INDIAN INSTITUTE OF TECHNOLOGY KANPUR
Department of Civil Engineering

CE672

MACHINE PROCESSING OF REMOTELY SENSED DATA

Different types of accuracy analysis procedures

Submitted by:

Name	Roll No.
Shreya Gupta	221027
Saloni Mittal	220942
Ashwani Kumar Singh	220224
Ayush Dixit	220262

Table of Contents

Table of Figures

Table of Tables

1. Abstract

This paper presents a comprehensive analysis of classification accuracy in remotely sensed images using statistical validation techniques. Accuracy assessment is essential for evaluating the reliability of classified maps, typically carried out through a confusion matrix. The study explores various accuracy measures including overall accuracy, user's and producer's accuracy, and extends to advanced indices such as the Kappa and Tau coefficients that adjust for chance agreement. Comparative methods using statistical tests and conditional metrics are also discussed. The paper highlights the strengths and limitations of each approach, emphasizing the importance of selecting appropriate accuracy indices for reliable map validation in remote sensing applications.

2. Introduction

2.1 Background

In the realm of remote sensing, the classification of satellite imagery plays a critical role in land cover mapping, environmental monitoring, and resource management. However, the reliability of these classifications hinges on a robust accuracy assessment framework. Over the decades, accuracy assessment has evolved from simple comparisons to sophisticated matrix-based evaluations that consider user needs and statistical validity. The confusion matrix, proposed as a standard early on by Congalton [1], remains central to this process. Story and Congalton [2] emphasized the importance of assessing classification from the end-user's perspective, which has become increasingly relevant as applications diversify.

2.2 Significance of the Study

A classification result is only as useful as its accuracy allows. Many studies rely heavily on overall accuracy, which, although informative, can mask significant class-wise misclassifications. Foody [3] has highlighted the importance of using complementary metrics such as the Kappa coefficient, user's accuracy, and producer's accuracy to gain a holistic understanding. This study builds on these ideas to perform a comparative evaluation of different accuracy metrics, aiming to help practitioners select appropriate tools for their specific use cases.

2.3 Objectives

The primary objectives of this study are:

- To understand and implement different accuracy assessment techniques for image classification.
- To compare the effectiveness of various accuracy metrics such as overall accuracy, user's and producer's accuracy, and Kappa coefficient.
- To explore how sampling techniques and ground truth data affect the outcomes of classification accuracy.

This objective framework is inspired by the standard methodologies outlined in Congalton and Green's comprehensive guide on the topic [4].

3. Literature Survey

3.1 Overview of Existing Work

The foundation for modern accuracy assessment in remote sensing was laid by **Congalton (1991)**, who formalized the use of error matrices for evaluating classification results [5]. This was further developed by **Foody (2002)**, who critically reviewed advancements in classification accuracy metrics and statistical validation techniques [6]. Later, **Olofsson et al. (2014)** proposed best practices for accuracy assessment, including stratified sampling, area estimation, and appropriate metric selection [7].

In parallel, land cover classification using satellite imagery has emerged as a crucial application area in remote sensing. The **Sentinel-2** mission, known for its high spatial resolution and rich spectral information, has become a widely adopted data source for monitoring land use and land cover dynamics. A variety of classification algorithms—both supervised and unsupervised—have been employed for this task, such as **Support Vector Machines (SVM)**, **Random Forests (RF)**, and **K-Nearest Neighbors (KNN)**.

Among recent contributions, **Zhang et al. (2020)** showcased the effectiveness of pixel-based classification using multispectral imagery for distinguishing land cover types like vegetation, water bodies, and built-up areas [8]. In addition, global datasets such as the **Copernicus Global Land Cover layers** [9] and **Google's Dynamic World** [10] have become valuable resources for open-access land classification, offering pre-processed and regularly updated land cover maps that support large-scale and regional analyses.

3.2 Accuracy Assessment Methods

3.2.1 Confusion Matrix Approach

The confusion matrix (error matrix) is the most widely used tool for assessing classification accuracy. It provides insights into user's and producer's accuracies, as well as overall performance [11][12]. Errors of omission and commission are directly extracted from this matrix, making it a practical evaluation framework.

3.2.2 Naïve Accuracy Measures

Naïve metrics, such as overall accuracy (Ao), user's accuracy (Ci), and producer's accuracy (Oj), provide basic insight into classification performance. These are calculated from diagonal and off-diagonal elements of the confusion matrix. While simple, they do not account for chance agreement [13].

3.2.3 Kappa Statistic

The Kappa coefficient (Cohen, 1960s) corrects overall accuracy for agreement occurring by chance. It is widely accepted in remote sensing for providing a more robust estimate of classification quality [14]. However, its reliance on pre-known marginal distributions limits its applicability in practical remote sensing scenarios.

3.2.4 Limitations of Kappa

Rosenfeld & Fitzpatrick-Lins (1986) and Ma & Redmond (1995) criticized the assumptions of Kappa, noting that in real-world applications, row (user) marginal probabilities are not known in advance, undermining its validity [15][16].

3.2.5 Tau Coefficient

The Tau coefficient was introduced to overcome limitations of Kappa. It assumes more realistic prior probabilities and allows comparison even with unequal class distributions [16][17]. Tau is interpretable and statistically testable, making it suitable for modern remote sensing tasks.

3.2.6 Comparative Studies

Recent works, including Rossiter (2001), emphasize the importance of using both Kappa and Tau, depending on data characteristics and classification context. Tau is recommended when class imbalance or unequal priors are involved [18].

3.3 Key Findings

- High-resolution imagery, such as that from Sentinel-2, provides enhanced capability for distinguishing fine-scale land features [7].
- Pixel-based classifiers (like KNN and Random Forests) are particularly effective when trained on pre-labeled rasters, such as those provided in your dataset [8].
- Spectral band combinations (e.g., NDVI, SWIR) improve separability of land cover classes, especially for vegetation and water bodies [9].

- Accuracy assessment using metrics such as overall accuracy and confusion matrices is essential for evaluating classifier performance [11].

3.4 Gaps in Literature

- **Temporal classification using multi-date imagery** is often underutilized. Your work focuses on a single date (April 2021), whereas time-series analysis could offer more dynamic insights.
- **Many studies overlook regional validation**, making it difficult to apply global models effectively in specific Indian contexts.
- **Though classical classifiers like KNN are simple and interpretable**, literature shows that deep learning models (e.g., CNNs, U-Nets) outperform them in accuracy and spatial coherence [19].
- **There's limited integration of remote sensing results with on-ground verification**, which could enhance the validity of classification outputs.

4. Theory

4.1 Introduction

In remote sensing, the classification of satellite images or aerial photos into thematic maps is followed by accuracy assessment. This ensures that the classified outputs accurately represent the ground reality. The most accepted methodology for this process is based on statistical evaluation using reference (ground-truth) data.

Accuracy assessment tools, such as the confusion matrix and Kappa coefficient, help quantify how well the classified data matches the reference data, providing insights into classification quality.

4.2 Confusion Matrix (Error Matrix)

4.2.1 Structure of the Confusion Matrix

The confusion matrix is an M × M table where:

- **Rows** represent the classified data.
- **Columns** represent the reference (true) data.
- **Diagonal elements (X_{ii})** represent correct classifications.
- **Off-diagonal elements (X_{ij}, $i \neq j$)** show misclassifications.

4.2.2 Errors in the Confusion Matrix

It summarizes two types of errors:

- **Error of Commission ($1 - C_i$)**: A class is wrongly assigned (false positive).
- **Error of Omission ($1 - O_i$)**: A class is missed where it should be assigned (false negative).

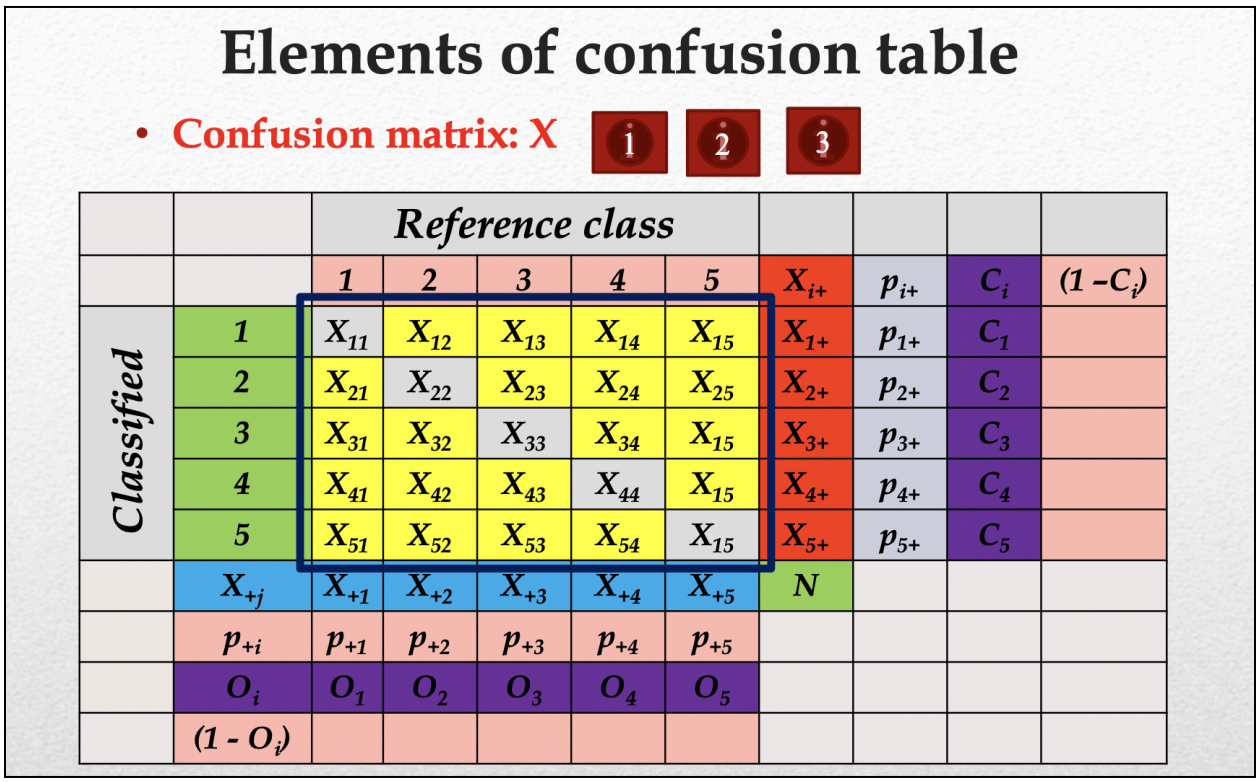

Figure 1: Elements of Confusion Table

4.3 Naïve Accuracy Measures

These are direct estimates from the confusion matrix without accounting for chance agreement.

4.3.1 Overall Accuracy (A_o or $\widehat{p}$)

The **overall accuracy** represents the proportion of all correctly classified samples out of the total number of samples. It is calculated from the confusion matrix as:

$$A_o = \frac{\sum X_{ii}}{N}$$

Where:

- X_{ii} are the diagonal elements of the confusion matrix representing correct classifications for class i,
- N is the total number of classified samples.

In the **naïve binomial approach** [20], the overall accuracy is expressed as:

$$\hat{p} = \frac{c}{N}, \quad \hat{q} = 1 - \hat{p}$$

Where:

- $c = \sum X_{ii}$, the number of correct predictions (successes),
- $\hat{p}$ is the estimate of classification accuracy,
- $\hat{q}$ is the estimate of classification inaccuracy.

This naïve method assumes all errors are equally likely and does not account for chance agreement [21].

4.3.2 User's Accuracy (C_i)

The **user's accuracy** (also known as reliability) quantifies the probability that a sample classified into a certain class on the map actually belongs to that class on the ground. It is calculated as [22]:

$$C_i = \frac{X_{ii}}{X_{i+}}$$

Where:

- X_{ii} is the number of correctly classified samples for class i,
- X_{i+} is the total number of samples classified as class i by the model (row total).

This helps identify **errors of commission**, where a class is **incorrectly assigned** to a sample [20].

4.3.3 Producer's Accuracy (O_j)

The **producer's accuracy** (also called completeness) represents the probability that a sample belonging to a class in the reference data has been correctly classified on the map. It is computed as [22]:

$$O_j = \frac{X_{jj}}{X_{+j}}$$

Where:

- X_{jj} is the number of correctly classified samples for reference class j,
- X_{+j} is the total number of reference samples belonging to class j (column total).

This identifies **errors of omission**, where a sample that **should be** classified into a class is **missed** [20].

4.3.4 Standard Errors and Confidence Intervals

To quantify the uncertainty in the estimated accuracy, the **standard error** of the naïve overall accuracy is calculated using a binomial model [21], [23]:

$$\hat{s} = \sqrt{\frac{\hat{p}(1-\hat{p})}{N}}$$

A **confidence interval** can then be constructed as [20], [24]:

$$\hat{p} \pm \left[Z_{1-\alpha} \cdot \hat{s} + \frac{1}{2N} \right]$$

Where:

- $Z_{1-\alpha}$ is the standard normal deviate corresponding to the confidence level (e.g., 1.96 for 95%),
- $\frac{1}{2N}$ is a finite sample correction factor.

These intervals provide a statistical range within which the true classification accuracy is expected to lie.

4.4 Kappa Coefficient (κ)

Kappa improves upon overall accuracy by adjusting for agreement that may happen by chance. It is defined as:

$$\kappa = \frac{p_o - p_c}{1 - p_c}$$

Where:

- p_o = observed agreement (same as overall accuracy A_o) [25],
- p_c = expected chance agreement, calculated as:

$$p_c = \sum (p_{i+} \cdot p_{+i})$$

Here, p_{i+} is the proportion of observations in the predicted class *i*, and p_{+i} is the proportion in the actual (reference) class *i*.

4.4.1 Interpretation of Kappa

The interpretation of Kappa values is generally as follows [26]:

- $\kappa < 0$: worse than chance agreement
- $\kappa = 0$: agreement is due to chance
- $0 < \kappa < 0.20$: slight agreement
- $0.21 \leq \kappa \leq 0.40$: fair agreement
- $0.41 \leq \kappa \leq 0.60$: moderate agreement
- $0.61 \leq \kappa \leq 0.80$: substantial agreement
- $0.81 \leq \kappa \leq 1.00$: almost perfect agreement

4.4.2 Variance and Confidence Interval for κ

The variance of Kappa can be estimated, and confidence intervals are constructed using Z-scores. The formula for the confidence interval of κ is:

$$\kappa \pm Z_{1-\alpha} \cdot \hat{s}_{\kappa}$$

Where:

- $Z_{1-\alpha}$ is the standard normal score for the desired confidence level,
- $\hat{s}_{\kappa}$ is the standard error of Kappa [25].

4.4.3 Conditional Kappa

Conditional Kappa allows the analysis of classification agreement for individual classes:

- **User's κ_i**: Measures agreement for individual map classes (rows)
- **Producer's κ_j**: Measures agreement for individual reference classes (columns) [27]

This is particularly useful when the classification accuracy varies significantly between different categories.

4.5 Tau Coefficient (τ)

The Tau coefficient (τ) was developed to address some limitations of Kappa, especially its reliance on the assumption that row and column marginal totals (class proportions) are known and reliable. Tau adjusts the observed agreement using prior probabilities of class membership, making it suitable when prior class distributions are known or assumed.

4.5.1 Formula for Tau Coefficient

The Tau coefficient is given by:

$$\tau = \frac{p_o - \theta}{1 - \theta}$$

Where:

- p_o = observed agreement (same as in Kappa),
- θ = chance agreement based on prior probabilities of class membership [28].

Advantages of Tau:

- **Interpretability**: Tau values are directly interpretable. For instance, τ=0.8τ=0.8 indicates the classification is 80% better than random chance.
- **Variance and Confidence**: Like κ, the variance of τ can also be estimated, and Z-tests can be applied to determine significance levels [29].

4.5.2 Conditional Tau

Tau also allows **conditional assessment**, similar to Conditional Kappa. It evaluates agreement for individual classes but requires prior knowledge or assumptions about class distributions.

However, a key limitation is that Tau assumes equal class priors unless otherwise specified, making it less flexible in situations with unequal class distributions [30].

4.6 Weighted Measures

In real-world applications, not all misclassifications are equally severe. To account for varying degrees of error, a **weight matrix** W is introduced, where:

- $w_i = 1$: indicates perfect agreement.
- $w_{ij} \in [0,1]$: represents the penalty for misclassifying class i as class j, based on the severity of error [28].

Using this matrix, **weighted accuracy** and **weighted Kappa** (κ_w) can be computed to reflect the relative seriousness of misclassifications.

Weighted Kappa is especially useful in ordinal classification tasks, where errors closer to the true class should be penalized less than those further away [29].

4.7 Comparison of Classifications

To statistically compare the results of **two classification maps**, the following **Z-statistic** can be used:

$$Z = \frac{\hat{p}_a - \hat{p}_b}{\sqrt{s_a^2 + s_b^2}}$$

- $\hat{p}_a$, $\hat{p}_b$ are the accuracies of the two classification maps.
- s^2_a, s^2_b are their respective variance estimates [30].

If Z exceeds the critical value from the **standard normal distribution** (e.g., 1.96 for 95% confidence), we can conclude that the two maps differ significantly.
This test can be extended to compare:

- **Overall accuracy**
- **Kappa coefficients**
- **User's/Producer's conditional accuracy metrics**

4.8 Summary

Metric	Purpose	Formula	Adjusts for Chance?
Overall Accuracy (A_o)	Total agreement	$\frac{\sum X_{ii}}{N}$	No
User's Accuracy (C_i)	Reliability of mapped class	$\frac{X_{ii}}{X_{i+}}$	No
Producer's Accuracy (O_j)	Reliability of ground class	$\frac{X_{jj}}{X_{+j}}$	No
Kappa (κ)	Agreement beyond chance	$\frac{p_o - p_e}{1 - p_e}$	Yes
Tau (τ)	Agreement accounting for priors	$\frac{p_o - \theta}{1 - \theta}$	Yes

Table 1: Assessment metrics used in land cover classification

5. Data Used and Study Area

5.1 Dataset Description

This study utilizes the **Dynamic World Version 1 (DW V1)** dataset, a global land use/land cover (LULC) classification product jointly developed by **Google** and the **World Resources Institute (WRI)**. DW V1 provides **near real-time, 10-meter resolution** LULC predictions based on **Sentinel-2 Level-1C imagery**, processed through a deep learning model trained on over 24,000 annotated points from around the world.

DW V1 offers a **per-pixel probability** distribution for nine land cover classes:

1. Water
2. Trees
3. Grass
4. Flooded Vegetation
5. Crops
6. Shrub & Scrub
7. Built Area
8. Bare Ground

9. Snow & Ice

The .tif Raster Dataset visualized in QGIS:

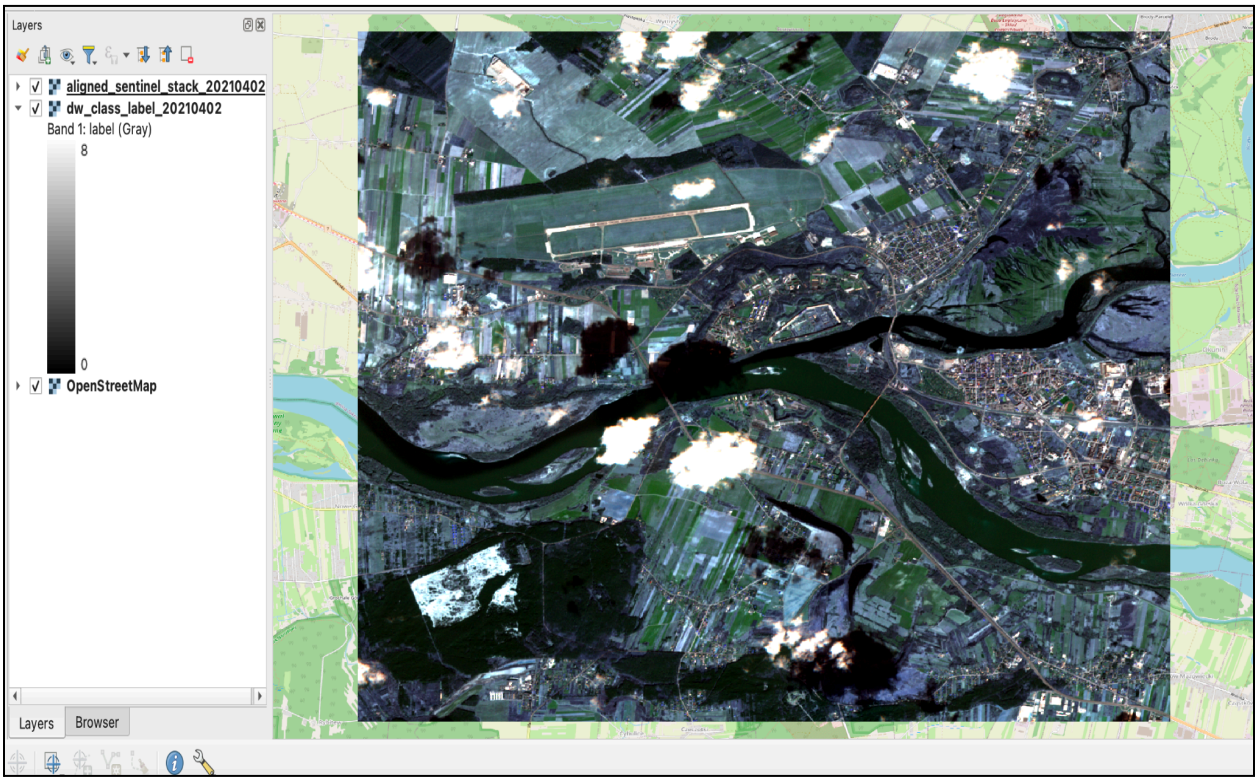

Figure 2: Satellite Raster on QGIS after rendering

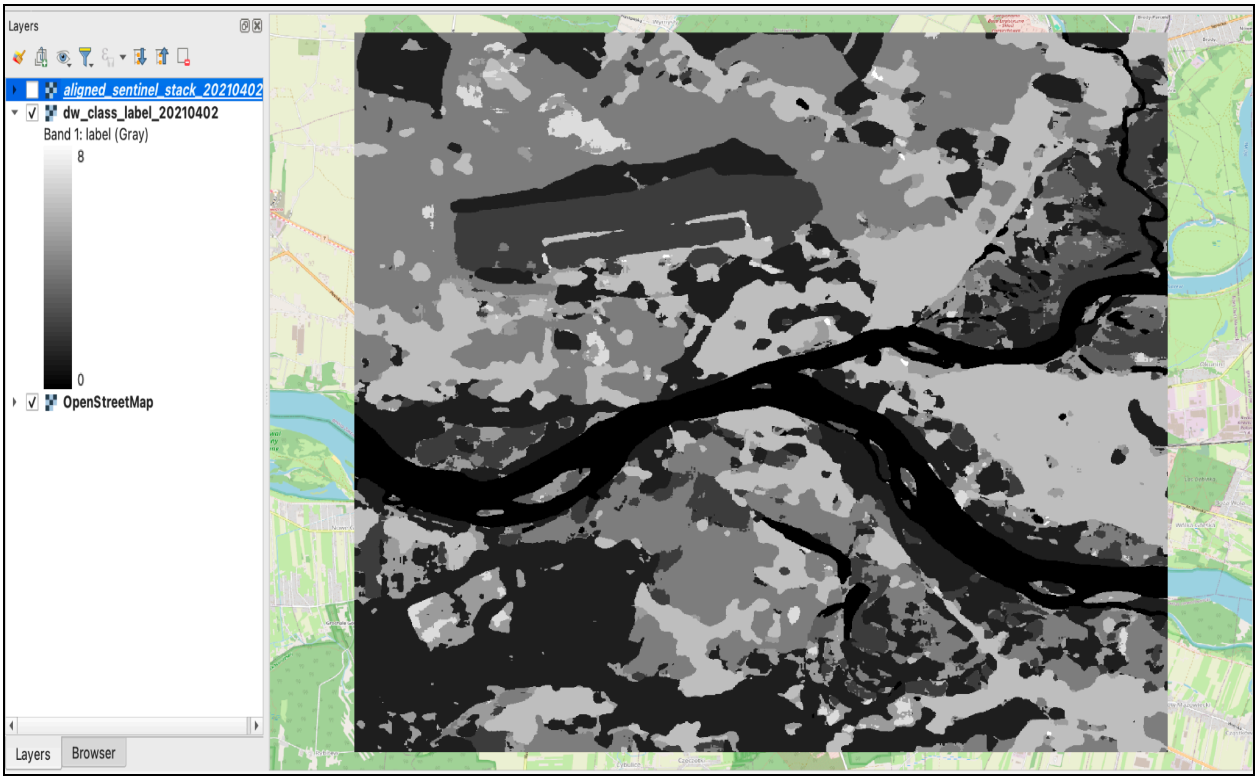

Figure 3: Class Labelled Raster on QGIS after rendering

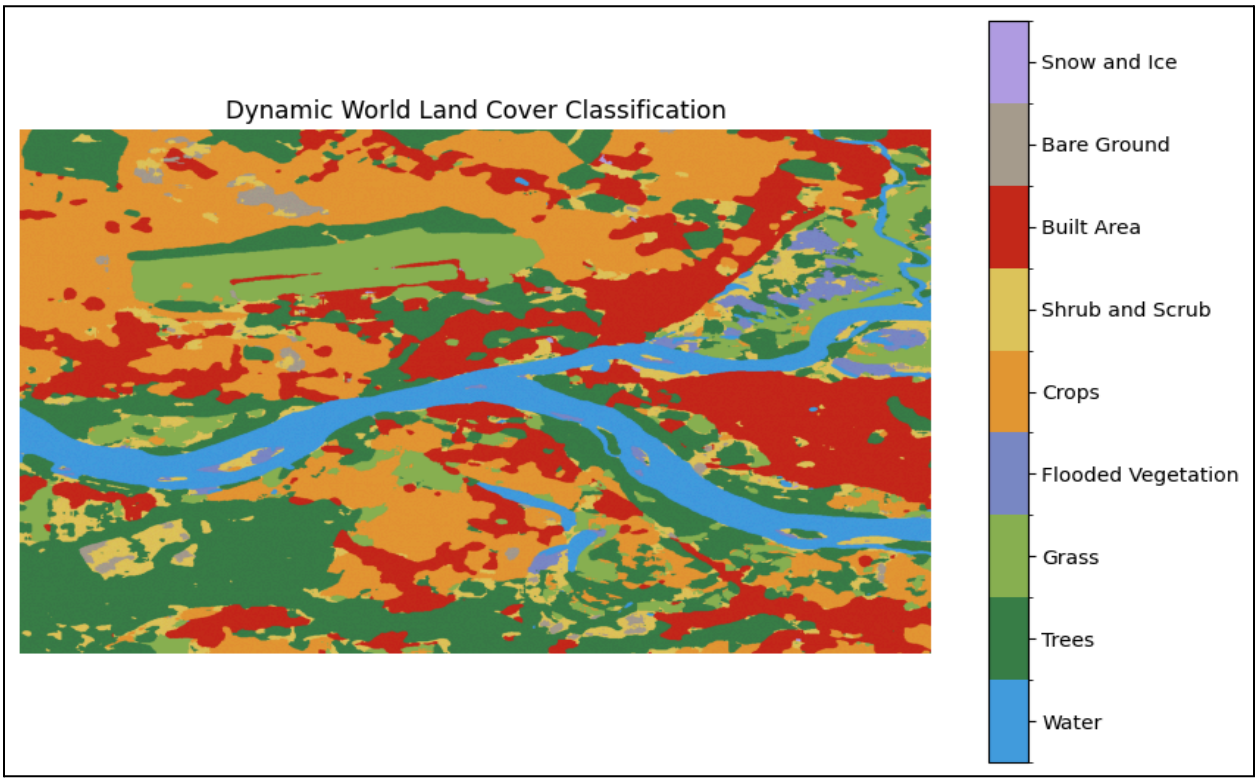

Figure 4: Dynamic World Land Cover Classification

These probabilities are accompanied by a discrete classification (`label`) band, selecting the class with the highest probability. The dataset spans from **June 27, 2015, to the present**, and is updated in **near real-time**, with a **2–5 day revisit time**, depending on latitude.

To ensure accuracy, **only Sentinel-2 images with < 35% cloud cover** are processed, and cloud/shadow areas are masked using an ensemble of cloud probability and spatial filtering algorithms, including the **Cloud Displacement Index (CDI)**and **Directional Distance Transform (DDT)**. The data is available as an Earth Engine ImageCollection, with each scene having a 9-band probability layer and a single-label classification band.

Reference: [31]

6. Methodology

The methodology adopted for land cover classification and accuracy analysis is implemented via a Gradio-based graphical user interface (GUI) named **"Land Cover Classifier & Accuracy Analyzer"**. This application is designed to classify Sentinel-2 satellite imagery using a Random Forest classifier and compute extensive accuracy metrics for assessment.

6.1 User Input via Gradio Interface

The first stage of the application is a **Gradio-powered input interface** that enables users to interactively load data and configure classification parameters.

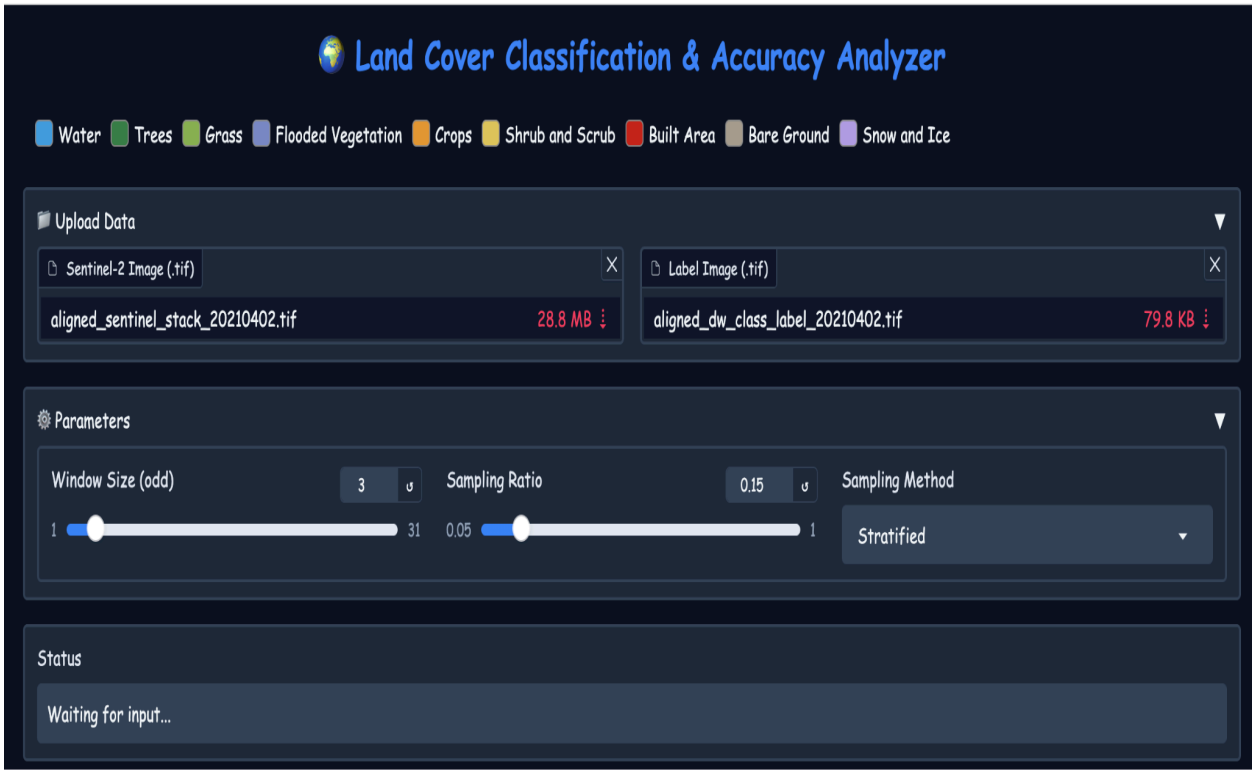

Figure 5: Web Interface

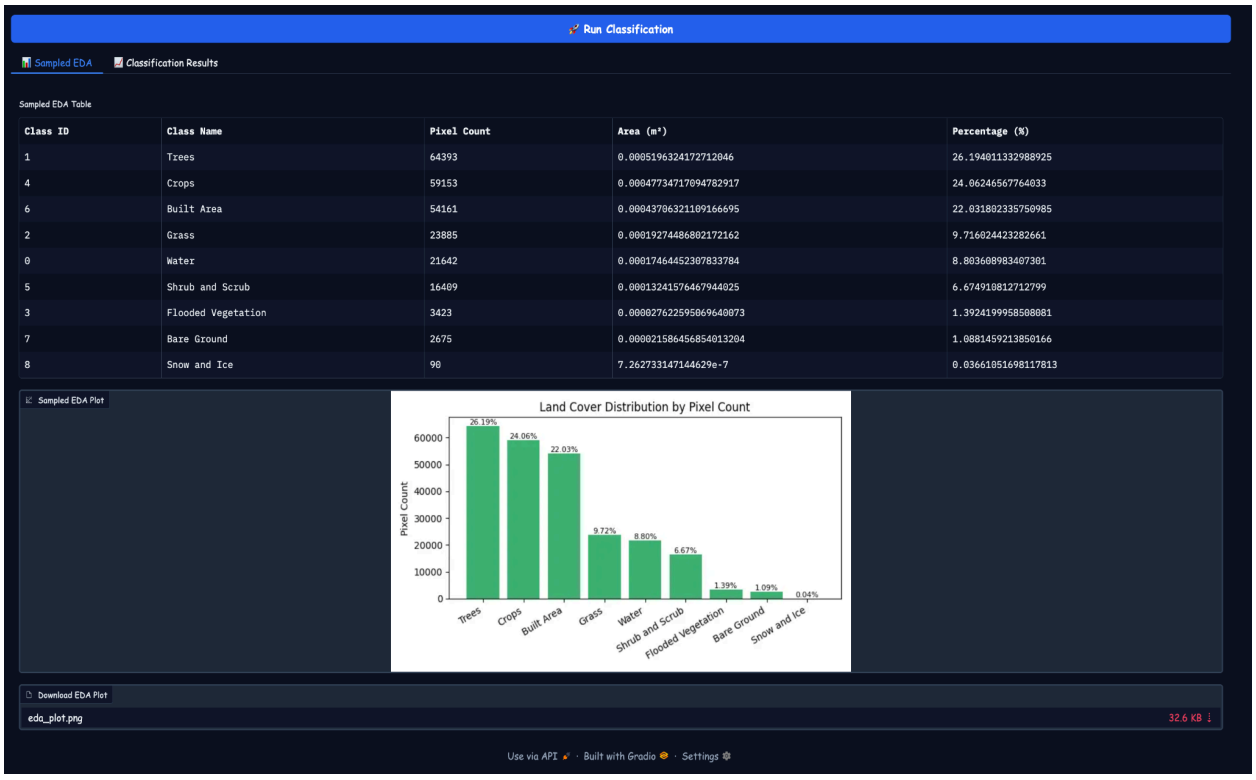

Figure 6: Sample EDA

Sampled EDA | Classification Results

Classification Report

Type	precision	recall	f1-score	support
Class 0	0.943	0.922	0.933	4328
Class 1	0.795	0.859	0.826	12879
Class 2	0.769	0.791	0.78	4777
Class 3	0.778	0.245	0.373	685
Class 4	0.77	0.8	0.785	11831
Class 5	0.532	0.386	0.448	3282
Class 6	0.735	0.748	0.742	10832
Class 7	0.76	0.243	0.368	535
Class 8	1	0.056	0.105	18
Accuracy	0.772	0.772	0.772	0.772
Macro Average	0.787	0.561	0.595	49167
Weighted Average	0.768	0.772	0.765	49167

Confusion Matrix

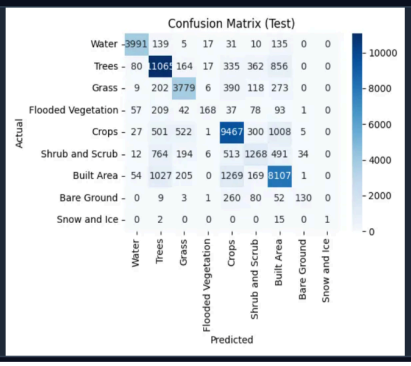

Figure 7: Classification Report and confusion matrix

	Water	Trees	Grass	Flooded Vegetation	Crops	Shrub and Scrub	Built Area	Bare Ground	Snow and Ice	Total	pi+	Ci	si	95% CI of Ci
Water	3991.000000	139.000000	5.000000	17.000000	31.000000	10.000000	135.000000	0.000000	0.000000	4328	0.088000	0.922100	0.004100	0.9142 - 0.9301
Trees	80.000000	11065.000000	164.000000	17.000000	335.000000	362.000000	856.000000	0.000000	0.000000	12879	0.261900	0.859200	0.003100	0.8531 - 0.8652
Grass	9.000000	202.000000	3779.000000	6.000000	390.000000	118.000000	273.000000	0.000000	0.000000	4777	0.097200	0.791100	0.005900	0.7796 - 0.8026
Flooded Vegetation	57.000000	209.000000	42.000000	168.000000	37.000000	78.000000	93.000000	1.000000	0.000000	685	0.013900	0.245300	0.016400	0.213 - 0.2775
Crops	27.000000	501.000000	522.000000	1.000000	9467.000000	300.000000	1008.000000	5.000000	0.000000	11831	0.240600	0.800200	0.003700	0.793 - 0.8074
Shrub and Scrub	12.000000	764.000000	194.000000	6.000000	513.000000	1268.000000	491.000000	34.000000	0.000000	3282	0.066800	0.386300	0.008500	0.3697 - 0.403
Built Area	54.000000	1027.000000	205.000000	0.000000	1269.000000	169.000000	8107.000000	1.000000	0.000000	10832	0.220300	0.748400	0.004200	0.7403 - 0.7566
Bare Ground	0.000000	9.000000	3.000000	1.000000	260.000000	80.000000	52.000000	130.000000	0.000000	535	0.010900	0.243000	0.018500	0.2066 - 0.2793
Snow and Ice	0.000000	2.000000	0.000000	0.000000	0.000000	0.000000	15.000000	0.000000	1.000000	18	0.000400	0.055600	0.054000	-0.0503 - 0.1614
Total	4230.000000	13918.000000	4914.000000	216.000000	12302.000000	2385.000000	11030.000000	171.000000	1.000000	49167	1.000000			
Producer's reliability (Oj)	0.943500	0.795000	0.769000	0.777800	0.769500	0.531700	0.735000	0.760200	1.000000					
si (Prod)	0.003600	0.003400	0.006000	0.028300	0.003800	0.010200	0.004200	0.032600	0.000000					
95% CI of Oj	0.9365 - 0.9505	0.7883 - 0.8017	0.7572 - 0.7808	0.7223 - 0.8332	0.7621 - 0.777	0.5116 - 0.5517	0.7268 - 0.7432	0.6962 - 0.8242	1.0 - 1.0					

Overall Accuracy Metrics

Metric	Value
Overall Accuracy	0.7724
Kappa	0.7139
Sigma Kappa	0.0024
Tau	0.7139
Z-Statistic	300.3717

Download Accuracy Report (.xlsx)

Confusion_Matrix_20250415_194510.xlsx 6.7 KB

Use via API · Built with Gradio · Settings

Figure 8: Confusion Matrix and Accuracy Metrics

6.1.1 File Uploads

Two Raster images are uploaded

- **Sentinel-2 Image (.tif)**
  A multiband GeoTIFF file containing spectral bands captured by the Sentinel-2 satellite. This serves as the primary data source for classification.
- **Label Image (.tif)**
  A single-band raster where each pixel has a class label (e.g., Water, Trees, Grass, Urban). This acts as the ground truth against which model performance is measured. The labels are expected to be encoded as integers, and pixels with label value '0' or NaN are considered invalid or unclassified.

6.1.2 Parameter Configuration

Users can set the following parameters directly through the GUI:

- **Window Size (int):**
  Determines the size of the spatial patch (e.g., 3, 5, 7) used to extract contextual features around each pixel. Larger windows capture more spatial detail but may smooth out local variations.
- **Sampling Ratio (float, between 0 and 1):**
  Specifies what fraction of the valid labeled pixels should be sampled for training and testing. For example, a sampling ratio of 0.6 means that 60% of the total valid labeled pixels will be used.
- **Sampling Method (str, 'random' or 'stratified'):**
  - Random Sampling: Uniform sampling from the set of valid pixels, without regard to class distribution.
  - Stratified Sampling: Sampling is proportional to the number of pixels in each class, helping to maintain class balance in the sample.

These parameters allow the user to explore different settings and observe their effect on classification performance, encouraging a more **exploratory, experiment-driven workflow**.

6.1.3 Real-Time Interaction

Gradio supports real-time execution: once the user provides all inputs and clicks the **"Run Classification"** button, the backend pipeline is triggered. Intermediate steps (like EDA plots and preview thumbnails) are displayed instantly, making the process **transparent and interactive**.

6.2 Data Preprocessing

Once the user provides the required inputs, the application performs a series of preprocessing steps to prepare the data for classification. This stage is crucial for ensuring that the input data is clean, properly aligned, and in a suitable format for the machine learning model.

6.2.1 Image and Label Loading

- Both the Sentinel-2 image and the label image are read using rasterio.
- The Sentinel-2 image is assumed to be a multiband image with shape (bands, height, width), which is transposed to (height, width, bands) for ease of pixel-wise operations.
- The label image is loaded as a 2D array of shape (height, width), containing integer class labels.
  Note: The two images must be spatially aligned (same resolution and georeference). Misalignment will lead to inaccurate feature-label pairing.

6.2.2. Valid Pixel Masking

- A valid mask is created by identifying pixels in the label image that are not equal to 0 and not NaN.
- This ensures that only labeled pixels are considered for sampling and model training.
  `valid_mask = (label_img != 0) &(~np.isnan(label_img))`

  This mask is applied to both the Sentinel-2 image and the label image to extract only the meaningful pixel data.

6.2.3. Windowed Feature Extraction

- For each valid pixel, a window of neighboring pixels is extracted from each band of the image.
- For example, with a window size of 3, a 3×3 patch around each pixel is used, and this patch is flattened per band and concatenated across all bands.
- The final feature vector per pixel has the shape:
  feature_dim = (window_size * window_size) * num_bands
  This step adds spatial context to the spectral features, enhancing the classifier's ability to distinguish between land cover types with similar spectral signatures but different textures or structures.

6.2.4. Sampling (Training & Testing Sets)

Based on the selected **sampling ratio** and **method**, the following steps are performed:

- A set of valid pixel coordinates is collected.
- From these coordinates:
  - **Random Sampling:** A random subset is chosen.
  - **Stratified Sampling:** Pixels are grouped by class, and sampling is done within each group according to the specified ratio
- The sampled pixel indices are then split into:
  - **Training Set:** Used to train the Random Forest classifier.
  - **Testing Set:** Used to evaluate classification performance.

This step ensures that the classification is **efficient**, even for large images, and that the training/testing sets are representative of the overall class distribution.

6.3 Exploratory Data Analysis (EDA)

Exploratory Data Analysis is a crucial part of understanding the characteristics of the input data before model training. In this GUI, EDA is performed **after sampling**, focusing on the pixels that will be used for classification.

6.3.1. Purpose of EDA

- To understand the distribution of classes in the sampled dataset.
- To analyze the spectral and spatial properties of each land cover class.

- To identify potential class imbalance, outliers, or noisy labels.

6.3.2. Class Distribution Histogram

- A bar plot shows the **number of sampled pixels per class**.
- This helps users assess whether the sampling was balanced, especially in the case of stratified sampling.
- If the class distribution is highly skewed, it may impact model performance.

Insight: *Class imbalance can lead to poor performance on minority classes unless handled properly (e.g., using class weights or more balanced sampling).*

6.4. Classification

The classification module is the core component of the GUI. It employs a supervised learning approach to assign land cover classes to each pixel in the input image, based on spectral and spatial information.

6.4.1. Model Used: Random Forest Classifier

- Why Random Forest?
  - Non-parametric and robust to noise.
  - Handles high-dimensional data well.
  - Performs well on small to medium-sized datasets.
  - Provides feature importance metrics.
- Implementation:
  - Scikit-learn's RandomForestClassifier is used.
  - We have used 100 decision trees, and set the random state at 42.

```
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)
```

6.4.2. Feature Representation: Window-Based Features

- Instead of using a single pixel's spectral values, a **square window (e.g., 3x3, 5x5)** is applied around each pixel.
- All pixel values within the window and across all bands are **flattened** into a single feature vector.
- This captures **contextual and spatial information**, improving classification accuracy in heterogeneous areas.

```
features = extract_window_features(image, window_size)
```

6.4.3. Training on Sampled Pixels

- The training mask is used to sample pixels and their corresponding features and labels.
- The model is trained on this sampled and split (80-20 stratified splitting) dataset.

```
X_train, y_train = sampled_features, sampled_labels
clf.fit(X_train, y_train)
```

6.4.4. Classification of Test Set Pixels

- Once trained, the classifier predicts labels for **all valid test pixels** in the image using the same window-based feature extraction method.

6.5 Accuracy Assessment

This module evaluates the performance of the classifier by comparing the predicted classification map with a reference ground truth map, using various accuracy metrics. The GUI allows the configuration of parameters such as accuracy type, sampling method, number of samples, and window size.

6.5.1 Sampling for Accuracy Assessment

Users can specify:
→ **Accuracy Type:** Stratified or Random.
→ **Window Size:** Neighborhood size around each sample.
→ **Number of Samples:** How many pixels to sample.
→ **Sample Selection Method:** E.g., per class (stratified) or total random.

Stratified Sampling code:

```
def stratified_sampling(mask, n_samples_per_class):
    coords = []
    for cls in np.unique(mask):
        indices = np.argwhere(mask == cls)
        selected = indices[np.random.choice(len(indices),
   n_samples_per_class, replace=False)]
        coords.extend(selected)
    return np.array(coords)
```

Random Sampling code:

```
def random_sampling(mask, n_samples):
    indices = np.argwhere(mask >= 0)
    selected = indices[np.random.choice(len(indices), n_samples,
   replace=False)]
    return np.array(selected)
```

6.5.2 Extraction of Predicted and Reference Labels

Once sample coordinates are selected, predicted and reference labels are extracted:

```
pred_values = classified_map[coords[:, 0], coords[:, 1]]
ref_values = reference_map[coords[:, 0], coords[:, 1]]
```

These arrays (pred_values, ref_values) are then used to compute all accuracy metrics.

6.5.3. Confusion Matrix Computation

The confusion matrix is computed using Scikit-learn's function:

```
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(ref_values, pred_values,
   labels=range(num_classes))
```

- Each row: Actual class (from reference map).
- Each column: Predicted class (from classification result).

6.5.4 Accuracy Metrics

The computed confusion matrix is used to calculate the following metrics:

- **Overall Accuracy:**
  ```
  overall_acc = np.trace(conf_matrix) / np.sum(conf_matrix)
  ```
- **User's Accuracy (Precision):**
  ```
  users_acc = np.diag(conf_matrix) / np.sum(conf_matrix, axis=0)
  ```
- **Producer's Accuracy (Recall):**
  ```
  producers_acc = np.diag(conf_matrix) / np.sum(conf_matrix, axis=1)
  ```
- **Kappa Coefficient:**
  ```
  kappa = cohen_kappa_score(ref_values, pred_values)
  ```
- **Tau Statistic:**
  ```
  tau = (overall_acc - expected_accuracy) / (1 - expected_accuracy)
  ```
- **Z-Statistic for Significance:**
  ```
  z = kappa / sigma_kappa
  ```
- **Sigma Kappa (Variance of Kappa):**
  ```
  sigma_kappa = np.sqrt((po * (1 - po) / (n * (1 - pe)**2)))
  # where po = observed accuracy, pe = expected accuracy by chance
  ```

6.5.5. Display and Export of Accuracy Results

Metrics are presented in a table using Pandas and styled with color gradients.

```
acc_table = pd.DataFrame({
    'Class': class_names,
    'User\'s Accuracy': users_acc,
    'Producer\'s Accuracy': producers_acc
})
acc_table.loc['Overall'] = ['Overall', overall_acc, '']
```

The confusion matrix is also converted into a styled data frame:

```
conf_df = pd.DataFrame(conf_matrix, index=class_names,
columns=class_names
conf_df.index.name = 'Reference'
conf_df.columns.name = 'Predicted'
```

Both the table and confusion matrix can be downloaded as an Excel file

```
with pd.ExcelWriter("accuracy_report.xlsx") as writer:
    acc_table.to_excel(writer, sheet_name="Accuracy Metrics",
   index=False)
    conf_df.to_excel(writer, sheet_name="Confusion Matrix"
```

6.6 GUI Design

The GUI is built using Gradio, providing an intuitive interface for land cover classification and accuracy assessment without needing to write code.

6.6.1 Key Components

- File Uploads: Users can upload Sentinel-2 images, label images, and class legend CSVs.
- Parameter Settings:
    - Window Size for feature extraction
    - Sampling Ratio and Sampling Method (Random or Stratified)
    - Accuracy Parameters: Accuracy type, number of samples, method
- EDA Section: Visualizes sampled class distributions and spectral stats.
- Classification Block: Runs Random Forest classification and displays the result using the legend colors.
- Accuracy Analysis: Computes confusion matrix, accuracy metrics (Overall Accuracy, Kappa, Tau, Z), and visual heatmaps.
- Export Section: Downloads Excel reports with all metrics and the confusion matrix.

6.6.2. Flow and Usability

The GUI follows a clear step-by-step layout:

1. Upload inputs
2. Set parameters
3. Run EDA
4. Classify
5. Assess accuracy
6. Download results

Each block is modular, with validations and tooltips to guide the user.

7. Results and Discussions

7.A Results of Preliminary Data Analysis

7.A.1 Initial Graphs

The Graph below shows the Area distribution by the Pixel Count and the Area covered by that class.

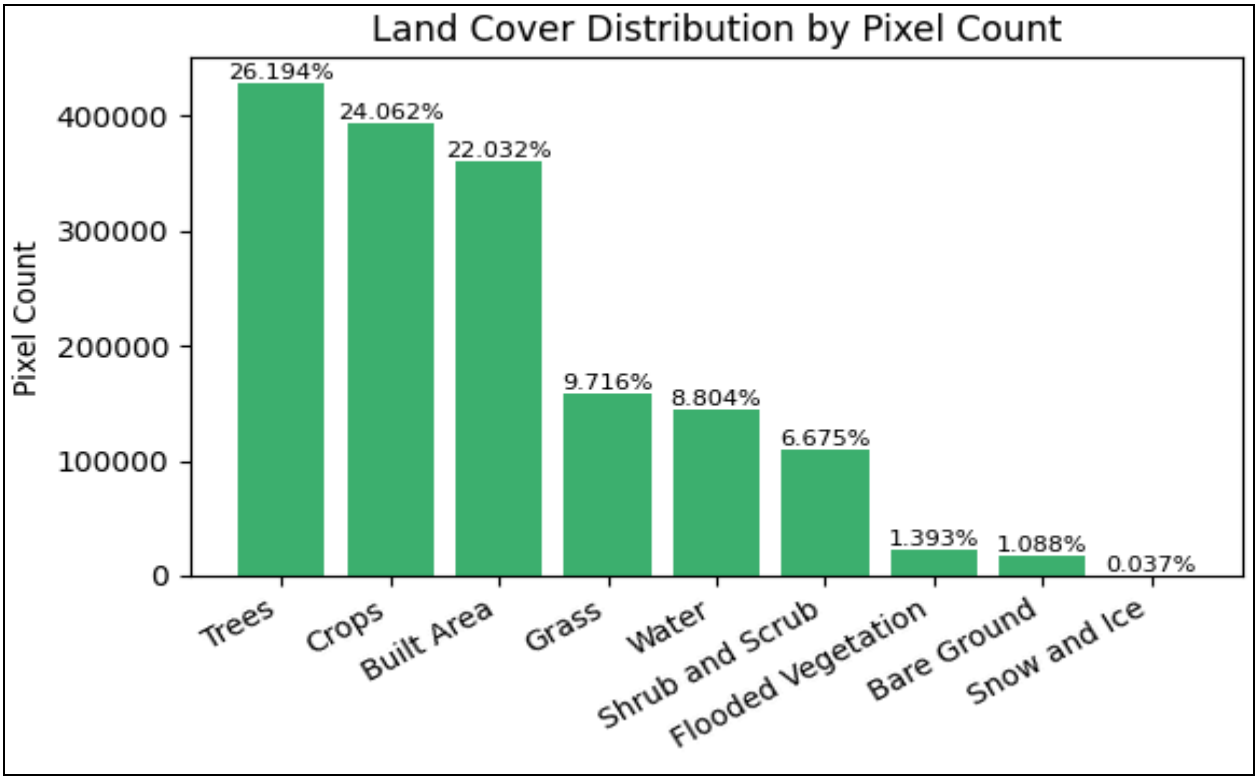

Figure 9: Land Cover Distribution by Pixel Count

Land Cover EDA Summary

	Class ID	Class Name	Pixel Count	Area (m²)	Percentage (%)
1	1	Trees	429292	0.003464	26.193879
0	4	Crops	394356	0.003182	24.062208
4	6	Built Area	361075	0.002914	22.031519
6	2	Grass	159235	0.001285	9.715956
5	0	Water	144286	0.001164	8.803821
3	5	Shrub and Scrub	109395	0.000883	6.674896
8	3	Flooded Vegetation	22826	0.000184	1.392762
2	7	Bare Ground	17835	0.000144	1.088229
7	8	Snow and Ice	602	0.000005	0.036732

Total valid pixels: 1,638,902
Total area: 0.013 m²

Figure 10: Land Cover EDA Summary

7.A.2 Bands with High correlation results

```
Highly correlated band pairs (corr > 0.94):
Band 1 vs Band 2 → Corr = 0.9822
Band 1 vs Band 3 → Corr = 0.9486
Band 2 vs Band 3 → Corr = 0.9761
Band 2 vs Band 5 → Corr = 0.9506
Band 3 vs Band 5 → Corr = 0.9593
Band 4 vs Band 6 → Corr = 0.9755
Band 4 vs Band 7 → Corr = 0.9801
Band 4 vs Band 8 → Corr = 0.9790
Band 6 vs Band 7 → Corr = 0.9950
Band 6 vs Band 8 → Corr = 0.9847
Band 7 vs Band 8 → Corr = 0.9934
Band 9 vs Band 10 → Corr = 0.9614
```

Figure 11: Highly Correlated Band Pairs

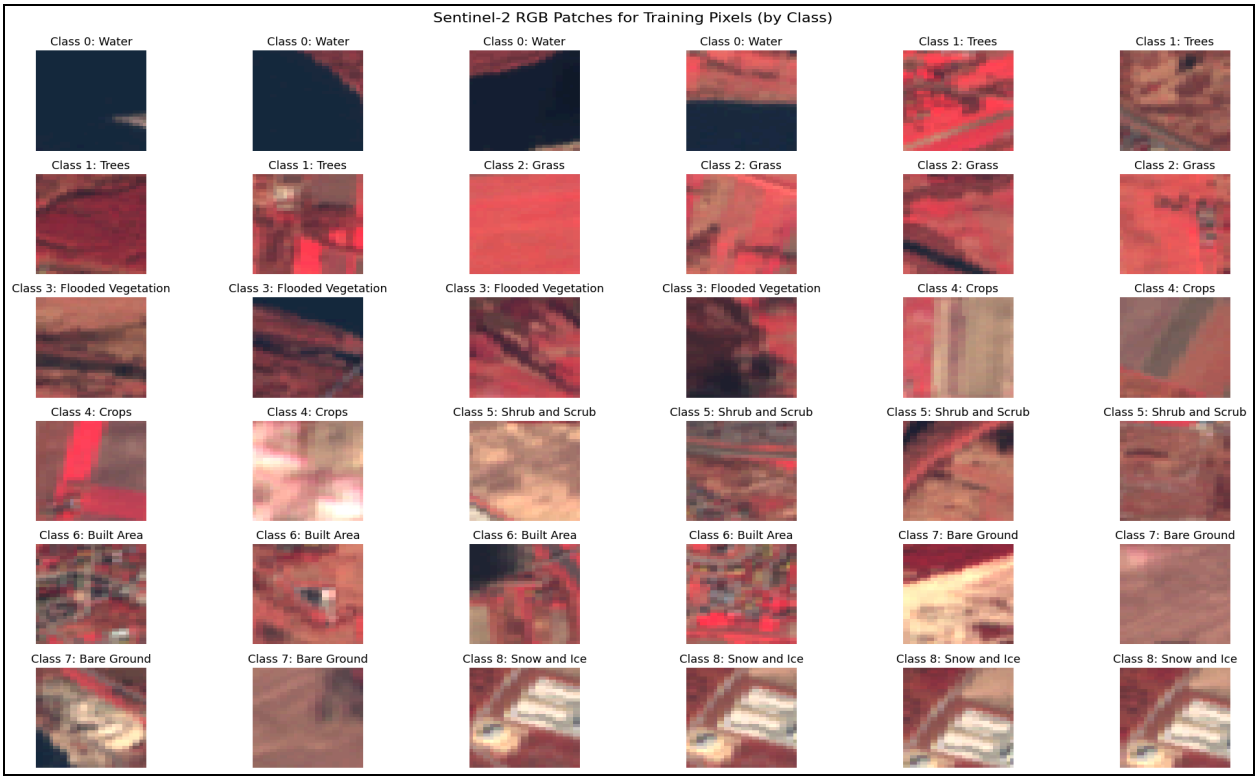

Figure 12: RGB Patches for Training Pixels

7.1 Experimental Setup

The land cover classification was performed using Sentinel-2 imagery and labeled data for the study area. The classification algorithm employed a Random Forest classifier, and performance was evaluated across varying parameters: **window sizes**, **sampling methods**, and **sampling ratios**.

7.2 Effect of Sampling Method

The model was evaluated using both **Stratified** and **Random** sampling. It was observed that:

- **Stratified sampling** consistently yielded higher classification accuracy. This is because it ensures balanced representation across all land cover classes.
- **Random sampling**, while faster, resulted in class imbalance, leading to reduced performance for underrepresented classes like 'Snow and Ice' or 'Flooded Vegetation'.
- The table below is for a Sampling ratio of **0.50** and a window size of **3x3.**

Sampling method	**Overall Accuracy**	**Kappa Score**
Stratified	0.8108	0.7623
Random	0.8102	0.7614

Table 2: Dependence on Sampling method

Significant changes in the accuracy could not be observed due to the initial imbalance of the no. of pixels for each class, as shown in EDA.

7.3 Influence of Window Size

Window size affects how much spatial context is included in each sample. The experiments used window sizes of 1, 3, 5, and 7 with the **sampling ratio constant at 0.2,** and **stratified** sampling was used. The findings are summarized as follows:

- A **moderate window size (5 or 7)** provided the best results, balancing context and local features.
- Larger windows (e.g., 21) began to introduce noise and redundant information, slightly reducing performance.
- Small windows (3) often lacked spatial patterns, leading to reduced classification accuracy.

Window Size	**Accuracy**
1x1	0.7909
3x3	0.7937
5x5	0.8160
7x7	0.8385

Table 3: Window Size vs Accuracy

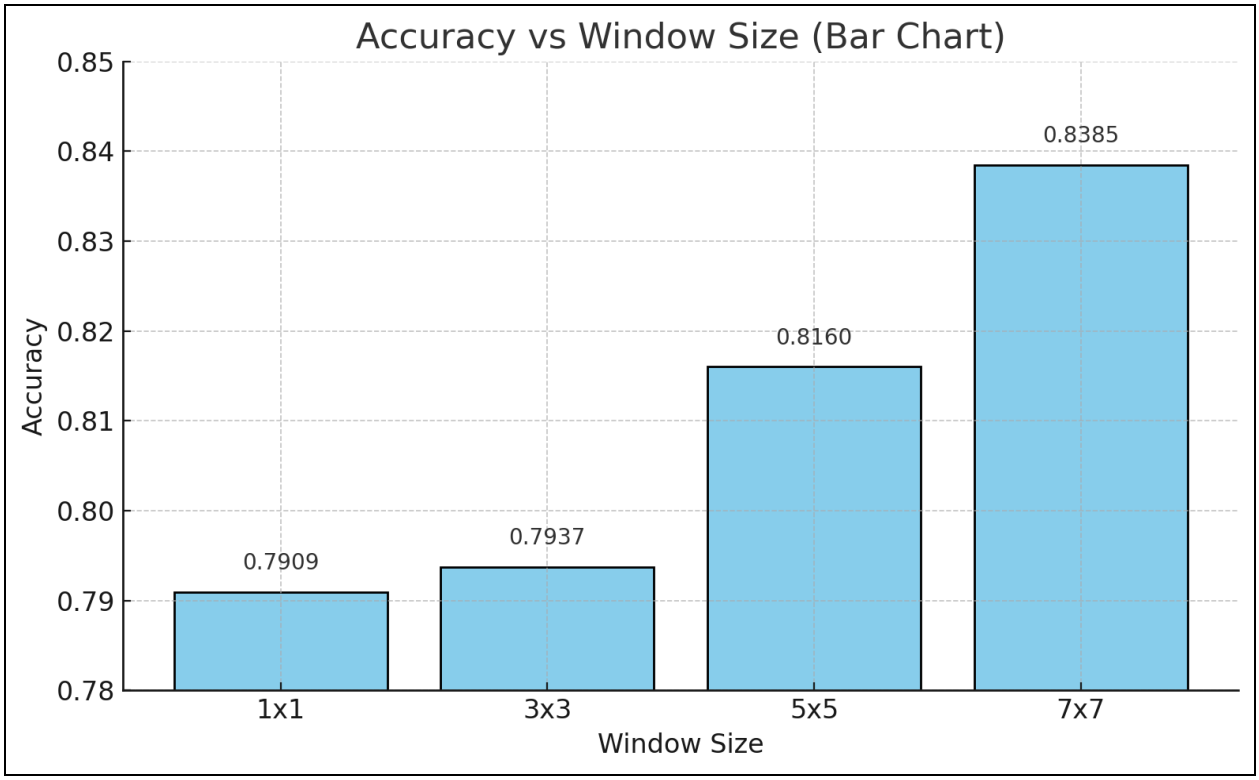

Figure 13: Accuracy Vs Window size (Bar Chart)

7.4 Sampling Ratio Impact

- The sampling ratio controls the proportion of labeled pixels used for training. Higher ratios provide more training data, which can improve model performance but also increase processing time.
- The results show that increasing the sampling ratio from 0.1 to 0.5 leads to a modest improvement in overall accuracy (from 0.7464 to 0.8246) for the window size 1X1. This trend is expected, as more training samples generally allow the classifier to learn more robust decision boundaries.

Ratio	**Accuracy**
0.05	0.7242
0.1	0.7464
0.2	0.7678
0.5	0.8246
1	0.9025

Table 4: Sample-Ratio Impact table

Graphical Representation:

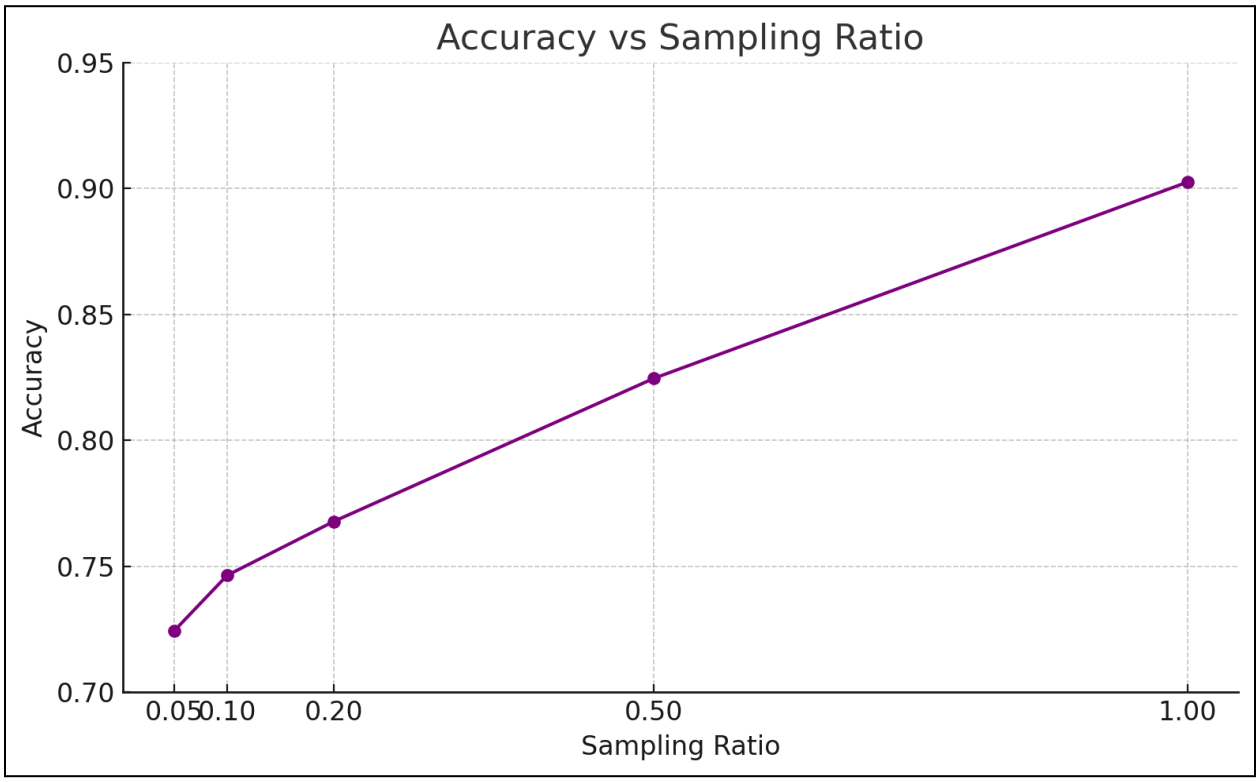

Figure 14: Line Chart: Accuracy vs Sampling Ratio

7.5 Class-wise Performance

From the classification report and confusion matrix:

- High accuracy was achieved for dominant classes such as Trees, Water, and Built Areas.
- Misclassifications commonly occurred between Grass and Crops and between Bare Ground and Built Area

Classification Report

Type	precision	recall	f1-score	support
Class 0	0.955	0.936	0.945	14405
Class 1	0.818	0.888	0.852	43062
Class 2	0.814	0.827	0.821	15914
Class 3	0.842	0.353	0.497	2299
Class 4	0.811	0.84	0.825	39354
Class 5	0.633	0.462	0.534	10904
Class 6	0.777	0.788	0.783	36107
Class 7	0.837	0.309	0.452	1790
Class 8	0.75	0.054	0.1	56
Accuracy	0.81	0.81	0.81	0.81
Macro Average	0.804	0.606	0.645	163891
Weighted Average	0.807	0.81	0.805	163891

Figure 15: Classification Report

Class	Precision	Recall	F1-Score
Trees	0.846	0.906	0.875
Water	0.959	0.963	0.961
Built Area	0.806	0.831	0.819
Flooded Vegetation	0.92	0.371	0.529
Shrub and Scrub	0.694	0.477	0.566

Table 5: Class-wise Performance Table

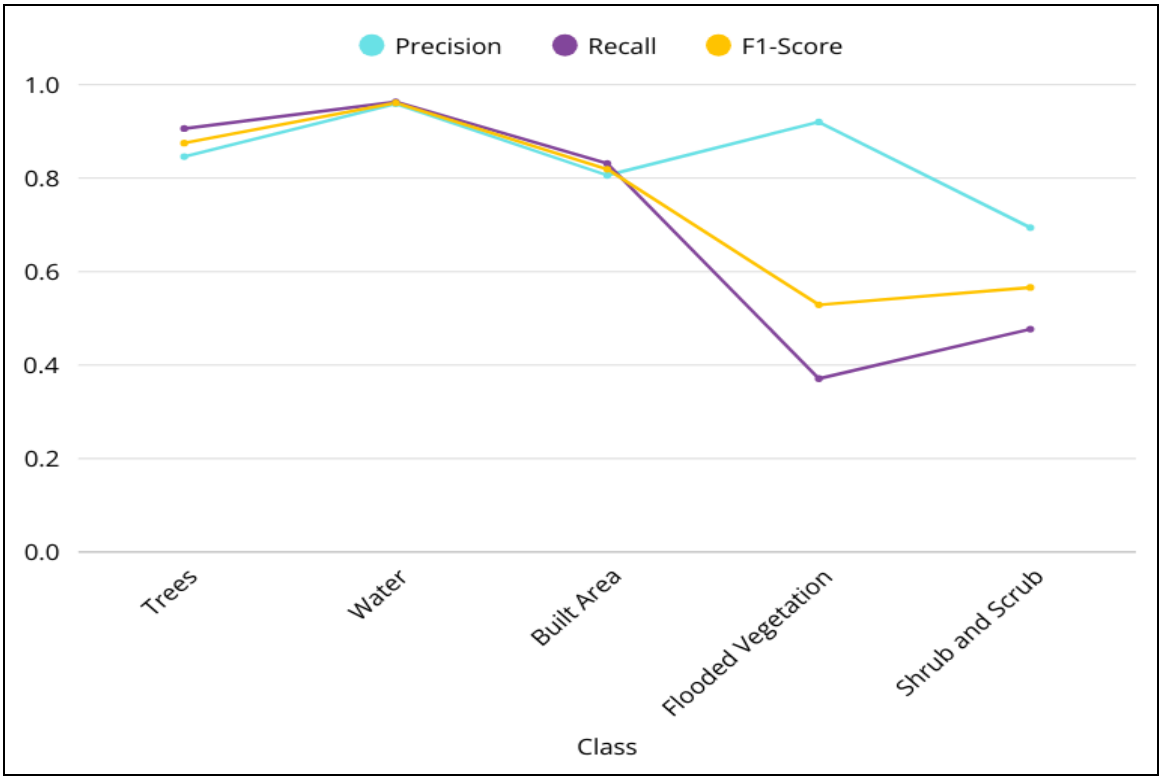

Figure 16: Line chart: Class-wise Performance

7.6 Overall Accuracy and Kappa

- Overall accuracy measures the proportion of correctly classified pixels. Values range from **0.7242 to 0.9025**, indicating strong classification performance across all parameter sets.
- The Kappa coefficient accounts for agreement occurring by chance. Values between 0.7096 and 0.7969 suggest substantial agreement between predicted and true labels, reinforcing the reliability of the classifier.

7.7 Z-Statistic

The Z-statistic quantifies the statistical significance of the kappa value. Higher values (up to 2.56) indicate greater confidence that the observed agreement is not due to random chance. The increase in Z-statistic with window size suggests that larger spatial contexts improve the robustness of the classification.

7.8 Visualizations

EDA plots confirmed class imbalance in the raw data, especially with fewer samples for Snow and Ice. Confusion matrices visually illustrated common misclassification paths. The use of color-coded legends made interpretation intuitive.

7.9 Interpretation and Recommendations

- Larger window sizes and higher sampling ratios generally improve classification accuracy and statistical confidence but at the cost of increased computation.
- Stratified sampling is recommended for smaller training sets to ensure all classes are represented, while random sampling can be effective when using a larger proportion of the data.
- The classifier performs robustly across a range of parameter settings, with only modest variations in accuracy and kappa, indicating the method's stability for land cover mapping tasks.

8. Limitations and Scope of future work

8.1 Limitations

- **Dependence on Reference Labels**: The code requires a reference label image for supervised classification and accuracy assessment. If the reference data is inaccurate, outdated, or misaligned with the Sentinel-2 image, the classification results and accuracy metrics will be unreliable.
- **Limited to Sentinel-2 and .tif Format:** The current implementation is tailored for Sentinel-2 imagery in .tif format. It does not natively support other satellite data sources (e.g., Landsat, MODIS) or file formats, limiting its applicability.
- **Fixed Class Definitions:** The class labels and color mappings are hardcoded for nine specific land cover types. The system cannot easily adapt to different classification schemes or a different number of classes without code modification.
- **Random Forest Only**: The code uses a Random Forest classifier with fixed hyperparameters. There is no option for users to select other algorithms (e.g., SVM, neural networks) or to tune hyperparameters, which may limit classification performance in some scenarios.
- **Computational Efficiency**: For large images or large window sizes, the extraction of windowed samples and training can be computationally intensive and memory-demanding, potentially limiting scalability to very large datasets.
- **No Handling of Missing Data or Clouds:** There is no explicit handling of missing data, cloud cover, or other common remote sensing artifacts, which can affect classification accuracy.

8.2 Scope for Future Work

- **Support for Multiple Data Sources and Formats**: Extend the code to handle other satellite sensors (e.g., Landsat, MODIS) and additional file formats, increasing its versatility for different remote sensing applications.
- **Dynamic Class Definitions**: This allows users to define their own class labels and color schemes through the interface, making the tool adaptable to various land cover classification schemes.

- **Algorithm Selection and Hyperparameter Tuning**: Provide options for users to select different classification algorithms (e.g., SVM, k-NN, neural networks) and tune their hyperparameters via the interface.
- **Scalability and Performance Optimization**: Optimize the code for large-scale processing, possibly by:
    - Implementing batch processing or tiling strategies
    - Leveraging parallel computing or GPU acceleration
- **Interactive Visualization**: Integrate interactive map viewers (e.g., using folium or ipyleaflet) to allow users to explore classification results spatially and compare them with original imagery.
- **Automated Accuracy Assessment**: Implement more advanced accuracy assessment techniques, such as cross-validation, bootstrapping, or spatially stratified sampling, to provide more reliable performance metrics.

9. Conclusion

The results demonstrate that careful tuning of window size, sampling ratio, and sampling method can optimize classification performance for land cover analysis using Sentinel-2 imagery. The code's flexibility allows users to balance accuracy, computational efficiency, and class representation according to their specific application needs

This study successfully demonstrated the application of machine learning techniques—specifically the Random Forest (RF) classifier—for land cover classification using Sentinel-2 satellite imagery. By experimenting with different parameters such as window size and sample ratio, the analysis explored how spatial context and data volume influence classification accuracy and class-wise performance. The results indicated that incorporating a moderate spatial window and an adequate, balanced training sample can significantly enhance model performance, especially for dominant land cover types such as 'Trees' and 'Water.'

Despite promising overall accuracy, the study revealed several challenges, including class imbalance, spectral confusion among similar classes, and limited generalizability. The analysis highlighted how underrepresented classes like 'Snow and Ice' or 'Flooded Vegetation' were difficult to classify accurately with traditional Random Forest models based solely on spectral reflectance. These limitations underscore the need for integrating additional features, richer datasets, and more sophisticated models in future work.

Overall, the project provides a foundational approach for automated land cover mapping using open-access remote sensing data and open-source tools. It affirms the utility of Random Forest as a reliable baseline classifier and offers a clear path for further enhancement through the adoption of advanced machine learning techniques, inclusion of multi-temporal and multi-sensor data, and implementation of automated tuning and sampling strategies.

With continued development and integration of more robust methodologies, this approach holds strong potential for scalable, accurate, and timely land cover monitoring in support of environmental management, urban planning, and climate resilience initiatives.

10. References

[1] R. G. Congalton, "A review of assessing the accuracy of classifications of remotely sensed data," *Remote Sensing of Environment*, vol. 37, no. 1, pp. 35–46, 1991. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/003442579190048B

[2] M. Story and R. G. Congalton, "Accuracy assessment: A user's perspective," *Photogrammetric Engineering and Remote Sensing*, vol. 52, no. 3, pp. 397–399, 1986. [Online]. Available: https://www.asprs.org/wp-content/uploads/pers/1986journal/mar/1986_mar_397-399.pdf

[3] G. M. Foody, "Status of land cover classification accuracy assessment," *Remote Sensing of Environment*, vol. 80, no. 1, pp. 185–201, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0034425701002954

[4] R. G. Congalton and K. Green, *Assessing the Accuracy of Remotely Sensed Data: Principles and Practices*, CRC Press, 1999. [Online]. Available: https://www.taylorfrancis.com/books/mono/10.1201/9780429052729

[5] Congalton, R. G. (1991). A review of assessing the accuracy of classifications of remotely sensed data. *Remote Sensing of Environment*, 37(1), 35–46. https://doi.org/10.1016/0034-4257(91)90048-B

[6] Foody, G. M. (2002). Status of land cover classification accuracy assessment. *Remote Sensing of Environment*, 80(1), 185–201. https://doi.org/10.1016/S0034-4257(01)00295-4

[7] Olofsson, P., et al. (2014). Good practices for estimating area and assessing accuracy of land change. *Remote Sensing of Environment*, 148, 42–57. https://doi.org/10.1016/j.rse.2014.02.015

[8] Zhang, C., et al. (2020). An object-based approach to land cover classification using Sentinel-2 data. *Remote Sensing*, 12(4), 626. https://doi.org/10.3390/rs12040626

[9] Buchhorn, M., et al. (2020). Copernicus Global Land Cover Layers—Collection 2. *Remote Sensing*, 12(6), 1044. https://doi.org/10.3390/rs12061044

[10] Google. (2021). Dynamic World - Near real-time land use classification. https://dynamicworld.app

[11] Congalton, R. G., & Mead, R. A. (1983). A quantitative method to test for consistency and correctness in photointerpretation. *Photogrammetric Engineering and Remote Sensing*, 49(1), 69–74. https://www.asprs.org/wp-content/uploads/pers/1983journal/jan/1983_jan_69-74.pdf:contentReference[oaicite:1]{index=1}

[12] Congalton, R. G., & Green, K. (2009). *Assessing the Accuracy of Remotely Sensed Data: Principles and Practices*(2nd ed.). CRC Press.

https://www.taylorfrancis.com/books/mono/10.1201/9780429052729/assessing-accuracy-remotely-sensed-data-russell-congalton-kass-green:contentReference[oaicite:3]{index=3}

[13] Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1), 37–46. https://doi.org/10.1177/001316446002000104:contentReference[oaicite:5]{index=5}

[14] Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1), 37–46. https://doi.org/10.1177/001316446002000104

[15] Rosenfield, G. H., & Fitzpatrick-Lins, K. (1986). A coefficient of agreement as a measure of thematic classification accuracy. *Photogrammetric Engineering and Remote Sensing*, 52(2), 223–227. https://www.asprs.org/wp-content/uploads/pers/1986journal/feb/1986_feb_223-227.pdf

[16] Ma, Z., & Redmond, R. L. (1995). Tau coefficients for accuracy assessment of classification of remote sensing data. *Photogrammetric Engineering and Remote Sensing*, 61(4), 435–439. https://www.asprs.org/wp-content/uploads/pers/1995journal/apr/1995_apr_435-439.pdf

[17] Næsset, E. (1996). Use of the τ coefficient in accuracy assessment of maps derived from remotely sensed data. *International Journal of Remote Sensing*, 17(6), 1183–1190. https://doi.org/10.1080/01431169608949067

[18] Rossiter, D. G. (2001). A theoretical framework for land evaluation. *Geoderma*, 90(3-4), 207–228. https://doi.org/10.1016/S0016-7061(99)00048-0

[19] Zhu, X. X., & Tuia, D. et al. (2017). Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources. *IEEE Geoscience and Remote Sensing Magazine*, 5(4), 8–36. https://doi.org/10.1109/MGRS.2017.2762307

[20] Congalton, R. G. (1991). *A review of assessing the accuracy of classifications of remotely sensed data*. Remote Sensing of Environment, 37(1), 35–46. https://doi.org/10.1016/0034-4257(91)90048-B

[21] Foody, G. M. (2002). *Status of land cover classification accuracy assessment*. Remote Sensing of Environment, 80(1), 185–201. https://doi.org/10.1016/S0034-4257(01)00295-4

[22] Story, M., & Congalton, R. G. (1986). *Accuracy assessment: A user's perspective*. Photogrammetric Engineering and Remote Sensing, 52(3), 397–399. https://www.asprs.org/wp-content/uploads/pers/1986journal/mar/1986_mar_397-399.pdf

[23] Stehman, S. V. (1997). *Selecting and interpreting measures of thematic classification accuracy*. Remote Sensing of Environment, 62(1), 77–89. https://doi.org/10.1016/S0034-4257(97)00083-7

[24] Olofsson, P., Foody, G. M., Herold, M., Stehman, S. V., Woodcock, C. E., & Wulder, M. A. (2014). *Good practices for estimating area and assessing accuracy of land change*. Remote Sensing of

Environment, 148, 42–57.
https://doi.org/10.1016/j.rse.2014.02.015

[25] Congalton, R. G. (1991). *A review of assessing the accuracy of classifications of remotely sensed data*. Remote Sensing of Environment, 37(1), 35–46. https://doi.org/10.1016/0034-4257(91)90048-B

[26] Landis, J. R., & Koch, G. G. (1977). *The measurement of observer agreement for categorical data*. Biometrics, 33(1), 159–174. https://doi.org/10.2307/2529310

[27] Foody, G. M. (2002). *Status of land cover classification accuracy assessment*. Remote Sensing of Environment, 80(1), 185–201. https://doi.org/10.1016/S0034-4257(01)00295-4

[28] Rosenfield, G. H., & Fitzpatrick-Lins, K. (1986). *A coefficient of agreement as a measure of thematic classification accuracy*. Photogrammetric Engineering and Remote Sensing, 52(2), 223–227. https://www.asprs.org/wp-content/uploads/pers/1986journal/feb/1986_feb_223-227.pdf

[29] Ma, Z., & Redmond, R. L. (1995). *Tau coefficients for accuracy assessment of classification of remote sensing data*. Photogrammetric Engineering and Remote Sensing, 61(4), 435–439. https://www.asprs.org/wp-content/uploads/pers/1995journal/apr/1995_apr_435-439.pdf

[30] Stehman, S. V. (1997). *Selecting and interpreting measures of thematic classification accuracy*. Remote Sensing of Environment, 62(1), 77–89. https://doi.org/10.1016/S0034-4257(97)00083-7

[31] Google Dynamic World Dataset Catalog. *https://developers.google.com/earth-engine/datasets/catalog/GOOGLE_DYNAMICWORLD_V1*

11. Appendix: Printout of Codes

```
import gradio as gr

import gradio.themes.base

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import rasterio

import matplotlib.colors as mcolors

from tqdm import tqdm

from collections import Counter

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report, confusion_matrix

from sklearn.model_selection import train_test_split

import tempfile

import os

from datetime import datetime



# --- Theme ---

custom_theme = gr.themes.Base(

   primary_hue="blue",
```

```
    secondary_hue="rose",
    neutral_hue="slate",
    font=["Comic Sans MS", "cursive"],
).set(
    body_background_fill="#f0f8ff",
    block_border_width="2px",
    block_border_color="#a0aec0",
    button_primary_background_fill="#3b82f6",
    button_primary_text_color="white",
)


# --- Class Map and Colors ---
class_map = {
    0: 'Water', 1: 'Trees', 2: 'Grass', 3: 'Flooded Vegetation',
    4: 'Crops', 5: 'Shrub and Scrub', 6: 'Built Area',
    7: 'Bare Ground', 8: 'Snow and Ice'
}
class_colors = {
    0: '#419bdf', 1: '#397d49', 2: '#88b053', 3: '#7a87c6',
    4: '#e49635', 5: '#dfc35a', 6: '#c4281b', 7: '#a59b8f', 8: '#b39fe1'
}
legend_html = "<div style='display:flex;flex-wrap:wrap;gap:10px;'>"
for cid, cname in class_map.items():
    color = class_colors[cid]
```

```
    legend_html += (
        f"<div style='display:flex;align-items:center;gap:5px;'>"
        f"<span
style='display:inline-block;width:18px;height:18px;background:{color};border-radius:4p
x;border:1px solid #888;'></span>"
        f"<span style='font-size:14px;'>{cname}</span>"
        f"</div>"
    )
legend_html += "</div>"
# --- Utility Functions ---
def read_raster(path):
    with rasterio.open(path) as src:
        return src.read(), src.profile, src.read(1)


def plot_class_image(class_data, title):
    cmap = mcolors.ListedColormap([class_colors[i] for i in sorted(class_colors)])
    bounds = np.arange(-0.5, 9.5, 1)
    norm = mcolors.BoundaryNorm(bounds, cmap.N)
    fig, ax = plt.subplots(figsize=(6, 6))
    im = ax.imshow(class_data, cmap=cmap, norm=norm)
    cbar = plt.colorbar(im, ticks=range(9), ax=ax, shrink=0.7)
    cbar.ax.set_yticklabels([class_map[i] for i in range(9)])
    ax.set_title(title)
    ax.axis('off')
    plt.tight_layout()
```

```
    temp_img_path = os.path.join(tempfile.gettempdir(), f"{title.replace(' ',
'_')}_{datetime.now().strftime('%H%M%S')}.png")

    fig.savefig(temp_img_path)

    plt.close(fig)

    return temp_img_path



def generate_eda(class_data, pixel_size):

    flat = class_data.flatten()

    flat = flat[np.isin(flat, list(class_map))]

    counts = Counter(flat)

    total_pixels = sum(counts.values())

    pixel_area = pixel_size ** 2

    df = pd.DataFrame([

        {

            'Class ID': cid,

            'Class Name': class_map.get(cid, 'Unknown'),

            'Pixel Count': count,

            'Area (m²)': count * pixel_area

        } for cid, count in counts.items()

    ])

    df['Percentage (%)'] = 100 * df['Pixel Count'] / total_pixels

    df = df.sort_values('Pixel Count', ascending=False)

    fig, ax = plt.subplots(figsize=(7, 4))

    bars = ax.bar(df['Class Name'], df['Pixel Count'], color='mediumseagreen')
```

```
    ax.set_xticklabels(df['Class Name'], rotation=30, ha='right')
    ax.set_ylabel("Pixel Count")
    ax.set_title("Land Cover Distribution by Pixel Count")
    for bar, pct in zip(bars, df['Percentage (%)']):
        ax.text(bar.get_x() + bar.get_width()/2, bar.get_height(), f'{pct:.2f}%',
                ha='center', va='bottom', fontsize=8)
    plt.tight_layout()
    return df, fig


def compute_detailed_accuracy(cm):
    n_classes = cm.shape[0]
    total = np.sum(cm)
    pi_plus = cm.sum(axis=1) / total
    pj_plus = cm.sum(axis=0) / total
    user_acc = np.diag(cm) / cm.sum(axis=1)
    prod_acc = np.diag(cm) / cm.sum(axis=0)
    si_user = np.sqrt(user_acc * (1 - user_acc) / cm.sum(axis=1))
    si_prod = np.sqrt(prod_acc * (1 - prod_acc) / cm.sum(axis=0))
    ci_user = [(round(u - 1.96*s, 4), round(u + 1.96*s, 4)) for u, s in zip(user_acc,
si_user)]
    ci_prod = [(round(u - 1.96*s, 4), round(u + 1.96*s, 4)) for u, s in zip(prod_acc,
si_prod)]
    oa = np.trace(cm) / total
    po = oa
    pe = np.sum(np.sum(cm, axis=0) * np.sum(cm, axis=1)) / (total ** 2)
```

```
    kappa = (po - pe) / (1 - pe) if pe != 1 else None
    sigma_kappa = np.sqrt((po * (1 - po)) / (total * (1 - pe)**2)) if pe != 1 else None
    z = (kappa / sigma_kappa) if sigma_kappa else None
    tau = (po - pe) / (1 - pe) if pe != 1 else None
    overall_df = pd.DataFrame({
        'Metric': ['Overall Accuracy', 'Kappa', 'Sigma Kappa', 'Tau', 'Z-Statistic'],
        'Value': [round(oa, 4), round(kappa, 4), round(sigma_kappa, 4) if sigma_kappa
else '',
                  round(tau, 4) if tau else '', round(z, 4) if z else '']
    })
    return overall_df


def save_confusion_matrix_excel(cm, oa, kappa, sigma_kappa, tau, z, class_names):
    total = np.sum(cm)
    pi_plus = cm.sum(axis=1) / total
    pj_plus = cm.sum(axis=0) / total
    user_acc = np.diag(cm) / cm.sum(axis=1)
    prod_acc = np.diag(cm) / cm.sum(axis=0)
    si_user = np.sqrt(user_acc * (1 - user_acc) / cm.sum(axis=1))
    si_prod = np.sqrt(prod_acc * (1 - prod_acc) / cm.sum(axis=0))
    ci_user = [(u - 1.96 * s, u + 1.96 * s) for u, s in zip(user_acc, si_user)]
    ci_prod = [(o - 1.96 * s, o + 1.96 * s) for o, s in zip(prod_acc, si_prod)]
    matrix_data = []
    for i in range(len(class_names)):
```

```
        row = [int(cm[i][j]) for j in range(len(class_names))]

        row.append(int(cm[i].sum()))

        row.append(round(pi_plus[i], 4))

        row.append(round(user_acc[i], 4))

        row.append(round(si_user[i], 4))

        row.append(f"{round(ci_user[i][0], 4)} - {round(ci_user[i][1], 4)}")

        matrix_data.append(row)

    columns = class_names + ['Total', 'pi+', 'Ci', 'si', '95% CI of Ci']

    df_matrix = pd.DataFrame(matrix_data, index=class_names, columns=columns)

    df_matrix.loc['Total'] = [int(cm[:, j].sum()) for j in range(len(class_names))] +
[int(total), 1.0, '', '', '']

    df_matrix.loc["Producer's reliability (Oj)"] = [round(prod_acc[j], 4) for j in
range(len(class_names))] + ['', '', '', '', '']

    df_matrix.loc['si (Prod)'] = [round(si_prod[j], 4) for j in
range(len(class_names))] + ['', '', '', '', '']

    df_matrix.loc['95% CI of Oj'] = [f"{round(ci_prod[j][0], 4)} -
{round(ci_prod[j][1], 4)}" for j in range(len(class_names))] + ['', '', '', '', '']

    overall_df = pd.DataFrame({

        'Metric': ['Overall Accuracy', 'Kappa', 'Sigma Kappa', 'Tau', 'Z-Statistic'],

        'Value': [round(oa, 4), round(kappa, 4), round(sigma_kappa, 4) if sigma_kappa
else '',

                  round(tau, 4) if tau else '', round(z, 4) if z else '']

    })

    temp_dir = tempfile.gettempdir()

    file_name = f"Confusion_Matrix_{datetime.now().strftime('%Y%m%d_%H%M%S')}.xlsx"

    output_path = os.path.join(temp_dir, file_name)
```

```
    with pd.ExcelWriter(output_path) as writer:
        df_matrix.to_excel(writer, sheet_name='Confusion Matrix')
        overall_df.to_excel(writer, sheet_name='Accuracy Metrics', index=False)
    return output_path, df_matrix.style.to_html()


def extract_window_samples(img, labels, window_size):
    pad = window_size // 2
    h, w = labels.shape
    img_padded = np.pad(img, ((0, 0), (pad, pad), (pad, pad)), mode='reflect')
    label_padded = np.pad(labels, ((pad, pad), (pad, pad)), mode='reflect')
    samples = []
    targets = []

    for i in range(pad, h + pad):
        for j in range(pad, w + pad):
            label = label_padded[i, j]
            if label in class_map:
                window = img_padded[:, i - pad:i + pad + 1, j - pad:j + pad + 1]
                features = window.flatten()
                samples.append(features)
                targets.append(label)
    return np.array(samples), np.array(targets)


def sample_pixels(X, y, ratio, method):
```

```
    X_sampled = []
    y_sampled = []
    rng = np.random.default_rng(42)
    if method == "Stratified":
        for cls in np.unique(y):
            idx = np.where(y == cls)[0]
            n = int(len(idx) * ratio)
            sampled_idx = rng.choice(idx, size=n, replace=False)
            X_sampled.append(X[sampled_idx])
            y_sampled.append(y[sampled_idx])
        return np.vstack(X_sampled), np.concatenate(y_sampled)
    elif method == "Random":
        n = int(len(y) * ratio)
        sampled_idx = rng.choice(len(y), size=n, replace=False)
        return X[sampled_idx], y[sampled_idx]
def label_type(idx):
    if str(idx).isdigit() and int(idx) in class_map:
        return f"Class {idx}"
    elif idx == "accuracy":
        return "Accuracy"
    elif idx == "macro avg":
        return "Macro Average"
    elif idx == "weighted avg":
        return "Weighted Average"
```

```
    else:
        return "Other"



def classify(s2_img_file, label_img_file, window_size, ratio, method):
    s2_img, s2_profile, _ = read_raster(s2_img_file.name)
    labels_img, label_profile, class_data = read_raster(label_img_file.name)
    height, width = class_data.shape
    pixel_size = abs(label_profile['transform'][0])

    X, y = extract_window_samples(s2_img, class_data, window_size)
    if X.shape[0] == 0:
        return "No samples extracted, please check inputs."

    X_sampled, y_sampled = sample_pixels(X, y, ratio, method)
    eda_df_sampled, eda_fig_sampled = generate_eda(y_sampled.reshape(-1, 1),
pixel_size)
    X_train, X_test, y_train, y_test = train_test_split(X_sampled, y_sampled,
stratify=y_sampled, test_size=0.2, random_state=42)

    clf = RandomForestClassifier(n_estimators=100, max_depth=30, n_jobs=-1)
    clf.fit(X_train, y_train)
    y_pred_test = clf.predict(X_test)
    test_report = classification_report(y_test, y_pred_test, output_dict=True)
    #report_df["Type"] = report_df.index.map(label_type)
```

```
    report_df = pd.DataFrame(test_report).transpose().round(3)

    report_df["Type"] = report_df.index.map(label_type)

    report_df["Type"] = report_df.index.map(label_type)

    report_df = report_df[["Type"] + [col for col in report_df.columns if col !=
"Type"]]




    cm = confusion_matrix(y_test, y_pred_test)

    cm_fig, ax = plt.subplots(figsize=(6, 5))

    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ax=ax,

                xticklabels=list(class_map.values()),
yticklabels=list(class_map.values()))

    ax.set_title("Confusion Matrix (Test)")

    ax.set_xlabel("Predicted")

    ax.set_ylabel("Actual")

    plt.tight_layout()



    overall_acc_df = compute_detailed_accuracy(cm)

    excel_path, styled_matrix_html = save_confusion_matrix_excel(

        cm=cm,

        oa=overall_acc_df.loc[0, 'Value'],

        kappa=overall_acc_df.loc[1, 'Value'],

        sigma_kappa=overall_acc_df.loc[2, 'Value'],

        tau=overall_acc_df.loc[3, 'Value'],

        z=overall_acc_df.loc[4, 'Value'],
```

```
        class_names=list(class_map.values())
    )
    return (
        eda_df_sampled,
        eda_fig_sampled,
        eda_fig_sampled.savefig("eda_plot.png") or "eda_plot.png",  # save plot if not already saved
        report_df,
        cm_fig,
        styled_matrix_html,
        overall_acc_df,
        excel_path,
        "✅ Classification completed successfully!"
    )

# --- Gradio App ---
with gr.Blocks(theme=custom_theme, title="Land Cover Classifier") as demo:
    gr.Markdown(
        "<h1 style='text-align: center; color: #3b82f6;'>🌍 Land Cover Classification & Accuracy Analyzer</h1>"
    )
    gr.HTML(legend_html, label="Class Color Legend")

    with gr.Accordion("📁 Upload Data", open=True):
        with gr.Row():
```

```
            s2_input = gr.File(label="Sentinel-2 Image (.tif)", file_types=[".tif"])

            label_input = gr.File(label="Label Image (.tif)", file_types=[".tif"])



    with gr.Accordion("⚙️ Parameters", open=False):

        with gr.Row():

            window_size_input = gr.Slider(minimum=1, maximum=31, step=2, value=5,
label="Window Size (odd)")

            sampling_ratio = gr.Slider(minimum=0.05, maximum=1.0, step=0.05, value=0.2,
label="Sampling Ratio")

            sampling_method = gr.Dropdown(choices=["Stratified", "Random"],
value="Stratified", label="Sampling Method")



    status_text = gr.Textbox(label="Status", value="Waiting for input...",
interactive=False)

    run_btn = gr.Button("🚀 Run Classification", variant="primary")



    with gr.Tab("📊 Sampled EDA"):

        eda_table = gr.Dataframe(label="Sampled EDA Table")

        eda_plot = gr.Plot(label="Sampled EDA Plot")

        eda_plot_download = gr.File(label="Download EDA Plot")



    with gr.Tab("📈 Classification Results"):

        report_table = gr.Dataframe(label="Classification Report")

        cm_plot = gr.Plot(label="Confusion Matrix")

        styled_cm = gr.HTML(label="Styled Confusion Matrix")

        accuracy_df = gr.Dataframe(label="Overall Accuracy Metrics")
```

```
        download_excel = gr.File(label="⬇️ Download Accuracy Report (.xlsx)")


    def run_classification(s2_img_file, label_img_file, window_size, ratio, method):
        result = classify(s2_img_file, label_img_file, window_size, ratio, method)
        if isinstance(result, str):
            return [None] * 8 + [result]
        (
            eda_df_sampled, eda_fig_sampled, eda_plot_path,
            report_df, cm_fig, styled_matrix_html,
            overall_acc_df, excel_path, message
        ) = result
        return (
            eda_df_sampled, eda_fig_sampled, eda_plot_path,
            report_df, cm_fig, styled_matrix_html,
            overall_acc_df, excel_path, message
        )


    run_btn.click(
        fn=run_classification,
        inputs=[s2_input, label_input, window_size_input, sampling_ratio,
sampling_method],
        outputs=[
            eda_table, eda_plot, eda_plot_download,
            report_table, cm_plot, styled_cm,
```

```
            accuracy_df, download_excel, status_text

        ]

    )



demo.launch(share=True)
```