Supplementary document on random forest classifier

1. Software implementation

The random forest model was built in Python 3.8.5 using the Scikit-learn implementation under version 0.23.2. The numpy Python packackage with version 1.19.4 was for the data preparation.

2. Data preparation

All used features are all based on multispectral Sentinel-2 data. The respective Sentinel-2 data was read into memory, rescaled to 0-1 and masked by a OpenStreetMap road mask. This mask sets values off the road types of interest to no data (nan). Features were first calculated on 2-dimensional grid. To be used for training or prediction in the random forest they had to be flattened, resulting in a 1-dimensional array per feature with as many samples as valid cells given in the Sentinel-2 data array. These arrays combined are a 2-dimensional array with the number of features on the x-axis and the number of samples on the y-axis. Its data type was float16. Only values that are not nan were fed into the model both for training and prediction. Classified values are an array of the length of number of samples.

3. Random forest parameters

Three hyper parameters were tuned using the scikit-learn RandomizedSearchCV with the following value ranges:

- Number of trees (n_estimators): 100-2000 with number of steps=20
- Maximum depth (max_depth): 10-100 with number of steps=20
- Minimum samples per split (min_samples_split): 2-7 with number of steps=6

200 iterations were used with a random state of 42.

The resulting hyper parameters were:

- Number of trees (n_estimators): 800
- Maximum depth (max_depth): 5
- Minimum samples per split (min_samples_split): 90

The random forest classifier was trained with the following parameters:

Table S1. Random forest parameters.

n_estimators	800
max_depth	5
min_samples_split	90
criterion	“gini”
min_samples_leaf	1
min_weight_fraction_leaf	0
max_features	“sqrt”
min_impurity_decrease	0
bootstrap	True
oob_score	True

The class for establishing the model is RandomForestClassifier(), which was trained using its fit() method. 2500 samples per class were included per class, resulting in a training dataset with 10,000 samples. Seven features were included. Hence, 70,000 values were the training input to the random

forest classifier.
Prediction was done using the predict_proba() method. This provides the classification probability per class. For each sample, the class with the highest probability was saved as classification value. Both classification and probabilities were reshaped to the 2-dimensional input grid, resulting in a classification map on the Sentinel-2 grid

4. Random forest evaluation

The evaluation dataset consisted of 3500 samples with seven features, resulting in 24,000 values. These values were fed into the random forest classifier. The result classification was evaluated in the scikit-learn metrics module. A confusion matrix was derived using the confusion_matrix method. Precision was calculated by the precision_score method, recall by the recall_score method, F1 by the f1_score method, the accuracy score using the accuracy_score method.

5. Feature correlation

The figure below shows feature correlations (Pearson r-value).

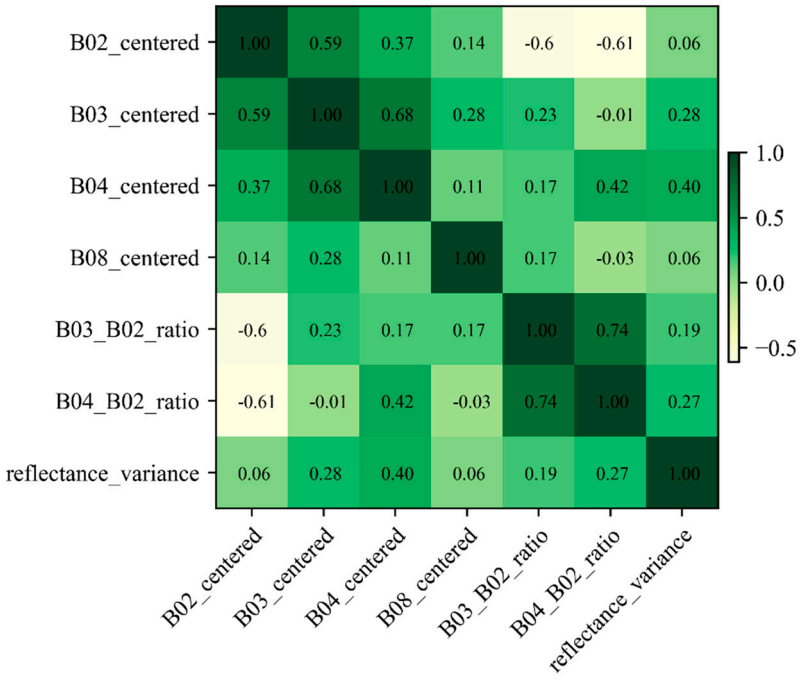


Figure S1. Training feature correlation.

6. Feature importance

The figure below shows resulting feature importance. These were obtained from the intrinsically calculated feature importance provided as an attribute of the trained random forest classifier (feature_importances_).

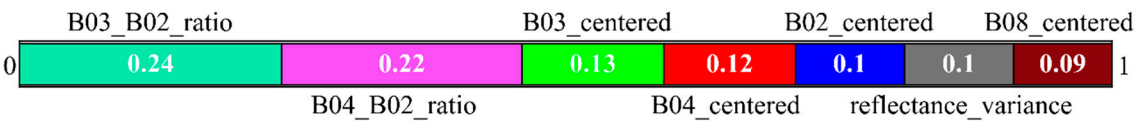


Figure S2. Random forest feature importances.