Sentinel-2 Land Cover Classification

Training Data Preparation and Random Forest Classification in QGIS and R

Teaching handout for students. This document describes a full workflow for preparing training data from Urban Atlas 2021, extracting Sentinel-2 predictors, and running Random Forest classification first in R and then in QGIS.

Document scope	Urban Atlas reclassification, random sampling, training patches, predictor extractio
Recommended platform	QGIS 3.x + R (terra, sf, randomForest or ranger, caret optional)
Target classes	1 Water, 2 Urban, 3 Crops, 4 Forest, 5 Other
Sentinel-2 resolution	10 m bands (for example B02, B03, B04, B08)

1. Input data

- Sentinel-2 image clipped to the study area. For a basic classification, use the 10 m bands: B02, B03, B04, and B08.
- Urban Atlas 2021 vector layer in FlatGeobuf format (.fgb).
- QGIS project with both layers in the same coordinate reference system.

Recommended check: confirm that the raster and Urban Atlas layer overlap correctly before any sampling starts.

2. Clip Urban Atlas to the raster extent

Step 2.1 - Create raster extent polygon

- Processing Toolbox -> Polygon from layer extent
- Input layer: Sentinel-2 raster
- Output: polygon representing the raster extent

Step 2.2 - Clip the Urban Atlas layer

- Processing Toolbox -> Clip
- Input layer: Urban Atlas .fgb
- Overlay layer: raster extent polygon

The result is a vector layer clipped exactly to the Sentinel-2 image footprint.

3. Reclassify Urban Atlas to 5 training classes

Open the attribute table of the clipped Urban Atlas layer and use **Field Calculator**.

- Create a new integer field called **LC_TRAIN**.
- Use the expression below to convert Urban Atlas class IDs into the target classes.

```
CASE

-- WATER
WHEN "code_2021" = '50000' THEN 1

-- URBAN
WHEN "code_2021" IN (
'11100','11210','11220','11230','11240',
'11300','12100','12210','12220','12230',
'12300','12400','13100','13300'
) THEN 2

-- CROPS
WHEN "code_2021" IN (
'21000','22000','24000'
) THEN 3

-- FOREST
WHEN "code_2021" = '31000' THEN 4

-- OTHER
WHEN "code_2021" IN (
'23000','32000','33000','40000','13400',
'14110','14120','14130','14200'
) THEN 5

END
```

Optional - add readable class names

Create a new text field called **LC_NAME** and use the expression below.

```
CASE
WHEN "LC_TRAIN" = 1 THEN 'Water'
WHEN "LC_TRAIN" = 2 THEN 'Urban'
WHEN "LC_TRAIN" = 3 THEN 'Crops'
WHEN "LC_TRAIN" = 4 THEN 'Forest'
WHEN "LC_TRAIN" = 5 THEN 'Other'
END
```

LC_TRAIN	Class name
1	Water
2	Urban
3	Crops
4	Forest
5	Other

4. Dissolve, clean, and sample the training areas

Step 4.1 - Dissolve by class

- Processing Toolbox -> Dissolve
- Input layer: reclassified Urban Atlas layer
- Dissolve field: **LC_TRAIN**

This creates one multipart feature per class, which is easier to manage during sampling.

Step 4.2 - Remove boundary zones with a negative buffer

- Processing Toolbox -> Buffer
- Distance: **-10 m** (or -20 m for stricter edge removal)
- Use the dissolved class layer as input

This step avoids placing training points on class edges, where Sentinel-2 pixels are often mixed.

Step 4.3 - Generate random points in polygons

- Run the tool separately for each class or use filtered class layers.
- Suggested candidate points: **70-100** per class.
- Suggested global minimum distance: **50-100 m**.
- After visual checking, keep approximately **50 training points** and **20 validation points** per class.

Why use candidate points instead of the final number directly?

- Some points may still fall into visually mixed or unsuitable locations.
- A larger candidate set allows manual cleaning while keeping enough final samples.

5. Create training patches around the points

Do not train the model from points alone unless the project explicitly requires single-pixel sampling. A small patch is more stable and less sensitive to noise.

- Recommended patch size: **5 x 5 pixels** = 50 x 50 m.
- Alternative patch size: **10 x 10 pixels** = 100 x 100 m.

In QGIS, the simple practical solution is to create polygons around the points:

- Processing Toolbox -> Buffer
- Distance **25 m** for 5 x 5 pixel patches
- Distance **50 m** for 10 x 10 pixel patches

A circular buffer is acceptable for teaching and practical exercises. If exact raster-aligned squares are required, create a grid aligned to the Sentinel-2 raster and select cells containing the points.

6. Attach class labels and extract predictor values

Step 6.1 - Join point attributes to the patch polygons

- Processing Toolbox -> Join attributes by location
- Input layer: patch polygons
- Join layer: sampled points
- Transfer field: **LC_TRAIN** (and LC_NAME if needed)

Step 6.2 - Extract Sentinel-2 values

There are two common options. For student exercises, the simplest is **Zonal statistics** on the patch polygons.

- Processing Toolbox -> Zonal statistics
- Input polygons: training patches
- Raster layer: Sentinel-2 band
- Statistic: **mean** (recommended)
- Repeat for each predictor raster

The output polygon layer will contain one row per training patch and one or more predictor fields, such as mean_B02, mean_B03, mean_B04, and mean_B08.

7. Export the training table

- Save the polygon layer with the extracted values as GeoPackage or Shapefile.
- For R, it is usually easiest to export a CSV table containing predictors and LC_TRAIN.
- Keep training and validation data in separate files if possible.

Minimum recommended predictors

Predictor	Meaning
B02	Blue
B03	Green
B04	Red
B08	Near infrared

You can also add spectral indices such as NDVI or NDBI later, but the workflow below starts from the basic 10 m bands.

8. Random Forest classification in R - step by step

This section assumes that you already have:

- a multi-band Sentinel-2 raster stack
- a training table with predictor values and the field **LC_TRAIN**
- a separate validation table, if available

Step 8.1 - Install and load packages

```
install.packages(c("terra", "sf", "randomForest", "ranger", "caret"))
library(terra)
library(sf)
library(randomForest)
library(ranger)
library(caret)
```

Step 8.2 - Load the Sentinel-2 raster

```
# Example: stack of 10 m bands
s2 <- rast(c("B02.tif", "B03.tif", "B04.tif", "B08.tif"))
names(s2) <- c("B02", "B03", "B04", "B08")
```

Step 8.3 - Load the training table

```
train <- read.csv("training_samples.csv")
train$LC_TRAIN <- as.factor(train$LC_TRAIN)

# Keep only rows with complete predictor data
train <- na.omit(train[, c("B02", "B03", "B04", "B08", "LC_TRAIN")])
```

Step 8.4 - Fit the Random Forest model

Option A uses the classic **randomForest** package. Option B uses **ranger**, which is faster for larger data.

```
# Option A: randomForest
set.seed(42)
rf_model <- randomForest(
  LC_TRAIN ~ B02 + B03 + B04 + B08,
  data = train,
  ntree = 500,
  importance = TRUE
)

print(rf_model)
importance(rf_model)
```

```
# Option B: ranger
set.seed(42)
rf_ranger <- ranger(
  LC_TRAIN ~ B02 + B03 + B04 + B08,
  data = train,
  num.trees = 500,
  importance = "impurity"
)

print(rf_ranger)
```

Step 8.5 - Validate the model with a hold-out table

```
valid <- read.csv("validation_samples.csv")
valid$LC_TRAIN <- as.factor(valid$LC_TRAIN)
valid <- na.omit(valid[, c("B02", "B03", "B04", "B08", "LC_TRAIN")])

pred_valid <- predict(rf_model, newdata = valid)
confusionMatrix(pred_valid, valid$LC_TRAIN)
```

Step 8.6 - Classify the whole raster

For prediction over the raster, use a wrapper that returns a numeric class code. The example below uses the randomForest model and converts the factor prediction to integer codes.

```
predict_rf <- function(model, data) {
  p <- predict(model, data = data)
  as.integer(as.character(p))
}

lc_map <- predict(s2, rf_model, fun = predict_rf, na.rm = TRUE)
writeRaster(lc_map, "RF_LC_map.tif", overwrite = TRUE)
```

Step 8.7 - Optional: convert numeric classes to labels

```
levels_df <- data.frame(
  value = c(1, 2, 3, 4, 5),
  class = c("Water", "Urban", "Crops", "Forest", "Other")
)
levels(lc_map) <- levels_df
plot(lc_map)
```

9. Random Forest classification in QGIS - step by step

There are several ways to classify in QGIS. For student teaching, the most transparent options are either:

- Semi-Automatic Classification Plugin (SCP), if you want a GUI-oriented image classification workflow, or
- QGIS Processing tools if you have a point or polygon training layer and want to build a machine learning model.

Because the exact tool names can vary slightly by QGIS version and installed providers, the workflow below is expressed in operational steps rather than one plugin-specific button sequence.

Step 9.1 - Prepare the predictor raster stack

- Make sure the Sentinel-2 predictor bands have the same extent, resolution, and projection.
- If needed, build a virtual raster or multiband raster from B02, B03, B04, and B08.

Step 9.2 - Prepare the training layer

- Use the patch polygons or points with extracted predictor values.
- The training layer must contain the target field **LC_TRAIN**.

Step 9.3 - Train a Random Forest model

In many QGIS installations, this is done through a processing algorithm under machine learning, classification, or provider-specific tools. The required logic is always the same:

- Input training dataset: vector layer with predictor fields and the class field LC_TRAIN
- Target field: LC_TRAIN
- Predictor fields: B02, B03, B04, B08 (and any additional indices)
- Model: Random Forest
- Trees: start with **500**

If your installation uses SCP, define the training input from regions of interest and run the classification directly on the raster stack.

Step 9.4 - Run the classification on the raster

- Input raster: Sentinel-2 multiband raster or predictor stack
- Trained model: Random Forest model from the previous step
- Output: classified raster

Step 9.5 - Style the output

- Open layer properties -> Symbology -> Paletted/Unique values
- Assign class colors consistently, for example blue for water, grey/red for urban, yellow for crops, dark green for forest, and light green or brown for other

Step 9.6 - Accuracy assessment in QGIS

- Use the independent validation points or validation patches.
- Sample the classified raster at validation locations.
- Compare predicted class with reference class and build a confusion matrix in QGIS, spreadsheet software, R, or Python.

10. Suggested default settings for student exercises

Parameter	Recommended value
Final classes	5
Candidate points per class	70-100
Final training points per class	50
Final validation points per class	20
Minimum point spacing	50-100 m
Negative buffer before sampling	-10 m
Patch size	5 x 5 pixels preferred; 10 x 10 acceptable in homogeneous areas
Random Forest trees	500

11. Frequent mistakes

- Sampling points at class edges or close to mixed boundaries.
- Using too many samples from a single large polygon and too few from the rest of the study area.
- Training and validating with samples that are spatially too close to each other.
- Using single pixels when the scene is noisy or heterogeneous.
- Forgetting to remove missing values before fitting the model in R.
- Applying the model to a raster whose predictor names do not match the names used in training.

12. One-sentence workflow summary

Clip Urban Atlas -> reclassify to LC_TRAIN -> dissolve -> negative buffer -> random points -> patches -> extract Sentinel-2 predictors -> train Random Forest in R or QGIS -> validate independently.