]: the pixel Y coordinate of the point
pt[2]: the geographic coordinates of the point

3. Landslide Sample Production

3.7 Randomly generate sample points

11. Save the created sample points as a CSV file

```
# Save the points as a CSV file, including geometry and pixel coordinates
samples[['geometry', 'pixel_x', 'pixel_y', 'label']].to_csv(csv_path, index=False)

print(f"Saved points to {csv_path}")
```

- **samples:** A pandas **DataFrame** containing the merged data of landslide points and background points.
  - **geometry:** Contains the geographic coordinates (points) of the landslide and background points.
  - **pixel_x:** The x-coordinate in pixel space (corresponding to the raster image) for each point.
  - **pixel_y:** The y-coordinate in pixel space (corresponding to the raster image) for each point.
  - **label:** Indicates whether the point is a landslide point (1) or a background point (0).
- **.to_csv(csv_path, index=False)**: Saves the selected columns to a CSV file at the path specified by csv_path.
  - **csv_path**: The file path where the CSV will be saved.
  - **index=False:** Prevents pandas from writing row indices to the CSV file.

3. Landslide Sample Production

3.8 Visualize random sample points

1. Import packages

```
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
import rasterio
from shapely import wkt
```

- pandas, geopandas for data handling
- rasterio to read TIFF files
- shapely.wkt to convert WKT strings to geometry
- matplotlib.pyplot for visualization

2. Read images

```
tiff_path = "image_path.tif"

# Read the TIFF file using rasterio
with rasterio.open(tiff_path) as dataset:
    tiff_array = dataset.read(1)
    transform = dataset.transform
    width = dataset.width
    height = dataset.height
```

band

the affine transformation matrix

image width

image height

3. Landslide Sample Production

3.8 Visualize random sample points

3. Read CSV file and convert to GeoDataFrame

```
points_df = pd.read_csv(csv_path)
points_df['geometry'] = points_df['geometry'].apply(wkt.loads)
points_gdf = gpd.GeoDataFrame(points_df, geometry='geometry', crs="EPSG:4326")
```

CSV file read using pandas.read_csv()

Converted geometry column (WKT) to shapely objects

Created a GeoDataFrame for spatial data

- points_df['geometry'   ]: Column containing geometric data in WKT format from the CSV file.
- apply(wkt.loads): Converts WKT strings to geometrical objects (like Points or Polygons) using the shapely library.
- gpd.GeoDataFrame(points_df, geometry='geometry', crs="EPSG:4326"):

- **gpd.GeoDataFrame()**: Transforms the original DataFrame into a GeoDataFrame for spatial operations.
- **points_df**: The original DataFrame that contains the converted geometries.
- **geometry='geometry'**: Specifies that the geometrical data is stored in the 'geometry' column.
- **crs="EPSG:4326"**: Sets the coordinate system to WGS 84

3. Landslide Sample Production

3.8 Visualize random sample points

4. Create figure and display image

```
# Visualization: display TIFF image and plot points
fig, ax = plt.subplots(figsize=(10, 10))
ax.imshow(tiff_array, cmap='gray', extent=[transform[2], transform[2] + width * transform[0], transform[5] + height * transform[4], transform[5]])
```

Set plot size

Display the TIFF as a grayscale image

5. Visualize landslide points and background points

```
# Plot landslide and background points
points_gdf[points_gdf['label'] == 1].plot(ax=ax, color='red', marker='o', markersize=5, label='Landslide Points', linestyle='')
points_gdf[points_gdf['label'] == 0].plot(ax=ax, color='blue', marker='o', markersize=5, label='Background Points', linestyle='')
```

- **points_gdf.plot():** plots landslide and background points
- **color='red':** Red for landslide points
- **marker='o':** Circle-shaped markers
- **markersize=5:** Size of the points
- **linestyle='':** No lines between points

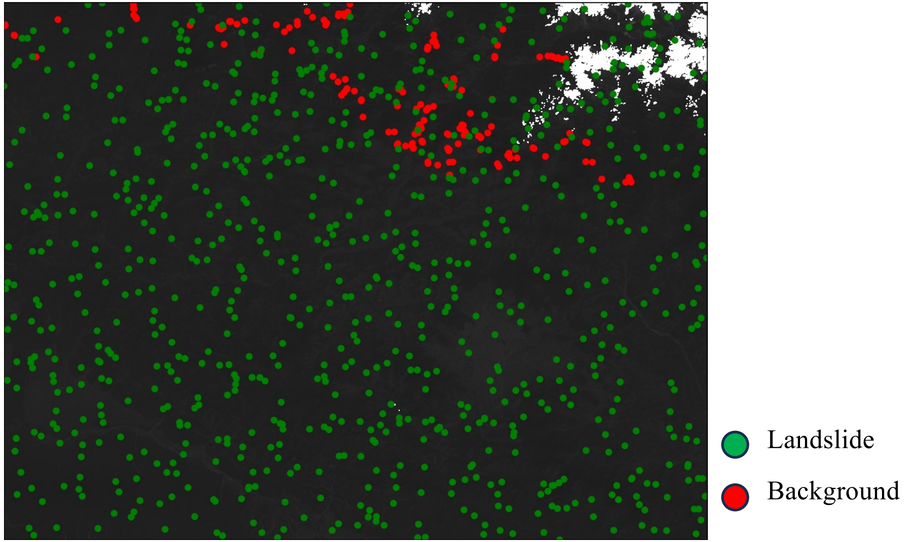
3. Landslide Sample Production

3.8 Visualize random sample points

6.Save and display the image

a resolution of 300 dots per inch

```
image_path = "point.png"
plt.savefig( *args: image_path, dpi=300)
plt.show()
```



Landslide to background ratio 1:5

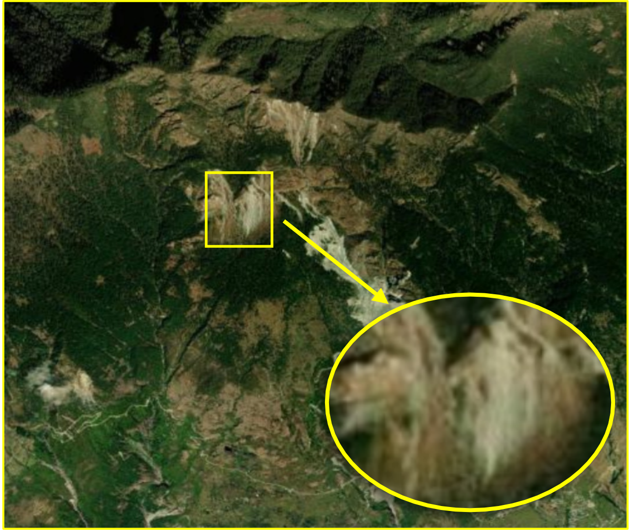
Outline

1. Model Invocation via OpenGMS Portal
2. Annual Image Composite
3. Landslide Sample Production
4. Random Forest Model Training
5. Random Forest Model Testing

4. Random Forest Model Training

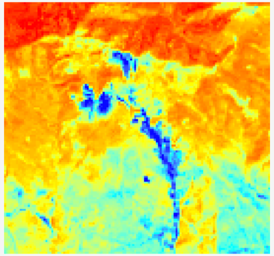
4.1 Feature Selection

Generate features

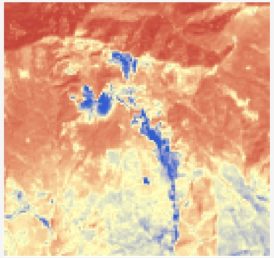


Landslide remote sensing image

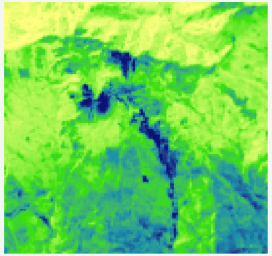
Feature selection



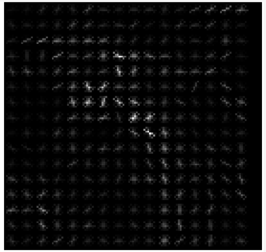
Blue band



Green band



Red band



HOG

4. Random Forest Model Training

4.1 Feature Selection

Generate features

1. Import packages

```
import numpy as np
import pandas as pd
from osgeo import gdal
from skimage.feature import graycomatrix, graycoprops, hog
```

- **numpy (np):** Handles multi-dimensional arrays and matrices.
- **pandas (pd):** Manages data structures like tables.
- **gdal (osgeo):** Reads geospatial raster data.
- **graycomatrix, graycoprops (skimage)**: Computes gray-level co-occurrence matrices (GLCM) and extracts texture features.
- **hog (skimage):** Calculates Histogram of Oriented Gradients (HOG) for shape features.

4. Random Forest Model Training

4.1 Feature Selection

Generate features

1. Import packages

```
import numpy as np
import pandas as pd
from osgeo import gdal
from skimage.feature import graycomatrix, graycoprops, hog
```

- **numpy (np):** Handles multi-dimensional arrays and matrices.
- **pandas (pd):** Manages data structures like tables.
- **gdal (osgeo):** Reads geospatial raster data.
- **graycomatrix, graycoprops (skimage)**: Computes gray-level co-occurrence matrices (GLCM) and extracts texture features.
- **hog (skimage):** Calculates Histogram of Oriented Gradients (HOG) for shape features.

4. Random Forest Model Training

4.1 Feature Selection

Generate features

2. Load image and sample points

```
# Read sample points file and output file path
sample_points_file = "point.csv"
output_file = "feature.csv"
# Read sample points file as DataFrame
sample_points = pd.read_csv(sample_points_file)

# Read the image data
image_file = "image_path.tif"

# Use GDAL to read the image data
dataset = gdal.Open(image_file)
bands_data = [dataset.GetRasterBand(i + 1).ReadAsArray() for i in range(dataset.RasterCount)]
bands_data = np.stack(bands_data, axis=-1)
# Get the number of rows, columns, and bands from the image
rows, cols, num_bands = bands_data.shape
```

Read sample points from a CSV file into a DataFrame

4. Random Forest Model Training

4.1 Feature Selection

Generate features

3. Calculate spectral features

```
for idx, row in sample_points.iterrows():
    x, y = int(row['pixel_x']), int(row['pixel_y'])
    label = 'landslide' if row['label'] == 1 else 'non-landslide'

    if 0 <= x < cols and 0 <= y < rows:
        spectrum_values = bands_data[y, x, :]

        mean_spectrum = np.mean(spectrum_values)
        std_spectrum = np.std(spectrum_values)
```

Pixel values across bands

Mean and standard deviation of spectral values

4. Random Forest Model Training

4.1 Feature Selection

Generate features

3. Calculate spectral features ——Indexs

```
def normalized_index(b1, b2):
    return (b1 - b2) / (b1 + b2)
```

```
    ndwi_year = normalized_index(spectrum_values_year[1], spectrum_values_year[3])
    ndvi_year = normalized_index(spectrum_values_year[3], spectrum_values_year[2])
```

Index	Calculation formula
NDVI	$NDVI = \frac{NIR - Red}{NIR + Red}$
NDWI	$NDWI = \frac{Green - NIR}{Green + NIR}$

4. Random Forest Model Training

4.1 Feature Selection

Generate features

4. Calculate texture features — Gray-Level Co-occurrence Matrix (GLCM)

```
def calculate_glcm_features(window, distances=[1], angles=[0]):
    if window.max() == window.min():
        return np.nan, np.nan, np.nan, np.nan, np.nan
    window = (255 * (window - window.min()) / (window.max() - window.min())).astype(np.uint8)
    glcm = graycomatrix(window, distances=distances, angles=angles, symmetric=True, normed=True)
    contrast = graycoprops(glcm, prop: 'contrast')[0, 0]
    dissimilarity = graycoprops(glcm, prop: 'dissimilarity')[0, 0]
    homogeneity = graycoprops(glcm, prop: 'homogeneity')[0, 0]
    energy = graycoprops(glcm, prop: 'energy')[0, 0]
    correlation = graycoprops(glcm, prop: 'correlation')[0, 0]
    return contrast, dissimilarity, homogeneity, energy, correlation
```

- window: 2D image patch
- distances: Pixel spacing, default [1]
- angles: Direction of pixel pairs, default [0]
- symmetric: Ensures matrix symmetry
- normed: Normalizes the GLCM to sum to 1

4. Random Forest Model Training

4.1 Feature Selection

Generate features

4. Calculate texture features — Gray-Level Co-occurrence Matrix (GLCM)

```
window_size_glcm = 3
half_window_glcm = window_size_glcm // 2
```

Size of the GLCM window

```
window_glcm = bands_data[y - half_window_glcm:y + half_window_glcm + 1,
                         x - half_window_glcm:x + half_window_glcm + 1, 0]

glcm_features = calculate_glcm_features(window_glcm)
```

Calculate GLCM features of the window

4. Random Forest Model Training

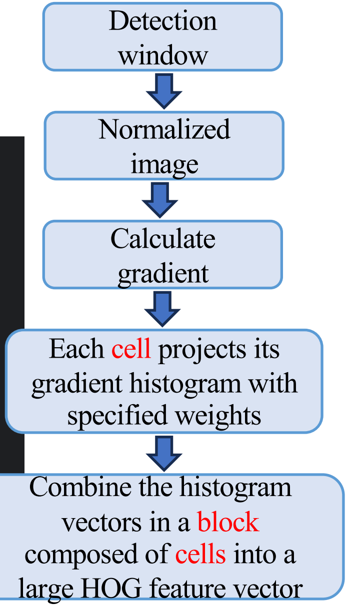
4.1 Feature Selection

Generate features

5. Calculate HOG features

```
def calculate_hog_features(window):
    hog_features = hog(window, orientations=8, pixels_per_cell=(8, 8),
                       cells_per_block=(1, 1), block_norm='L2-Hys',
                       feature_vector=True)
    return hog_features
```

- **orientations:** Number of gradient bins, default 8.
- **pixels_per_cell:** Cell size, default 8x8.
- **cells_per_block:** Block size, default 1x1.



Extract HOG features from a given image patch by dividing gradient into 8 directions, using 8x8 pixel cells, and L2-Hys normalization blocks to return a flattened feature vector to capture local shape and texture information

4. Random Forest Model Training

4.1 Feature Selection

Generate features

5. Calculate HOG features

```
    window_size_hog = 16
    half_window_hog = window_size_hog // 2

if (y - half_window_hog >= 0 and y + half_window_hog < rows and
        x - half_window_hog >= 0 and x + half_window_hog < cols):

    window_hog = bands_data[y - half_window_hog:y + half_window_hog + 1,
                            x - half_window_hog:x + half_window_hog + 1, 0]

    hog_features = calculate_hog_features(window_hog)
```

Size of the HOG window

Check if a 16x16 pixel window around sample points is within the image boundary

calculate and return the HOG features

4. Random Forest Model Training

4.1 Feature Selection

Generate features

6. Export features

```
# Add feature names as the first row
feature_names = [f'Band_{i}' for i in range(num_bands)] + \
                ['Mean_spectrum', 'Std_spectrum',
                 'Contrast', 'Dissimilarity', 'Homogeneity', 'Energy', 'Correlation'] + \
                ['HOG_feature_' + str(i) for i in range(len(hog_features))] + \
                ['Label']

df_results.columns = feature_names
df_results.to_csv(output_file, index=False, encoding='utf-8')  # Save the result
```

Convert the results list to a DataFrame and remove rows containing NaN values, and save the results as a CSV file

4. Random Forest Model Training

4.1 Feature Selection

Generate features

6. Export features

Band_0	Band_1	Band_2
9667	10639	10593

......

HOG_featu	HOG_featu	HOG_featu	HOG_featu	HOG_featu	HOG_featu	HOG_featu	HOG_featu	HOG_featu	HOG_featu
0.222575	0.532373	0.532373	0.532373	0.057658	0.007777	0.395456	0.324733	0.395456	0.17286

......

4. Random Forest Model Training

4.2 Random forest model construction

Train random forest model

1. Load dataset

- Read the feature CSV file based on the file path
- **x_train** ：Contains **all rows and all columns except the last one** in the CSV file as feature variables
- **y_train**：Contains **the last column** in the CSV file as the model classification label
- Convert the last column of the CSV file's data label into **1 or 0**

Read the feature data file path

```
df = pd.read_csv(data_file)
```

Use the iloc function to split the CSV file data

```
X_train= df.iloc[:, :-1]
y_train = df.iloc[:, -1]
```

```
def convert_to_numeric(value):
    if value == 'landslide':
        return 1
    elif value == 'non-landslide':
        return 0
    else:
        return value  # 如果有其他情况，保持原值不变
```

4. Random Forest Model Training

4.2 Random forest model construction

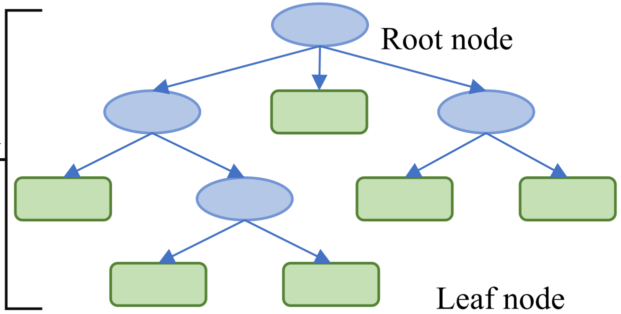
Train random forest model

2. Model construction

- criterion: The evaluation metric used for splitting decision tree nodes.
- max_depth: The maximum depth of the decision tree, prevents over-complexity
- max_features: The number of features considered at each split node, controls randomness and model diversity
- Number of estimators (n_estimators) : determines the computational time
- min_sample_split：The minimum number of samples required to split an internal node

```
param_grid = {
    'criterion': ['entropy', 'gini'],  #
    'max_depth': [5, 6, 7, 8],
    'max_features': [0.3, 0.4, 0.5],  #
    'min_samples_split': [4, 8, 12, 16],
    'n_estimators': [11, 13, 15],  # 决策
}
```

Decision Tree



4. Random Forest Model Training

4.2 Random forest model construction

Train random forest model

3. Model training

Create random forest classifier

```
rfc = ensemble.RandomForestClassifier()
rfc_cv = GridSearchCV(estimator=rfc, param_grid=param_grid, scoring='roc_auc', cv=4)
# 训练
rfc_cv.fit(X_train, y_train)

joblib.dump(rfc_cv, model_save_path)
```

Model training

Save the best model and its parameters

- **estimator=rfc**：Specify the base estimator for the model
- **param_grid=param_grid**：Model parameters
- **scoring='roc_auc'**：Scoring criterion for evaluating model performance
- **cv = 4** ：Number of times the model is validated

Outline

1. Model Invocation via OpenGMS Portal
2. Annual Image Composite
3. Landslide Sample Production
4. Random Forest Model Training
5. Random Forest Model Testing

5. Random Forest Model Testing

5.1 Model Testing

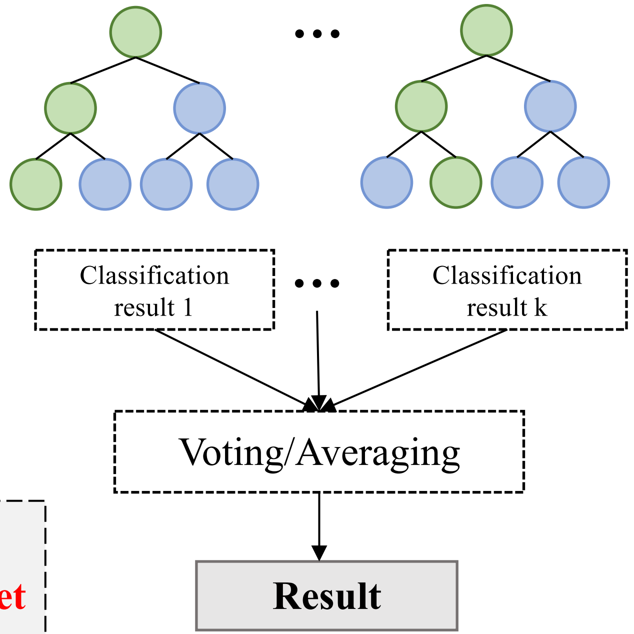
```python
# 加载模型
rf_classifier = joblib.load(model_load_path)

y_pred = rf_classifier.predict(X_test)

# 打印precision, recall, f1-score
report = classification_report(y_test, y_pred, output_dict=True)
#print(report)
print('\n准确率-Precision:', report['1']['precision'])
print('召回率-Recall:', report['1']['recall'])
print('F1-Score:', report['1']['f1-score'])
```

Import the trained parameters

Predict using the model



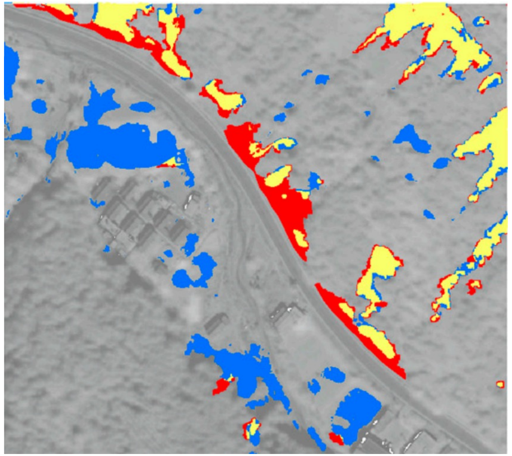
Model evaluation

Evaluate the performance of the random forest model using **the test set** by calculating metrics such as **accuracy**, **recall**, and **F1 score** to assess model performance

5. Random Forest Model Testing

5.2 Model Evaluation

Remote sensing image



Landslide extraction result map

Actual values	Extraction results	
	Positive	**Negative**
Positive	TP	FN
Negative	FP	TN

FN：An instance for which predicted value is negative but actual value is positive

FP：An instance for which predicted value is positive but actual value is negative

TP：An instance for which both predicted and actual values are positive

TN：An instance for which both predicted and actual values are negative

5. Random Forest Model Testing

5.2 Model Evaluation

Evaluation Metrics	Formula
P (Precision): The proportion of predicted positive samples among all predicted positive samples	$\mathrm{P} = \frac{TP}{\mathrm{TP} + FP}$
R (Recall): The proportion of predicted positive samples among all actual positive samples	$\mathrm{R} = \frac{TP}{\mathrm{TP} + FN}$
F-1: The weighted harmonic mean of precision (P) and recall (R), used to evaluate the performance of a classification model	$F_1 = \frac{2PR}{\mathrm{P} + R}$

5. Random Forest Model Testing

5.2 Model Evaluation

Precision evaluation statistics

Ground truth	Extraction results: Landslide	Extraction results: Background
Landslide	202	77
Background	69	5539

Evaluation Metrics	Result
Precison	74.54%
Recall	72.40%
F1	73.45%