Sen4CAP - Sentinels for Common Agricultural Policy

Design Justification File

ATBD for Bare Soil detection

Milestone	CCN2 - Milestone 2
Authors	Diane HEYMANS, Sophie BONTEMPS, Pierre DEFOURNY, Laurentiu NICOLA, Cosmin UDROIU
Distribution	ESA - Zoltan SZANTOI

This page is intentionally left blank

Table of recorded changes

Version	Date	Reason
V0.1	13/01/2023	Internal version
v1.0	15/03/2023	First version delivered to ESA (CCN2 - MS2)
v1.1	29/03/2023	Update of the first version based on ESA comments (CCN2 MS2)

Table of contents

List of figures

List of tables

List of algorithms

References

ID	Title
RD.1	Sen4CAP Design Definition File - ATBD for the Subsidy Application Layer Preparation, version 1.1, 30 March 2021
RD.2	Sen4CAP Design Justification File: ATBD for Markers DataBase (July 2022) version 1.0
RD.3	Rouibah, K., Belabbas, M. 2020. Applying Multi-Index Approach from Sentinel-2 Imagery to Extract Urban Areas in Dry Season (Semi-Arid Land in North East Algeria). Revista de Teledetección, 56, 89-101. https://doi.org/10.4995/raet.2020.13787

List of acronyms

Acronym	Definition
ATBD	Algorithm Theoretical Basis Document
AL	Arable Land
BCK	Backscatter
BS	Bare Soil
BSI	Bare Soil Index
EAA	Eligible Agricultural Area
GSAA	GeoSpatial Aid Application
LPIS	Land Parcel Identification System
LUT	Look-Up Table
NBR2	Normalized Burned Ratio 2
NBS	Non-Bare Soil
NDTI	Normalized Difference Tillage Index
NDVI	Normalized Difference Vegetation Index
NDWI	Normalized Difference Water Index
MDB	Marker DataBase
PA	Paying Agency
ROI	Region Of Interest
RF	Random Forest
S1	Sentinel-1
S2	Sentinel-2
SAR	Synthetic Aperture Radar
Sen2-Agri	Sentinel-2 for Agriculture
SMOTE	Synthetic Minority Over-Sampling Technique
SWIR	Short-Wave Infrared
UTM	Universal Transvere Mercator

1. Logical model – overview of the processor

The developed bare soil module relies on a Random Forest (RF) algorithm, run on optical Sentinel-2 (S2) surface reflectance and Synthetic Aperture Radar (SAR) Sentinel-1 (S1) time series at the parcel level. The RF algorithm is run on each individual date so that the results can be analyzed in the form of a time series and inform about the duration of the bare soil period(s) during the monitoring seaon.

Figure 1-1 presents the general workflow of the bare soil detection algorithm which is organized into 4 main components:

1. Input data preparation:
   a. Declaration data (hereafter referred to as “subsidy application layer”);
   b. Optical data;
   c. SAR data;
2. Calibration dataset construction:
   a. S2 dataset construction;
   b. S1 dataset construction;
3. Bare Soil Classification:
   a. S2 RF model;
   b. S1 RF model;
4. Time series analysis:
   a. Bare soil periods analysis
   b. S1 and S2 periods combination

Optical and SAR data pre-processing is done in the Sen4CAP system. From the pre-processed data, S1 and S2 signal statistics extractions are performed at the parcel-level and stored into the markers database (MDB). The S2 time series are stored in the MDB1 that contains therefore the basic single-date markers of S2 bands and L3B variables. The S1 time series, on the other hand, are generated using weekly resampled images extracted at the parcel level (which are stored in the MDB L4A in the actual system v3.1).

The algorithm can be launched with only S2, with S1 and S2 but cannot be launched with S1 only since the S1 calibration dataset is based on the S2 values. Nevertheless, the S1 informations is usefull to caracterise the soil status when the S2 is not available.

In the following sections, the different steps are presented in details. For most of these steps, the specific input and output variables, as well as the code or pseudo-code, are given.

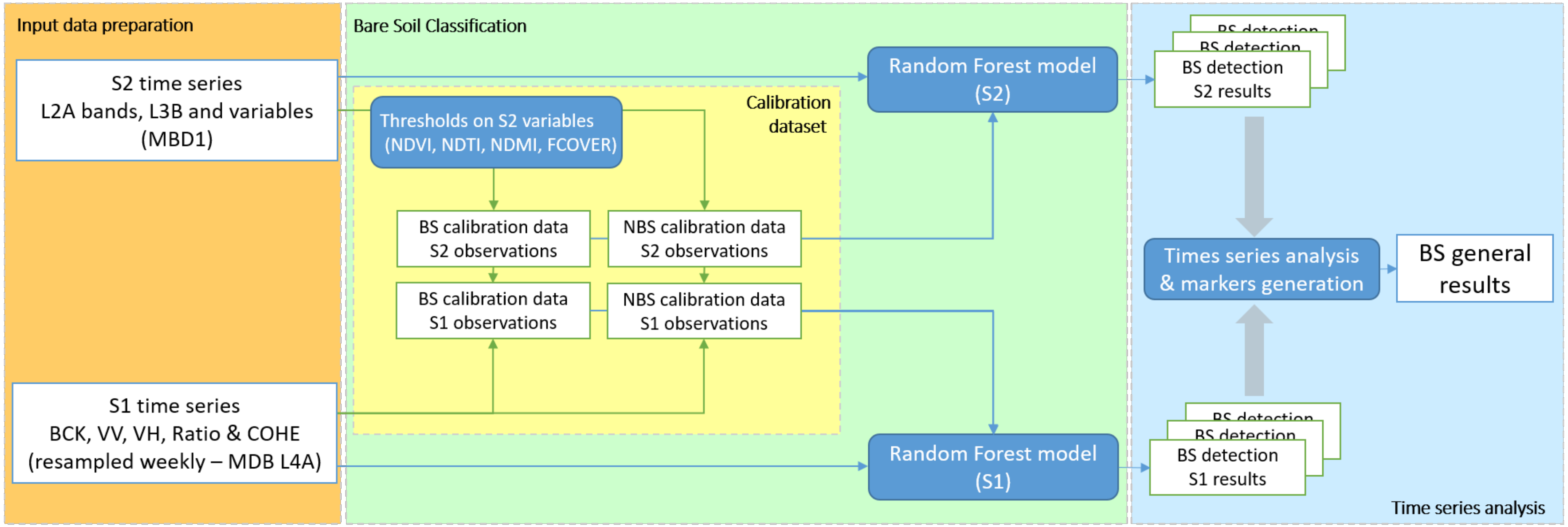

Figure 1-1. General workflow of the L4E bare soil detection algorithm

2. Input data preparation

2.1 Subsidy application layer

In order to ensure a certain level of consistency between the different Sen4CAP processors, the preparation of the subsidy application layer is performed prior to the execution of all processors. The subsidy application layer preparation is described in a dedicated ATBD (RD.1). The outputs of the subsidy application layer preparation that are used by the L4E bare soil detection processor are described below.

2.1.1 Standardized subsidy application layer with quality flags

The standardized subsidy application layer with quality flags (Table 2-1):

- is stored as a PostGIS layer in the PostgreSQL database of the system;
- is projected in national projection;
- has the following name: decl_{site}_{year};
- has the same number of rows (parcels) than the original subsidy application layer.

Table 2-1. Standardized subsidy application layer with quality flags

Output data	Description	Default value [format]
decl_{site}_{year}	The standardized version of the subsidy application layer with the quality flags: geometry and spectral information	[GPKG] & [CSV]

It contains the attribute fields listed in Table 2-2 (fields in orange are already present in the original subsidy application layer). Attributes coming from the Look-Up Table (LUT) shown in Table 2-3 are also available in the layer at the parcel level. This layer is available as .gpkg and .csv.

Table 2-2. Content of the standardized subsidy application layer with quality flags

Field name	Role	Default value [format]
Ori attributes	All the original attributes of the original delaration dataset	[integer, float or string]
ori_id	Copy of the content of the attribute field defined by the user with the parcel id	[string]
ori_hold	Copy of the content of the attribute field defined by the user with the holding id	[string]
ori_crop	Copy of the content of the attribute field defined by the user with the crop code	[input format: string or integer]
NewID	New sequential ID of the parcel	[integer]
HoldID	New sequential ID of the holdings	[integer]
GeomValid	Identify parcels for which no polygon exists in the subsidy application layer or with a not valid geometry	[integer, binary]
Duplic	Identify parcels that have the exact same geometry as another	[integer, binary]

Area_meters	Parcel area in the UTM projection (m²)	[integer]
Overlap	Identify parcels which overlaps with neighbouring parcels	[integer, binary]
ShapeInd	The crop type name	[float]
S1pix	Indicates the number of used S1 pixels in the parcel	[integer]
S2pix	Indicates the number of used S2 pixels in the parcel	[integer]

2.1.2 Crop code LUT

If the original subsidy application layer contains a large number of crop types, it considerably improves the classification accuracy to group together the crop types that are by definition very similar or that have a very similar phenology. It is done in the crop code LUT, which makes this grouping and defines new crop codes (CTnumL4A) and crop names (CTL4A).

In addition, to check the compliancy of the holdings regarding the crop diversification rules, a series of information should be defined by crop type: the crop diversification class (CTnumDIV and CTDIV) and whether or not it belongs to one or more of the categories Eligible Agricultural Area (EAA), Arable Land (AL), Permanent grassland, Temporary grassland, Fallow land and Crop under water.

All this information is summarized in a csv file, the crop code LUT, which:

- is stored as a table in the PostgreSQL database of the system;
- is named lut_{site}_{year};
- contains the following information (Table 2-3).

These attributes are also stored at the parcel level in the standardized subsidy application (decl_{site}_{year}).

Table 2-3. Content of the L4A crop code LUT

Field name	Role	Default value [format]
Ori_crop	The initial crop code from the subsidy application layer	[integer or string]
CTnum	The new crop type code (each Ori_crop being associated to a unique CTnum)	[integer]
CT	The name of the crop type in English	[string]
LC	The main land cover class of the crop type: ○ 0: other natural areas ○ 1: annual crop ○ 2: permanent crop ○ 3: grassland ○ 4: fallow land ○ 5: greenhouse and nursery	[integer]
CTnumL4A	The new crop type code resulting of the grouping of the CTnum for the classification	[integer]
CTL4A	The crop type name associated to CTnumL4A	[string]
CTnumDIV	The crop diversification class code	[integer]
CTDIV	The crop diversification class name	[string]

EAA	Eligble agricultural area: value 1 if the crop type belongs to this category, value 0 otherwise	[integer, binary]
AL	Arable Land: value 1 if the crop type belongs to this category, value 0 otherwise	[integer, binary]
PGrass	Permanent grassland: value 1 if the crop type belongs to this category, value 0 otherwise	[integer, binary]
TGrass	Temporary grassland: value 1 if the crop type belongs to this category, value 0 otherwise	[integer, binary]
Fallow	Fallow land: value 1 if the crop type belongs to this category, value 0 otherwise	[integer, binary]
Cwater	Crop under water: value 1 if the crop type belongs to this category, value 0 otherwise	[integer, binary]

2.2 Optical data from MDB1

The MDB1 contains basic single-date markers, which correspond to S1 and S2 signal statistics aggregated at the parcel-level [RD.2]. Only the following S2 signal statistics are used in the bare soil processor:

- Blue B2: mean values by parcel;
- Green B3: mean values by parcel;
- NIR B8: mean values by parcel;
- SWIR1 B11: mean values by parcel;
- SWIR2 B12: mean values by parcel;
- NDVI: mean values by parcel;
- LAI: mean values by parcel;
- FAPAR: mean values by parcel;
- FCOVER: mean values by parcel.

NDVI, LAI, FAPAR and FCover statistics are calculated for each L3B product generated from S2 time series by the Sen4CAP L3B processor.

Since the S2 single-dates bands values are extracted for each S2 tile, one parcel can have more than one value for a single-date and for single-bands (overlapping area). In those cases, only the mean of those values is kept.

These S2 signal statistics extracted will be used to compute new indices in the bare soil calibration component (see section 3.1).

The access to this dataset in done using the API of the Sen4CAP system.

2.3 SAR data from MDB4

The MDB4 contains the markers specifically used by the L4A crop type processor. They are the results of a 10-day temporal resampling in the case of S2 data and of a statistics calculation over different periods in the case of S1 data. The MDB4 is devided into four sub-MDB [RD.2]. Only the L4A SAR main features are used in the bare soil processor:

- SAR S1 signal statistics: every week (7 days), based on 20-meters resolution data:

- Mean Amplitude Ascending VV: mean values by parcel;
- Mean Amplitude Ascending VH: mean values by parcel;
- Mean Amplitude Descending VV: mean values by parcel;
- Mean Amplitude Descending VH: mean by parcel;
- Mean Amplitude VV/VH ratio: mean by parcel;
- Mean Coherence Ascending VV: mean by parcel;
- Mean Coherence Ascending VH: mean values by parcel;
- Mean Coherence Descending VV: mean values by parcel;
- Mean Coherence Descending VH: mean values by parcel.

The access to this dataset in done using the API of the Sen4CAP system.

3. Calibration dataset construction

The rationale behind this module is to create a calibration dataset for “bare soil” and “non bare soil” classes based on S2 data without the need for field data but using thresholding techniques. Then, “bare soil” and “non bare soil” dates identified in the S2 dataset are transposed to the S1 dataset to build the S1 calibration dataset. These two steps are done using different python scripts.

A minimum of 3 months - hereafter referred to as “calibration period” or “Period_C” - is needed to build the calibration dataset. Those months can either be included in the “monitoring period” (“Period_M”) if this period is longer than 3 months or come before this monitoring period. In the first case (Period_C included in Period_M), the model will be more accurate than in the second case (in which Period_C belongs to another site and/or another year).

3.1 S2 calibration dataset construction

The calibration dataset is built by applying “bare soil” and “non bare soil” thresholds on various spectral indices and biophysical indicators. The threshold values have been defined using field data in Belgium and other tests on pilots’ countries; they could be slightly adapted for each site but this adaptation would need expert-knowledge and a validation by the user.

During this calibration process, a first step consists of retrieving the signal values of the spectral indices and biophysical indicators from the MDB during the calibration period “Period_C”. New indices are computed for all the parcels. It contains now all the features used to build the model (see next section - 4.1).

At each date with data, the threshold values are applied on the signal values of the selected variables of calibration and the trio “date-parcel-category” is defined into the calibration dataset. Two categories are possible: “bare soil” and “non bare soil”. A small filter is applied on the size of the parcels in order to select only large enough parcels. Only agricultural parcels are kept. The second filter is done with the EAA attributes from the Crop code LUT (see section 2.1.2).

The inputs of the S2 calibration dataset construction step are described in Table 3-1. They are related to the site, monitoring and calibration periods, the minimum parcel size to be used for the calibration and the S2 variables and associated threshold values.

User can specify the S2 variables contained in the list bellow that he finds relevant for discriminating between “bare soil” and “non bare soil” categories in his region. The only constraint is to include the NDVI, which is mandatory. For the selected features, the users will be asked to define the threshold values specific to the “bare soil” and “non bare soil” categories.

The list of S2 features proposed to the users to use as calibration variables:

- NDVI;
- FCover;
- Normalized Difference Water Index (NDWI):

$$\frac{B8 - B11}{B8 + B11}$$

- Normalized Difference Tillage Index – NDTI [RD.3], sometimes called Normalized Burned Ratio 2 (NBR2):

$$\frac{B11 - B12}{B11 + B12}$$

- Bare Soil Index – BSI [RD.3]:

$$\frac{(B11 + B4) - (B8 + B2)}{(B11 + B4) + (B8 + B2)}$$

Table 3-1. Inputs of the S2 calibration dataset construction

Input names	Role	Default value [format]
site	Site of the sen4cap system related to the AOI	[string]
Year	Year used to build the calibration dataset	[int]
Period_C	Period on which the calibration dataset is built. This period is related to the site and year available. Ex. From 01/09/2020 to 01/12/2020. (from *fromdate* to *todate*)	dates
Sel_S2Pix	Minium number of S2 pixels in the parcel to allow a parcel being used in the calibration dataset	50 [int]
features_BS	List of features to be tested to build the calibration dataset of “bare soil”. Only NDVI is mandatory. By-default. [‘NDVI’, ’NDWI’, ’NDTI’, ’BSI’]	List of possible S2 features
features_NBS	List of features to be tested to build the calibration dataset of “non-bare soil”. Only NDVI is mandatory. By default. [‘NDVI’, ’NDWI’, ’NDTI’, FCOVER’, ’BSI’]	List of possible S2 features
thr_bs_{features_BS}	Threshold values for “bare soil” (by default): **NDVI <** NDWI < NDTI < BSI >	[Float] **0.15** 0 0.1 0.15
thr_nbs_{features_NBS}	Threshold values for “non-bare soil” (by default): **NDVI >** NDWI > NDTI > FCOVER > BSI <	[Float] **0.45** 0.3 0.25 0.45 0
list_tile	List of S2 tiles in the site. Exemple. [‘31UFS’,’31UGR’]	List of string
markers_opt_main	S2 statistics to be retrieved from the MDB1. The list of markers varies according to the tiles in the list_tile. By default. ['mean_FAPAR', 'mean_FCOVER', 'mean_LAI', 'mean_NDVI', f'mean_L2A_'+tile+'_B2', f'mean_L2A_'+tile+'_B3', f'mean_L2A_'+tile+'_B4', f'mean_L2A_'+tile+'_B8', f'mean_L2A_'+tile+'_B11', f'mean_L2A_'+tile+'_B12']	List of markers (see section 2.1.2)

3.1.1 Parcels selection and values retrieval

This first step extracts, from the MDB1 and for each S2 tile, the S2 signal statistics that will be needed to build the calibration dataset (markers_opt_main in Table 3-1). This step retrieves markers values from the MDB1 using the API and can therefore be launched from the local host (directly on the Sen4CAP machine) or from another computer having access to the machine.

The inputs of this request are those related to the API as described in the MDB ATBD [RD.2]: “site”, “year”, “parcel_id”, “product_type” and “marker_name”. The “product_type” is related to the MDB in the system for S2 (*s4c_mdb1* - see section 2.2). The “marker_name” parameter identifies the list of markers to retrieve (*markers_opt_main* - see section 2.2). Another parameter is the “wkt_tile” which corresponds to the Region of Interest (ROI) or to the extent of each tile in WKT format[1]. The two last parameters are the “fromdate” and the “todate” that are used to identify the “Period_C” (see Table 3-1).

The markers available can be identified using the following request in the API (Algorithm 3-1).

Algorithm 3-1. Listing available S2 markers in MDB1

```
markers_all =
requests.get(f'http://'+ip_sen4cap+':8080/markers/names?site='+site_name+'&productType=s4c_mdb1&year='+
year, headers=headers).json()["data"]
```

The function also requires the information to connect to the Sen4CAP machine (IP address) and the user name and password of one of the users (by default user: sen4cap, password: sen4cap). The user name and password are needed for the creation of a safe connection with an API token.

For each tile of the List_tile, each statistic identified in *markers_opt_main* are stored in a first table (*traint_all*) – see Algorithm 3-2. The markers related to the S2 bands are renamed to be independent of the tile. This table has X+2 columns and Y rows, X being the number of markers/statistics retrieved and Y being the number of parcels multiplied by the number of dates.

As mentioned in section 2.2, the values of the S2 bands coming from different tiles are averaged to obtained only one value by parcel.

Algorithm 3-2. S2 statistics retrieval for all markers_opt_main list

```
payload = {"user": "sen4cap", "password": "sen4cap"}
result1 = requests.post(f'http://'+ip_sen4cap+':8080/login', data=payload).json()
api_token = result1["data"]["sessionToken"]
headers = {"X-Auth-Token": api_token}

traint_all = pd.DataFrame()
product_type = 's4c_mdb1'
fromdate = '2021-10-01'
todate = '2021-11-01'

for tile in tile_list:
    print(tile)
     markers_opt_main = [ 'mean_FAPAR', 'mean_FCOVER',f'mean_L2A_'+tile+'_B2', f'mean_L2A_'+tile+'_B3',
f'mean_L2A_'+tile+'_B4',f'mean_L2A_'+tile+'_B8',f'mean_L2A_'+tile+'_B11',
f'mean_L2A_'+tile+'_B12','mean_LAI', 'mean_NDVI']
    list_m = ','.join(markers_opt_main)

    raster_5mBuffer = path_lpis+'/decl_'+site+'_'+year+'_'+tile+'_S2.tif'
    rasterR = rioxarray.open_rasterio(raster_5mBuffer)
    xds_lonlat = rasterR.rio.reproject("EPSG:4326")
    extent_tile = xds_lonlat.rio.bounds(())
    geom = box(*extent_tile)
    wkt_tile = urllib.parse.quote(geom.wkt)
    print(wkt_tile)
```

[1] This parameter works only from the version 3.2 of the system

```
                                                    url                                                    =
f"http://{ip_sen4cap}:8080/markers?site={site}&productType={product_type}&year={year}&markers={list_m}&
roi={wkt_tile}&from={fromdate}&to={todate}"
    print(url)
    re = requests.get(url, headers=headers)
    result = re.json()["data"]
    p=0
    tt = len(result['parcels'])
    print('total number of parcels in the tile: ',tt)
    while p < tt:
        if p > tt-2000 :
            re_p = result['parcels'][p:tt]
        else:
            re_p = result['parcels'][p:(p+2000)]
        traint_all1 = pd.DataFrame()
        for i in re_p:
            newid = i['id']
            traint_i = pd.DataFrame()

            traint_i['dates'] = result["dates"]
            for m in markers_opt_main:
                m_b = m.replace(tile+'_',"")
                traint_i[m_b] = i['markers'][m]
            traint_i['NewID'] = i['id']
            traint_all1 = pd.concat([traint_all1,traint_i])

        traint_all = pd.concat([traint_all,traint_all1])
        p=p+2000

traint_all = traint_all.groupby(['NewID','dates']).mean().reset_index()
```

3.1.2 Bare soil and non bare soil features computation

The statistics extracted in the previous step are then used to compute variables not included in the MDB1. These variables can be used as calibration variables and are therefore needed for the threshold application. In all cases, they are used as features in the model. The script to generate them is shown in Algorithm 3-3. The values of these new variables are also stored into the *traint_all* table.

Note: This step is needed in case some variables are not present in the MDB1. If those variables are added in the MDB1, they should be removed of this step and added to the marker_opt_main list.

Algorithm 3-3. Computation of S2 features not included in the MDB1

```
traint_all['mean_NDTI'] = (traint_all[f'mean_L2A_'+tile+'_B11']-
traint_all[f'mean_L2A_'+tile+'_B12'])/(traint_all[f'mean_L2A_'+tile+'_B11']+traint_all[f'mean_L2A_'+til
e+'_B12'])

traint_all['mean_NDWI'] = (traint_all[f'mean_L2A_'+tile+'_B8'] - traint_all[f'mean_L2A_'+tile+'_B11'])
/ (traint_all[f'mean_L2A_'+tile+'_B11']+traint_all[f'mean_L2A_'+tile+'_B8'])

traint_all['mean_BSI'] = ((traint_all[f'mean_L2A_'+tile+'_B11']+traint_all[f'mean_L2A_'+tile+'_B4']) -
(traint_all[f'mean_L2A_'+tile+'_B8']+traint_all[f'mean_L2A_'+tile+'_B2'])) /
((traint_all[f'mean_L2A_'+tile+'_B11']+traint_all[f'mean_L2A_'+tile+'_B4']) +
(traint_all[f'mean_L2A_'+tile+'_B8']+traint_all[f'mean_L2A_'+tile+'_B2']))
```

3.1.3 Threshold application

As mentioned in the first paragraph of this section, not all the parcels imported in the system are used to build the calibration dataset: only those having a number of S2 pixels higher than *Sel_S2Pix* and belonging to the EAA category are kept and stored in *calibration_id*.

Once all the “bare soil” and “non bare soil” features are retrieved and computed over the selected parcels and during the “Period_C”, “bare soil” and “non bare soil” thresholds are applied to build the corresponding S2 calibration dataset.

On one hand, “bare soil” thresholds (thr_bs) are applied on the features_BS in order to identify the duo’s parcel-date that correspond to “bare soil” and thus feed the “bare soil” calibration dataset.

On the other hand, “non bare soil” thresholds (thrs_nbs) are applied on the features_NBS in order to identify the duo’s parcel-date that correspond to “non bare soil” and thus feed the “non-bare soil” calibration dataset.

A new column is created in the *traint_all* table: ‘*cat*’, which will document the bare soil (BS) or non bare soil (NBS) categories of the parcels. In the example below (Algorithm 3-4), the BS and NBS categories are identified based on four S2 features, which are the NDVI, the NDTI, the FCover and the NDWI. All the trio’s parcel-date-category with no assignment of category are removed from the dataset.

A special case occurs when the NDWI is present in the features_NBS. In this case, the parcels with low NDVI (below thr_bs_ndvi) and high NDWI (above thr_nbs_ndwi) are placed in a new category called 'NBS_Water'. This allows a prediction to be made for parcels covered with water and/or snow.

Algorithm 3-4. Identification of bare soil and non bare soil parcels and dates, that will constitute the bare soil and non bare soil calibration datasets

```
traint_all = traint_all[traint_all.NewId.isin(calibration_id)]

traint_all['cat'] = np.nan
traint_all.loc[(traint_all['mean_NDVI'] < thr_bs_ndvi*1000) & (traint_all['mean_NDTI'] <= thr_bs_ndti) & (traint_all['mean_NDWI'] < thr_bs_NDWI),'cat'] = 'BS'

traint_all.loc[ ((traint_all['mean_NDVI'] > thr_nbs_ndvi*1000) & (traint_all['mean_NDTI'] > thr_nbs_ndti) & (traint_all['mean_FCOVER'] > thr_nbs_fcover*1000)) ,'cat'] = 'NBS'

if ‘mean_NDWI’ in features_NBS:

    traint_all.loc[ ((traint_all['mean_NDVI'] < thr_bs_ndvi*1000) & (traint_all['mean_NDWI'] > thr_nbs_NDWI)),'cat'] = 'NBS_Water'

traint_all = traint_all[traint_all['cat'].notnull()]
```

3.1.4 Outputs

The output is the *traint_all* table containing the calibration dataset with the category (BS, NBS, and possibly NBS_Water) and the complete set of S2 features (marker_opt_main and newly created indices) that will be used in the next step of bare soil classification.

The output is named **L4E_BS_CalibrationS2_{site}_{Period_C}.csv**

3.2 S1 calibration dataset construction

The calibration dataset for S1 is built based on the S2 calibration dataset. For each trio’s parcel-date-category coming from the S2 calibration dataset, S1 features are retrieved and stored. The output of this step contains all S1 & S2 features needed to build the two models (see section 4).

The inputs related to the construction of the S1 calibration dataset can be found in Table 3-2. The site, year and “Period_C” are similar to the S2 inputs (Table 3-1).

Table 3-2. Inputs of the S1 calibration dataset construction

Input names	Role	Default value [format]
site	Site of the sen4cap system related to the AOI.	[string]
Year	Year used to build the calibration dataset	[int]
Period_C	Period on which the calibration dataset is build. This period is related to the site and year available. Ex. From 01/09/2020 to 01/12/2020.	dates
Markers_SAR_main	SAR features to be retrieved for the mdb4: ['ASC_VV_BCK_MEAN', 'ASC_VV_COHE_MEAN', 'ASC_VH_BCK_MEAN', 'ASC_VH_COHE_MEAN', 'ASC_RATIO_BCK_MEAN', 'DESC_VV_BCK_MEAN', 'DESC_VV_COHE_MEAN', 'DESC_VH_BCK_MEAN', 'DESC_VH_COHE_MEAN', 'DESC_RATIO_BCK_MEAN']	List of markers from mdb4 (see section 2.3)
traintS2	Calibration dataset of S2 for the Period_C named L4E_BS_CalibrationS2_{site}_{Period_C}.csv	[.csv]

3.2.1 S1 values retrieval from S2 trio's parcel-date-category

The python function to retrieve the S1 values from the MDB is the same as the one used with S2. The only differences are the "product_type" and the "marker_name" inputs. The loop over the S2 tile is not needed here since the values retrieved are available for each parcel every 7 days (see section 2.3).

The "product_type" is related to the MDB in the system for S1 (*s4c_mdb_l4a_sar_main* - see section 2.3). The "marker_name" parameters identify the markers to retrieve (*markers_sar_main* - see section 2.3). The markers available can be identified using the following request in the API (Algorithm 3-6).

Algorithm 3-5. Listing available S1 markers in MDB4

```
markers_allS1 =
requests.get(f'http://'+ip_sen4cap+':8080/markers/names?site='+site_name+'&productType=s4c_mdb_l4a_sar_main&year='+year, headers=headers).json()["data"]
```

The trio's parcel-date-category are stored into a new table *traintS1*. The request on the API is done. The algorithm filters the parcels and dates to keep only those identified in the S2 calibration dataset (*traintS2*).

- The dates included in the table *traintS1* for the parcel *i* are stored into the *idx_i* list;
- When there exist S1 values for the parcel *i* (which is not always the case since S1 pixels are twice as large as S2 pixels), the S1 dates are identified (dates_s) and converted as dates;
- For each date in the *idx_i*, the loop looks for the closest date in the *dates_s* list and stores the corresponding S1 marker values into the *traintS1* table together with its corresponding S1 dates.

Since the s4c_mdb_l4a_sar_main contains only weekly values of SAR variables, the maximum distance between the S2 date and the S1 date is 3 days.

Algorithm 3-6. Steps to retrieve the S1 statistics from the MDB4

```
product_type = "s4c_mdb_l4a_sar_main"
list_m = ','.join(markers_sar_main)
```

```
url=
f"http://{ip_sen4cap}:8080/markers?site={site}&productType={product_type}&year={year}&markers={list_m}"
print(url)
re = requests.get(url, headers=headers)
result = re.json()["data"]
p=0
tt = len(result['parcels'])

for i in tqdm(result['parcels']):
    newid = i['id']
    idx_i = traintS1.loc[traintS1.NewID==i].index

    dates_s = [datetime.strptime(f,'%Y-%m-%d') for f in result["dates"]]
    for idx in idx_i:
        d=datetime.strptime(traintS1.dates[idx],'%Y-%m-%d')
        d_m = min(range(len(dates_s)), key=lambda ii: abs(dates_s[ii]- d))
        for m in markers_sar_main:
            traintS1.at[idx,f'{m}'] = i['markers'][m][d_m]
            traintS1.at[idx,'datesS1'] = dates_s[d_m]
```

3.2.2 Outputs

The output is the table *traintS1* for the “Period_C”. It contains the S1 features as defined in marker_sar_main for the parcels, dates and categories obtained from the S2 calibration dataset, together with the S2 features. The output is named **L4E_BS_CalibrationS2&S1_{site}_{Period_C}.csv**

4. Bare Soil Classification

The bare soil classification step constructs two RF models based on the S1 and S2 calibration datasets. Once the models are created, they are applied to each S1 and S2 date of the monitoring period "Period_M".

4.1 S2 Random Forest Model

The python script here works in two steps. First, a RF model is built based on the S2 calibration dataset. Then, the model is applied on all the parcels of the LPIS having in the MDB for the "Period_M".

The inputs and outputs of the S2 RF model step are presented in Table 4-1

Table 4-1. Inputs of the S2 RF algorithm

Input names	Role	Default value [format]
Training_S2	S2 calibration dataset. Named. **L4E_BS_CalibrationS2_{site}_{Period_C}.csv**	[.csv]
site	Site of the sen4cap system related to the AOI.	[string]
year	Year of the model application	[int]
Period_M	Period of Monitoring.This period is related to the site and year. Ex. From 01/01/2020 to 01/12/2020.	dates
N_estimators	Number of trees in the RF	30 [int]
list_tile	List of S2 tiles in the site. Exemple. ['31UFS','31UGR']	List of string
markers_opt_main	S2 statistics to be retrieved from the mdb1. ['mean_FAPAR', 'mean_FCOVER', 'mean_LAI', 'mean_NDVI', f'mean_L2A_'+tile+'_B2', f'mean_L2A_'+tile+'_B3', f'mean_L2A_'+tile+'_B4', f'mean_L2A_'+tile+'_B8', f'mean_L2A_'+tile+'_B11', f'mean_L2A_'+tile+'_B12']	List of markers (see section 2.1.2)
Markers_S2_new	New S2 indices created from the markers_opt_main statistics in order to complete the S2 features set so that they are compatible with the calibration set. Those variables were defined in section 0. By default. ['mean_NDTI','mean_BSI', 'mean_NDWI']	[list]
Outputs	**Description**	**Default value [format]**
S2_model_output	S2 RF model for the bare soil classification named: **L4E_BS_S2model_{Period_C}.sav**	[.sav]
S2_fig_importance	Figure showing the features importances of the S2 RF model named: **L4E_BSmodelS2featuresimportance_{Period_C}.png**	[.png]

S2_results	S2 results of the bare soil classification named: **L4E_BS_S2results_{site}_{Period_M}.csv**	[.csv]

4.1.1 Computation of the model and features importance

After the import of the S2 calibration dataset in the environment as *training_S2*, the dataset is analysed to see the number of values available in the BS and NBS pools. The number of parcels in the dataset is also printed.

In order to compute the RF model, the columns from the training_S2 used as features in the model are identified due to the fact that they start with 'mean' (*col_var*).

A last check is done to remove any wrong values in the dataset (NaN, Infinite, etc.). The RF is created (*clf*) with *n_estimators* number of trees. Then the model is fit to *x* and *y*, *x* being the values of the features (from *col_var*) and *y* the categories defined in the calibration dataset (column '*cat'*).

The model is then saved.

The plot showing the features' importance is also created as a bar plot containing the importance score for each features of the col_var used in the model. This plot is saved.

All these steps are summarized in the Algorithm 4-1.

Algorithm 4-1. RF algorithm

```
col_var = [col for col in training_dt if col.startswith(('mean'))]
print(col_var)
training = training_dt[np.isfinite(training_dt[col_var]).all(1)]
print(f'Total number of values in the training dataset: {len(training)}')
y = training['cat']
x = training[col_var]
clf = RandomForestClassifier(n_estimators=n_estimators,random_state=11)
clf.fit(x,y)
pickle.dump(clf, open(S2_model_output, 'wb'))

feature_imp = pd.Series(clf.feature_importances_,index=col_var).sort_values(ascending=False)
feature_imp

# Creating a bar plot
plt.figure(figsize=(7, 5))
fig = sns.barplot(x=feature_imp, y=feature_imp.index)
# Add labels to your graph
fig.set(xlabel='Feature Importance Score', ylabel='Features', title="Visualizing Important Features : "+site+" "+Period_C)
plt.legend()
plt.tight_layout()
fig.figure.savefig(S2_fig_importance)
```

4.1.2 Predictions at the parcel level

A request on the API allows to get all the values for all the *markers_opt_main* for each S2 tile. Each parcel is analysed as followed (Algorithm 4-2):

- The values of the parcels retrieved from the MDB over the "Period_M" are stored for each parcel into a table called *df_marker*;
- The dates with no-data are removed from the *df_marker*;
- Another table (*df_marker_all*) containing the values for all the parcels is filled in with all the *df_marker* tables;

- Variables not available in the MDB (*marker_S2_new*) are created and added to *df_marker*;
- The dates and NewID are isolated from *df_marker* since this information is not used in the prediction;
- If values are available in *df_marker* for the parcel, predictions of bare soil for the parcels are stored into *df_results* for all the dates with values during the "Period_M".

If the parcel is in the overlap of S2 tiles, values of the S2 bands are averaged and used to create a new prediction for this parcel thanks to the *df_marker_all* table.

Algorithm 4-2. RF model application

```
df_results_all = pd.DataFrame()
df_results_all['NewID'] = np.nan
df_marker_all = pd.DataFrame()
product_type='s4c_mdb1'

fromdate = '2021-01-01'
todate = '2021-12-31'

for tile in list_tile:
    print(tile)
     markers_opt_main = [ 'mean_FAPAR', 'mean_FCOVER',f'mean_L2A_'+tile+'_B2', f'mean_L2A_'+tile+'_B3',
f'mean_L2A_'+tile+'_B4',f'mean_L2A_'+tile+'_B8',f'mean_L2A_'+tile+'_B11',
f'mean_L2A_'+tile+'_B12','mean_LAI', 'mean_NDVI']
    list_m = ','.join(markers_opt_main)

                                                            url                          =
f"http://{ip_sen4cap}:8080/markers?site={site}&productType={product_type}&year={year}&markers={list_m}&
from={fromdate}&to={todate}"
    print(url)
    re = requests.get(url, headers=headers)
    result = re.json()["data"]
    parcels = result['parcels']
    tt = len(parcels)

    for p in tqdm(range(0,10000,1000)):
        re_p = parcels[p:p+1000]
        df_results = pd.DataFrame()
        for i in tqdm(re_p):

            newid = i['id']
            df_results_1 = pd.DataFrame()

            if newid in df_results_all['NewID'].unique():
                df_marker_old = df_marker_all[df_marker_all.NewID==newid]
                df_marker_tile = pd.DataFrame({'dates': result["dates"]})

                for m in markers_opt_main:
                    m_b = m.replace(tile+'_',"")
                    df_marker_tile[m_b] = i['markers'][m]

                df_marker_new = pd.concat([df_marker_old,df_marker_tile])
                df_marker = df_marker_new.fillna(-1).groupby(['NewID','dates']).mean().reset_index()
            else:
                df_marker = pd.DataFrame()
                df_marker['dates'] = result["dates"]

                for m in markers_opt_main:
                    m_b = m.replace(tile+'_',"")
                    df_marker[m_b] = i['markers'][m]

            df_marker['NewID'] = newid
            df_marker = df_marker.replace(-1,np.nan)
            df_marker = df_marker.dropna()
            df_marker_all = pd.concat([df_marker_all,df_marker])
```

```
                                        df_marker['mean_NDWI'] = (df_marker[f'mean_L2A_B8']-
df_marker[f'mean_L2A_B12'])/(df_marker[f'mean_L2A_B8']+df_marker[f'mean_L2A_B12'])
                                        df_marker['mean_NDTI'] = (df_marker[f'mean_L2A_B11']-
df_marker[f'mean_L2A_B12'])/(df_marker[f'mean_L2A_B11']+df_marker[f'mean_L2A_B12'])
                   df_marker['mean_BSI'] = ((df_marker[f'mean_L2A_B11']+df_marker[f'mean_L2A_B4']) -
(df_marker[f'mean_L2A_B8']+df_marker[f'mean_L2A_B2'])) /
((df_marker[f'mean_L2A_B11']+df_marker[f'mean_L2A_B4']) +
(df_marker[f'mean_L2A_B8']+df_marker[f'mean_L2A_B2']))

            df_dates = df_marker.copy()
            df_marker = df_marker.drop(['dates','NewID'],axis=1)

            if len(df_marker)==0:
                df_results_1['dates'] = np.nan
                df_results_1['pred'] =  np.nan
                df_results_1['conf'] =  np.nan
                df_results_1['NewID'] = np.nan

            else:

                df_results_1['dates'] = df_dates['dates']
                df_results_1['pred'] =  clf.predict(df_marker)
                df_results_1['conf'] =  clf.predict_proba(df_marker).max(axis=1)
                df_results_1['NewID'] = newid

            df_results = pd.concat([df_results,df_results_1])

        df_results_all = pd.concat([df_results_all,df_results])
```

4.1.3 S2 results description

The table containing the S2 results of the bare soil classification (*df_result_all*) has the columns described in Table 4-2. This output is named **L4E_BS_S2results_{site}_{Period_M}.csv** and should ideally be integrated into the MDB.

Table 4-2. Columns of the S2_results

Names	Role	Default value [format]
NewID	Identifier of the parcel in the Sen4CAP system	[int]
dates	Dates of the bare soil observation during the "Period_M" – corresponding to dates with valid S2 observations	[dates]
pred	Prediction of "bare soil" or "non bare soil" from the RF model	BS or NBS (or NBS_Water) [string]
conf	Confidence level in the prediction	0 to 1 [float]

4.2 S1 Random Forest model

The steps with S1 are similar to those of S2. First, the S1 model is created from the S1 calibration dataset then the model is applied to each parcel and each date of the "Period_M".

The inputs and outputs of the S1 RF model step are presented inTable 4-3.

Table 4-3. Inputs of the S1 RF algorithm

Input names	Role	Default value [format]
Training_S1	S1 calibration dataset. Named **L4E_BS_CalibrationS2&S1_{site}_{Period_C}.csv**	[.csv]
site	Site of the sen4cap system related to the AOI	[string]
year	Year of the model application	[int]
Period_M	Period of Monitoring	dates
N_estimators	Number of trees in the RF	30 [int]
Markers_SAR_main	SAR features to be retrieved for the mdb4: ['ASC_VV_BCK_MEAN', 'ASC_VV_COHE_MEAN', 'ASC_VH_BCK_MEAN', 'ASC_VH_COHE_MEAN', 'ASC_RATIO_BCK_MEAN', 'DESC_VV_BCK_MEAN', 'DESC_VV_COHE_MEAN', 'DESC_VH_BCK_MEAN', 'DESC_VH_COHE_MEAN', 'DESC_RATIO_BCK_MEAN']	List of markers from mdb4 (see section 2.3)
Outputs	**Description**	**Default value [format]**
S1_model_output	S1 RF model for the bare soil classification named: **L4E_BSmodelS1_{Period_C}.sav**	[.sav]
S1_fig_importance	Figure showing the features importances of the S1 RF model named: **L4E_BSmodelS1featuresimportance_{Period_C}.png**	[.png]
S1_results	S1 results of the bare soil classification named: **L4E_BS_S1results_{site}_{Period_M}.csv**	[.csv]

4.2.1 Model creation, predictions and results

The model is created such as explained in section 4.1.1. The only change is that the columns used as features in the training_S1 are those finishing with ‘MEAN’ instead of those starting with ‘mean’ in the training_S2.

The Algorithm 4-1 that computes the RF model can be reused in this S1 context and in this case, it will be saved under the name **L4E_BSmodelS1_{Period_C}.sav**

The model works more ore less in the same way as for S2: it gives the prediction at the parcel level. The only difference is that there are no new markers to be computed since all the S1 markers (markers_sar_main) are available in the MDB4. This step in the Algorithm 4-2 is therefore not needed and no need to loop over the S2 tile since all the parcels have the same S1 markers.

The results at the parcel-level for each date of the “Period_M” is saved into **L4E_BS_S1results_{site}_{Period_M}.csv** and has the same caracteristics as the ones described in Table 4-2.

5. Time series analysis

The time series analysis and marker generations should be as understandable as possible. We **strongly recommend to code this part as a Jupyther notebook**.

There are two main parts to this script. First, the analysis of the bare soil periods is done independently for S1 and S2. Then, the creation of a final output that contains the periods of S2 and S1 with an analysis of the overlapping days of the bare soil periods.

The bare soil period is defined as a specific period that starts when a first bare soil prediction is observed and ends when the vegetation is grows back. This period can be observed more than once and is analysed using a set of markers. Some markers aim to count the number of strong detections in the bare soil period, while others help to determine the end of this period.

This analysis is carried out for S2 and for S1 in two different tracks. The S1 and S2 outputs are then compared to see if there is any overlap between the two lines.

5.1 Bare Soil periods analysis

The GetPeriodBS is a Python function that reads the bare soil predictions for each date and (i) looks for a bare soil detection that has been confirmed, then (ii) determines the start and end dates of the "bare soil" period and (iii) gives a confidence level on the period based on the marker's values.

The markers that are used in this analysis are:

- Marker 1 - M1: used to identify the first "bare soil" detection with strong confidence. Once this marker is on, the "bare soil" period is started;
- Marker 2 - M2: used to confirm the first "bare soil" detection. Once this marker is on, the end of the "bare soil" period is searched and, in the meantime, the number of "bare soil" detections is counted;
- Marker 3 - M3: used to inform about the "noise" in the "bare soil" period. It is incremented (+2) for each strong "bare soil" detection, incremented (+1) for each low "bare soil" detection and decremented (-2) for each "non bare soil" detection;
- Marker - M4: used to identify the first "non bare soil" detection with strong confidence. Once this marker is on, the "bare soil" period is ended;
- Marker 5 - M5: used to count the number of "non bare soil" detections during a certain period after the end of the "bare soil" period;
- Marker 6 - M6: used to inform about the "noise" in the "non bare soil" period. It is incremented (+2) for each strong "non bare soil" detection, incremented (+1) for each low "non bare soil" detection and decremented (-1) for each "bare soil" detection.

Once a first period has been detected with the function (i.e. the end date of the period has been defined), the analysis is restarted on the following period to see if other(s) bare soil period(s) can be observed over the parcel.

For each "bare soil" detected by the analysis, a traffic light approach is implemented which interprets the values of the different markers M1 to M6 and associates a confidence level with the bare soil period detection (Conf_BS).

The markers and the function are the same for S1 and S2 but have only differents threshold values (Table 5-1).

The function takes as input the following arguments (Table 5-1). The S_i corresponds to the bare soil prediction results for the parcel "i" coming from the **L4E_BS_S2results_{site}_{Period_M}.csv & L4E_BS_S1results_{site}_{Period_M}.csv**

Table 5-1. Arguments and outputs of the GetPeriodBS function

Arguments	Role	Default value [format]
S_i	Prediction of bare soil results for the parcel i. It contains the columns as described in Table 4-2.	S2_i S1_i [dataframe]
Plong	Standing for "Long Period". It is used as the duration to look for vegetation (NBS) after the end of the bare soil period (END_BS). It has an impact on the M5 and M6.	60 [days]
Pshort	Standing for "Short Period". It is used in two contexts: 1. as the maximum number of days after the start of the bare soil period (START_BS) where if the END_BS is not found, the END_BS is equal to the START_BS. 2. to see if there is a strong detection of BS after the set of the END_BS. When it happens two times, the end of the bs period is restarted.	30 [days]
thr_bs	Threshold of "bare soil" (BS) that indicates the minimum confidence level in the **BS** prediction (coming from the RF algorithm) to consider this detection as a **strong detection**.	S2: 0.8 S1: 0.65 [float]
thr_nbs	Threshold of "non bare soil" (NBS) that indicates the minimum confidence level in the **NBS** prediction (coming from the RF algorithm) to consider this detection as a **strong detection**.	S2: 0.8 S1: 0.7 [float]
Outputs	**Role**	**Default value [format]**
M1	Marker 1 to identify the first "bare soil" detection with strong confidence: - 0: when no observation of strong BS is done - 1: at the first observation of strong BS	0 or 1 [int]
M2	Marker 2 to confirm the first "bare soil" detection. It starts only once M1 is 1. Each time a strong BS is observed, M2 = M2 + 1. The increment stops when the END_BS is observed.	>= 0 [int]
M3	Marker 3 to inform about the "noise" in the "bare soil" period. It starts only once M1 is 1. Each time a strong BS is observed, M3 = M3 + 2 Each time a low BS is observed, M3 = M3 + 1 And each time a low NBS is observed M3 = M3 - 2 The increment stops when the END_BS is observed.	-x to x [int]

M4	Marker 4 to identify the first "non bare soil" detection with strong confidence: - 0: when no observation of strong NBS is done1: at the first observation of strong NBS after M2 >= 1	0 or 1 [int]
M5	Marker 5 used to count the number of "non bare soil" detections after the end of the "bare soil" period. It starts only once M4 is 1. When a strong NBS is observed, M5 = M5 +1. The increment stops *Plong* days after the END_BS.	>= 0 [int]
M6	Marker 6 used to inform about the "noise" in the "non bare soil" period. It starts only once M4 is 1. When a strong NBS is observed, M6 = M6 +2 When a low NBS is observed M6 = M6 + 1 When a BS is observed, M6 = M6 - 2. The increment stops *Plong* days after the END_BS.	-x to x [int]
START_BS	Date of the start of the "bare soil" period, which corresponds tothe date the 1st strong BS detection	[date]
END_BS	Date of the end of the "bare soil" period. After the confirmation of the BS (second strong BS – M2 =1), it corresponds to the date before the first observation of the strong NBS. As long as there is no strong NBS after the bare soil confirmation, the END_BS is set as 'Continue'.	[date]
Conf_BS	Confidence in the BS period. It can take 5 values: 1. 'Strong' when M2 >= 3, M3 > = 2 and M6 >= 0 2. 'Good' when M2 > 0, M3 >= 0 and M5 > 0 3. 'Medium' when only M2 > 0 and M3 >= 0 4. 'Poor' when M2 >= 0 and M3 < 0 5. 'Doubtful' when only M1 = 1 (only one strong BS)	[string]
NbrBS	Number of dates analysed between START_BS and END_BS. This number helps to understand the value of M2 and M3. If START_BS = END_BS, the number of dates analysed is in the period between START_BS and *Pshort* number of days after.	[int]
NbrTT	Number of dates analysed. i.e. total number of values during the *Period_M* for each parcel.	[int]

5.1.1 BS period function description

The function works by parcel in three main steps, described here and in the Algorithm 5-1. The first two steps run in a loop over each date with a prediction coming from the S_i. The last step is performed when all the dates have been analysed and corresponds to the traffic light approach.

1. The first step looks for the start of the BS period ("START_BS") and gives values to M1, M2 and M3.
    a. if the prediction is a BS with a confidence level above the *thr_bs* (strong BS):

- M1 is set to 1
- "START_BS" is set as the date of this prediction
- The number of predictions "NbrBS" is set to 0
- The loop is going to the next date

b. 'd_continue' is a variable in the loop and has as value the date of the prediction

c. a serie of conditions deals with the few cases of BS not confirmed in a short period after the first detection. These conditions are if M1 = 1, "END_BS" is not set, the period between 'd_continue' and "START_BS" is greater than *Pshort* and M2 = 0. In this case:
- the 'END_BS' is set to "START_BS"
- the 'EndLook' is false (which means that the function stops looking for the "END_BS")
- the 'LookNext' is true (which means that there will be a second analysis in the period after the first BS period using this GetBSperiod function)

d. and another specific case deal when the END_BS is not found after *Plong* period and that a date is found using the transition period (see e.). In that case:
- the 'END_BS is set to baresoil_lastdate, 'EndLook' is false and 'LookNext' is true;

e. In the case where only the M1 = 1 and the "END_BS" is not set (i.e. the period between the START_BS and the Pshort days after the start):
- the number of predictions in the BS period ("NbrBS") is incremented by 1
- if the prediction is a BS, the function starts looking for the end of the bare soil period ("END_BS"), i.e. "EndLook" is True and M4=0. The *baresoil_lastdate* is set as 'nan' and M3_transition is activated (set as 0). Those variables are used when there is a transition zone between the last bare soil predicted and the strong vegetation needed for the set of the "END_BS".
- if the confidence level of the BS is above the *thr_bs,* M2 and M3 are incremented by 1 and 2 respectively
- if the confidenve level of BS is lower than the thr_bs, M3 is incremented by -1
- and if the confidenvel level of NBS is lower than *thr_nbs* (low vegetion predicted), the M3 is incremented of -2 while the M3_transition is incremented of 2 and the baresoil_lastdate is set as the previous date between this low vegetation.

2. The second step looks for the end of the BS period ("END_BS") and gives values to M4, M5 and M6. It is run in parallel of the first step, as soon as M1 = 1 and 'EndLook' is True.

a. The 'LookNext' is True.

b. If the prediction is not a BS with a confidence level above the *thr_nbs*:
- In case the baresoil_lastdate is activated and set, the M3 is incremented of M3_transition and the "NbrBS" is reduced by the number of low vegetation found
- Otherwise, the "END_BS" is set to the date of the previous prediction and M4 is set to 1.

c. If M4 = 1 and the period between "d_continue" and "END_BS" is shorter than *Plong* (only the predictions in the long period after the "END_BS" are taken into account for the calculation of M5 and M6):
- If the prediction is not a BS with a confidence level above the *thr_nbs,* M5 and M6 are incremented by 1.
- If the prediction is a BS with a confidence level above the *thr_bs* in the short period after the "END_BS", M6 is incremented by -2 and if M6 is < -4 (at least two strong BS predictions), the "END_BS" is reset (i.e. "END_BS" is "nan" and M4 = 0).

➢ If the prediction is a BS, M6 is incremented by -1.

There are two specific cases of END_BS:

- ➢ If there is a confirmation of a strong BS (M2 > 0) and M4 is 0, the "END_BS" is set as 'Continue', which means that the BS period is not finished;
- ➢ In the case where M2 = 0 (i.e. the bare soil was not confirmed by a strong 'bare soil') and the M4 is not activated (0 or nan), the END_BS = START_BS.

3. The third step assigns a confidence level to the "bare soil" period detection based on the values of the markers defined in the first two steps. The confidence level is:
   a. Strong if M2 >= 3, M3 > = 2 and M6 >= 0
   b. Good if M2 > 0, M3 >= 0 and M5 > 0
   c. Medium if only M2 > 0 and M3 >= 0
   d. Poor if (M2 >= 0 and M3 <= 0) or (M2 = 0 and M3 >=0)
   e. Doubtful if only M1 = 1 and M3 < 0 (only one strong BS)

Algorithm 5-1. GetPeriodBS function

```
def GetPeriodsBS(S_i,Pshort,Plong,thr_bs,thr_nbs):
    #Analyse S2 - START period
    START_BS = 'nan'
    END_BS = 'nan'
    Conf_BS = np.nan
    M1 = 0
    M2 = np.nan
    M3 = np.nan
    M4 = np.nan
    M5 = np.nan
    M6 = np.nan
    NbrBS = np.nan
    NbrTT = 0
    EndLook = False
    LookNext = False
    ind = S_i.index
    M3_transition = 0
    baresoil_lastdate = 'nan'

    for idx in ind:
        NbrTT += 1
        # look for START_BS

        if (np.isnan(M1)| (M1==0)) & (S_i.at[idx,'pred']=='BS') & (S_i.at[idx,'conf']>=thr_bs):
            M1 = 1
            START_BS = S_i.at[idx,'dates']
            NbrBS = 0
            d_start = datetime.strptime(START_BS,'%Y-%m-%d')
            continue
        elif (M1 != 1):
            d_start = datetime.strptime(f'{year}-01-01','%Y-%m-%d')

        d_continue = datetime.strptime(S_i.at[idx,'dates'],'%Y-%m-%d')

        if (M1 == 1) & (d_continue-d_start > Pshort) & (END_BS=='nan') & (M2 == 0):
            if baresoil_lastdate != 'nan':
                END_BS = baresoil_lastdate
            else:
                END_BS = START_BS
            EndLook = False
            LookNext = True

        elif (M1==1) and (END_BS=='nan'):
            NbrBS += 1
            if np.isnan(M2):
                M2 = 0
```

```
            M3 = 0
        if (S_i.at[idx,'pred']=='BS'):
            EndLook = True
            M4 = 0
            baresoil_lastdate = 'nan'
            M3_transition = 0

        if (S_i.at[idx,'conf'] > thr_bs) & (S_i.at[idx,'pred']=='BS') :
            M2 += 1
            M3 += 2

        elif (S_i.at[idx,'pred']=='BS') & (S_i.at[idx,'conf'] <= thr_bs) :
            M3 += 1

        elif ((S_i.at[idx,'pred']=='NBS') & (S_i.at[idx,'conf'] <= thr_nbs)):
            M3 += -2
            M3_transition += 2
            if baresoil_lastdate=='nan':
                baresoil_lastdate = S_i.at[idx-1,'dates']

    # look for the END_BS

    if (M1==1) & (EndLook) :
        LookNext = True
        if (M4==0) & (S_i.at[idx,'pred']=='NBS') & (S_i.at[idx,'conf']> thr_nbs):
            M4 = 1
            if baresoil_lastdate=='nan':
                END_BS = S_i.at[idx-1,'dates']

            else:
                END_BS = baresoil_lastdate
                M3 += M3_transition
                NbrBS += - M3_transition/2

            d_end = datetime.strptime(END_BS,'%Y-%m-%d')

            continue
        elif (M4!=1):
            M4 = 0
            d_end = d_start

        if (M4==1) & (d_continue-d_end < Plong):
            if np.isnan(M5):
                M5 = 0
                M6 = 0
            if (S_i.at[idx,'conf']>thr_nbs) & (S_i.at[idx,'pred']=='NBS') :
                M5 += 1
                M6 += 2
            elif (S_i.at[idx,'conf']<= thr_nbs) & (S_i.at[idx,'pred']=='NBS'):
                M6 += 1
              elif  (S_i.at[idx,'pred']=='BS')  &  (S_i.at[idx,'conf']>thr_bs)  &  (d_continue-d_end  <
Pshort):
                M6 += -2
                #print('end_next')
                if M6 <= -4 :
                    M4 = 0
                    END_BS = 'nan'
                    continue
            elif (S_i.at[idx,'pred']=='BS'):
                M6 += -2
            else :
                M6 += 0

    #print(d_continue)
    #print(START_BS)
    #print('M2: ' + str(M2))
    #print('M3: ' + str(M3))
    #print('M3_t: ' + str(M3_transition))
    #print(END_BS)
```

```
        #print(M6)
        #print(baresoil_lastdate)

    if ((M2 >= 1) & (M4 == 0)):
        END_BS = 'Continue'
        LookNext = False

    if (END_BS=='nan'):
        Conf_BS = 'nan'
# set the END date if M1 not confirm and END date note found

    if (M2 == 0) & (np.isnan(M4) | (M4 == 0)) :
        END_BS = START_BS

# according to the value of M1-M6 --> assess the confidence level

    if (M1 == 1) & (M2 >= 3) & (M3 >= 2) & (M6 > 0):
        Conf_BS = 'Strong'

    elif (M1 == 1) & (M2 > 0) & (M3 >= 0) & (M5>0):
        Conf_BS = 'Good'

    elif (M1 == 1) & (M2 > 0) & (M3 >= 0):
        Conf_BS = 'Medium'

    elif ((M1==1) & (M2 == 0) & (M3 >= 0)) | ((M2 >= 0) & (M3 <=0)):
        Conf_BS = 'Poor'

    if (M1==1) & (M3 < 0) & (END_BS == START_BS):
        Conf_BS = 'Doubtful'

    return(M1,M2,M3,M4,M5,M6,START_BS,END_BS,Conf_BS,NbrBS,LookNext,NbrTT)
```

5.1.2 Next periods and outputs

Once a first bare soil period has been found, the search for the next period is activated using the variable 'LookNext'. If this variable is True, the S_i is cut to keep only the dates after the "END_BS" (more precisely, it cuts to keep only the dates Pshort days after "END_BS") and it creates the S_i2. The markers related to this new period are saved into a new specific markers output.

This restart depends on the total number of bare soil periods asked by the user (*NperiodS1 and NperiodS2*). This is done for S1 and for S2 independently.

The whole markers and periods analyses are done in a function that calls the GetPeriodsBS function according to the *NPeriod* (S1 or S2). This function takes as inputs:

- ***Sat*** as the satellite type;
- ***NPeriod*** number of "bare soil" periods analysed in the S1 and S2 anlaysis;
- And the inputs of the GetPeriodsBS function.

It returns for each parcel all the markers for each of the *Nperiod* (S1 or S2) of BS as described in Table 5-1. The *LastObs* is another argument added to the markers to inform about the last date analysed in the Period_M.

The outputs for S1 and S2 bare soil periods analysis using the markers are stored into a .csv named respectively **L4E_BS_MarkersS1_{site}_{Period_M}.csv** and **L4E_BS_MarkersS2_{site}_{Period_M}.csv**.

Algorithm 5-2. OutputMarkers function

```
def OutputMarkers(Sat,NPeriod,df_results,i,Pshort,Plong,thr_bs,thr_nbs):
    for p in range(1,NPeriod+1):
        if p == 1:
            S2_i = df_results.loc[df_results.NewID==i]
```

```
            markers =
list(GetPeriodsBS(S_i=S2_i,Pshort=Pshort,Plong=Plong,thr_bs=thr_bs,thr_nbs=thr_nbs))
            NbrTT = markers[11]
            LastObs = lambda x: np.nan if x.empty else x.iloc[-1]['dates']
            df_out1 = {
            'NewID' : i,
            f'NbrTT{Sat}' : NbrTT,
            f'LastObs{Sat}': LastObs(S2_i),
            f'M1_{Sat}_{p}' : markers[0],
            f'M2_{Sat}_{p}' : markers[1],
            f'M3_{Sat}_{p}' : markers[2],
            f'M4_{Sat}_{p}' : markers[3],
            f'M5_{Sat}_{p}' : markers[4],
            f'M6_{Sat}_{p}' : markers[5],
            f'STARTBS_{Sat}_{p}' : markers[6],
            f'ENDBS_{Sat}_{p}' : markers[7],
            f'Conf_{Sat}_{p}' : markers[8],
            f'Nbr{Sat}BS_{p}' : markers[9],
            f'Look_next{Sat}_{p}' : markers[10]
            }

        else :
            if markers[10]:
                            S2_i2   =   df_results.loc[(df_results.NewID==i)   &   (df_results.date_d   >
(datetime.strptime(markers[7],'%Y-%m-%d') + Period_c1))]
                markers = list(GetPeriodsBS(S2_i2,Pshort,Plong,thr_bs,thr_nbs))
            else:
                S2_i2 = df_results.head(0)
                markers = list(GetPeriodsBS(S2_i2,Pshort,Plong,thr_bs,thr_nbs))

            df_out1.setdefault(f'M1_{Sat}_{p}',markers[0])
            df_out1.setdefault(f'M2_{Sat}_{p}',markers[1])
            df_out1.setdefault(f'M3_{Sat}_{p}',markers[2])
            df_out1.setdefault(f'M4_{Sat}_{p}',markers[3])
            df_out1.setdefault(f'M5_{Sat}_{p}',markers[4])
            df_out1.setdefault(f'M6_{Sat}_{p}',markers[5])
            df_out1.setdefault(f'STARTBS_{Sat}_{p}',markers[6])
            df_out1.setdefault(f'ENDBS_{Sat}_{p}',markers[7])
            df_out1.setdefault(f'Conf_{Sat}_{p}',markers[8])
            df_out1.setdefault(f'Nbr{Sat}BS_{p}',markers[9])
            df_out1.setdefault(f'Look_next{Sat}_{p}',markers[10])
    return(df_out1)
```

5.2 Combination of S1 and S2 periods

At the end of the monitoring period, the S2 and S1 “bare soil” periods are compared (if the two lines have been computed) and the confidence level in the S2 detection is increased when there is an overlapping between the detected S1 and S2 periods.

This last function (named S1_S2_comparaison_and_countdays) also allows the calculation of the total number of bare soil days during period_M. This is done for S2, S1 and when the days overlap.

The function also allows to format all the outputs of the **L4E_BS_MarkersAll_{site}_{Period_M}.csv** for each parcel.

5.2.1 S1_S2_comparaison_and_countdays function

The inputs of the function are shown in the table Table 5-2 together with their outputs.

Table 5-2. Inputs and outputs of the S1_S2_comparaison_and_countdays function

Inputs	Description	Default [format]
i	NewID of the parcel to analyse	[int]
dti_S2	Subset of the dataframe containing the S2 results of the bare soil period analysis for the parcel i	one raw dataframe
dti_S1	Subset of the dataframe containing the S1 results of the bare soil period analysis for the parcel i	one raw dataframe
NPeriodS2	Number of periods analysed in the S2 bare soil period analysis	[int]
NPeriodS1	Number of periods analysed in the S1 bare soil period analysis	[int]
Outputs	**Description**	**Default [format]**
i	NewID of the parcels analysed	[int]
PeriodsS2_out	List containing the bare soil periods as analysed in the bare soil period analysis. With p being the S2 period number: - STARTBS_S2_{p} - ENDBS_S2_{p} - Conf_S2_{p} The Conf_S2_{p} is improved when there is an overlapping with the S1 period.	List
NbrS2TT	Number of dates analysed with S2. i.e. total number of cloud-free values during the *Period_M* for each parcel	[int]
LastObsS2	Last date analysed for S2 of the *Period_M*	[date]
tt_daysS2	Total number of days of bare soil observed with S2. i.e sum of all the bare soil periods observed in the parcel	[int]
tt_daysS1S2	Total number of days of bare soil observed with S2 and S1. i.e sum of all the bare soil periods overlapping observed in the parcel	[int]
PeriodsS1_out	List containing the bare soil periods as analysed in the bare soil period analysis. With p being the S1 period number: - STARTBS_S1_{p} - ENDBS_S1_{p} - Conf_S1_{p}	List
NbrS1TT	Number of dates analysed with S1. i.e. total number of values during the *Period_M* for each parcel	[int]
LastObsS1	Last date analysed for S1 of the Period_M	[date]
tt_daysS1	Total number of days of bare soil observed with S1. i.e sum of all the bare soil periods observed in the parcel	[int]

This function works as detailed in Algorithm 5-3. There are three steps which depend on the NPeriod of S1 and S2. The first condition for the whole function is that NPeriodS2 is greater 0:

- If NPeriodS1 is 0, there can be no overlapping. There is only the calculation of the number of days of bare soil with the S2 analysis (1) and the export of the periods of BS for the S2 analysis.
- If NPeriodS1 and NPeriod S2 are greater than 0, overlapping can occur. For each period of the NPeriodS2 (*p*), the calculation of the number of days of bare soil with the S2 analysis is performed first (1). Then for each of the NPeriodS1 (*p1*), it checked whether the M1 of p and the M1 of p are equal to 1. In this case, we take the maximum of the two starts of each period (i.e. the latest date) and the minimum of the two ends of each period (i.e. the earlier date). The delta is calculated. If the delta is greater than 0, there is an overlap. The tt_daysS1S2 is increased by this delta of days and the confidence level of the S2 period (p) is improved (2).
- If NPeriod S1 is greater than 1, the calculation of the number of bare soil days with the S2 analysis is also performed.

(1) <u>Calcultation of the number of bare soil days for one period (p or p1):</u>

When a BS period is detected (M1 of a period is 1), the delta between the ENDBS and the STARTBS is calculated. If this delta is 0, i.e. ENDBS=STARTBS, the tt_days{sat} is incremented by 1, otherwise the tt_days{sat} is incremented by this delta in days.

(2) <u>Calcultation of the improved confidence level</u>:

In the case of overlap between S1 and S2 periods, the value of the confidence level is upgraded to the next level using the levels in the *niv_conf* list. The string '_S1' is added to the confidence level to indicate that it has been improved using the S1 detection.

Algorithm 5-3. ImproveS2confwithS1 function

```
def S1_S2_comparaison_and_countdays(i,dti_S2,dti_S1,NPeriodS2,NPeriodS1):
    tt_daysS2=np.nan
    tt_daysS1=np.nan
    tt_daysS1S2 = np.nan
    dti_S2 = dti_S2.reset_index()
    dti_S1 = dti_S1.reset_index()
          ending    =   lambda   x,s,t:   datetime.strptime(x.loc[0,f'LastObs{s}'],'%Y-%m-%d')   if
x.loc[0,f'ENDBS_{s}_{t}'] in ('Continue','nan') else datetime.strptime(x.loc[0,f'ENDBS_{s}_{t}'],'%Y-%m-
%d')
    delta_zero = lambda x: 0 if x < 0 else x
    periodsS2_out = []
    periodsS1_out = []

    if NPeriodS2!= 0 :
        tt_daysS2 = 0
        NbrS2TT = dti_S2.loc[0,'NbrTTS2']
        LastObsS2 = dti_S2.loc[0,'LastObsS2']
        if NPeriodS1 == 0:

            for p in range(1,NPeriodS2+1):
                periodsS2_out.append(dti_S2.loc[0,f'STARTBS_S2_{p}'])
                periodsS2_out.append(dti_S2.loc[0,f'ENDBS_S2_{p}'])
                periodsS2_out.append(dti_S1.loc[0,f'Conf_S2_{p1}'])
                if dti_S2.loc[0,f'M1_S2_{p}'] == 1:
                                                          deltaS2    =    ending(dti_S2,'S2',p)    -
datetime.strptime(dti_S2.loc[0,f'STARTBS_S2_{p}'],'%Y-%m-%d')
                    if deltaS2.days == 0:
                        tt_daysS2 += 1
                    else:
                        tt_daysS2 += deltaS2.days

        else:
            tt_daysS1S2 = 0
            for p in range(1,NPeriodS2+1):
```

```
                periodsS2_out.append(dti_S2.loc[0,f'STARTBS_S2_{p}'])
                periodsS2_out.append(dti_S2.loc[0,f'ENDBS_S2_{p}'])
                Conf_S2 = dti_S2.loc[0,f'Conf_S2_{p}']
                if dti_S2.loc[0,f'M1_S2_{p}'] == 1:
                                                            deltaS2    =     ending(dti_S2,'S2',p)    -
datetime.strptime(dti_S2.loc[0,f'STARTBS_S2_{p}'],'%Y-%m-%d')
                        if deltaS2.days == 0:
                            tt_daysS2 += 1
                        else:
                            tt_daysS2 += deltaS2.days
                for p1 in range(1,NPeriodS1+1):
                    if (dti_S2.loc[0,f'M1_S2_{p}'] == 1) & (dti_S1.loc[0,f'M1_S1_{p1}'] == 1):
                        startS2 = datetime.strptime(dti_S2.loc[0,f'STARTBS_S2_{p}'],'%Y-%m-%d')
                        startS1 = datetime.strptime(dti_S1.loc[0,f'STARTBS_S1_{p1}'],'%Y-%m-%d')
                        start = max(startS2,startS1)
                        end = min(ending(dti_S2,'S2',p),ending(dti_S1,'S1',p1))
                        delta =  end - start
                        #print(delta)
                        tt_daysS1S2 += delta_zero(delta.days)
                        #print('tt_daysS1S2 : ',tt_daysS1S2)
                        if delta.days > 0:
                            Conf_S2 = niv_Conf[niv_Conf.index(dti_S2.loc[0,f'Conf_S2_{p}'])+1]+'_S1'
                periodsS2_out.append(Conf_S2)

    if NPeriodS1!= 0 :
        tt_daysS1 = 0
        NbrS1TT = dti_S1.loc[0,'NbrTTS1']
        LastObsS1 = dti_S1.loc[0,'LastObsS1']
        for p1 in range(1,NPeriodS1+1):
            periodsS1_out.append(dti_S1.loc[0,f'STARTBS_S1_{p1}'])
            periodsS1_out.append(dti_S1.loc[0,f'ENDBS_S1_{p1}'])
            periodsS1_out.append(dti_S1.loc[0,f'Conf_S1_{p1}'])
            if dti_S1.loc[0,f'M1_S1_{p1}'] == 1:
                                                      deltaS1      =      ending(dti_S1,'S1',p1)      -
datetime.strptime(dti_S1.loc[0,f'STARTBS_S1_{p1}'],'%Y-%m-%d')
                #print(delta)
                if deltaS1.days == 0:
                    tt_daysS1 += 1
                else:
                    tt_daysS1 += deltaS1.days


    return(i,*periodsS2_out,NbrS2TT,LastObsS2,tt_daysS2,tt_daysS1S2,*periodsS1_out,NbrS1TT,LastObsS1,tt
_daysS1)
```

5.2.2 Output of S1 and S2 combination time series analysis

It has the following characteristics:

- It's a table exported in '.csv' called: **L4E_BS_MarkersAll_{site}_{Period_M}.csv;**
- It contains the attributes as described in Table 5-3.

Table 5-3. Output attributes of S1 and S2 combined time series analysis

Attributes	Description	Default value [format]
NewID	New sequential ID of the parcel in the Sen4CAP sytem	[integer]
START_S2_{p}	Start date of the *p* BS period with S2 detection. (p is the period number from 1 to NPeriodS2)	[date]

END_S2_{p}	End date of the *p* BS period with S2 detection. (p is the period number from 1 to NPeriodS2)	[date]
Conf_S2_{p}	Confidence level in the *p* BS period with S2 detection. (p is the period number from 1 to NPeriodS2)	[string]
NbrTTS2	Number of dates analysed. i.e. total number of S2 values during the Period_M for the parcel.	[int]
LastObsS2	Last date analysed for S2 of the Period_M.	[date]
TTdaysS2	Total number of days of bare soil observed with S2. i.e Sum of all the bare soil periods observed in the parcel	[int]
TTdaysS1S2	Total number of days of bare soil observed with S2 and S1. i.e Sum of all the bare soil periods overlapping observed in the parcel.	[int]
START_S1_{p1}	Start date of the *p1* BS period with S1 detection. (p1 is the period number from 1 to NPeriodS1)	[date]
END_S1_{p1}	End date of the *p1* BS period with S1 detection. (p1 is the period number from 1 to NPeriodS1)	[date]
Conf_S1_{p1}	Confidence level in the *p1* BS period with S1 detection. (p1 is the period number from 1 to NPeriodS1)	[string]
NbrTTS1	Number of dates analysed. i.e. total number of S1 values during the Period_M for the parcel.	[int]
LastObsS1	Last date analysed for S1 of the Period_M.	[date]
TTdaysS1	Total number of days of bare soil observed with S1. i.e Sum of all the bare soil periods observed in the parcel	[int]

6. Output

The outputs of the bare soil detection algorithm are divided in three categories, the calibration outputs for S2 only and for the combination S1&S2, the classification outputs for S2 and S1 and markers outputs for S2, S1 and the combination S1&S2.

6.1 Calibration outputs

The calibration outputs consist in two tables:

- The first table contains the dates, the S2 features and the categories of BS and NBS. It is named **L4E_BS_CalibrationS2_{site}_{Period_C}.csv**;
- The second table is a combination of the information of this first table (S2 features and categories) and the information coming from S1 (S1 features and dates). It is named **L4E_BS_CalibrationS2&S1_{site}_{Period_C}.csv.**

6.2 Bare Soil Classification outputs

For each sensor, three outputs are produced in the classification steps of the algorithm.

For S2:

- The RF model for S2 named **L4E_BS_S2model_{Period_C}.sav;**
- The features importance of the S2 RF model named **L4E_BSmodelS2featuresimportance _{Period_C}.png;**
- The S2 results of the classification model named **L4E_BS_S2results_{site}_{Period_M}.csv**, which contains the columns described in Table 4-2.

For S1:

- The RF model for S1 named **L4E_BS_S1model_{Period_C}.sav;**
- The features importance of the S1 RF model named **L4E_BSmodelS1featuresimportance _{Period_C}.png;**
- The S1 results of the classification model named **L4E_BS_S1results_{site}_{Period_M}.csv,** which contains the columns described in Table 4-2.

6.3 Markers outputs

Three outputs are generated during the time series analysis:

1. The table of the S2 bare soil period analysis, which is named **L4E_BS_MarkersS2_ {site}_{Period_M}.csv** and contains the information for each parcel (with values in the MDB1) of the periods detected and their markers values (see Table 5-1);
2. The table of the S1 bare soil period analysis, which is named **L4E_BS_MarkersS1_ {site}_{Period_M}.csv** and contains the information for each parcel (with values in the MDB4) of the periods detected and their markers values (see Table 5-1);
3. The table that contains the periods of S2 and S1 with an analysis of the overlapping days of the bare soil periods. This table is named **L4E_BS_MarkersAll_{site}_{Period_M}.csv** and as the columns as described in Table 5-3.