Supplement of Nat. Hazards Earth Syst. Sci., 22, 3751–3764, 2022
https://doi.org/10.5194/nhess-22-3751-2022-supplement

Natural Hazards and Earth System Sciences
Open Access
EGU

Supplement of

Landsifier v1.0: a Python library to estimate likely triggers of mapped landslides

Kamal Rana et al.

Correspondence to: Kamal Rana (kr7843@rit.edu)

Supplement

S1. Random forest

Random forest (RF) is a decision-tree based ensemble-learning method, a proven and powerful technique for classification and regression (Barnett et al., 2019; Biau, 2012; Biau and Scornet, 2016; Breiman, 2001; Kursa, 2014; Chaudhary et al., 2016; Rodriguez-Galiano et al., 2012). The random forest classifier consists of multiple classifiers, where each classifier bootstraps the training data samples (Breiman, 2001; Liaw et al., 2002). Bootstrapping in each random forest classifier is done by selecting N samples randomly from training samples of size N with replacements. For N training samples bootstrapping N times leads to the approximate selection of $2/3$ of training samples (Azar et al., 2014; Belgiu and Drăguţ, 2016). Hence, each tree in a random forest classifier is trained independently using around $2/3$ of the training samples selected using bootstrapping.

In a binary classifier, as in our case, each parent node q splits into two daughter nodes: right r and left l. Instead of selecting all the p features for node split, a subset of features m (m=$\sqrt{p}$) is selected randomly for each node split (Azar et al., 2014; Okun and Priisalu, 2007). Among m features, one of the features selected for the node split is based on optimizing a criterion. The criterion is called the 'Gini Index,' which measures the features' impurity to the classes. The Gini index of right r and left l daughter nodes are calculated as:

$$G_r = 1 - P_{r1}^2 - P_{r2}^2 \tag{1}$$

$$G_l = 1 - P_{l1}^2 - P_{l2}^2, \tag{2}$$

where P_{rj} (P_{lj}) is the probability of samples in the right (left) daughter nodes having class j. The Gini index is calculated for each predictor in the subset of predictors m, and the features that maximize the change in Gini index is chosen for node split. Change in Gini-index is calculated as:

$$\Delta\theta(s_q) = G_q - \rho_{rq} G_r - \rho_{lq} G_l, \tag{3}$$

where ρ_{rq} (ρ_{lq}) are the ratio of the number of data points in daughter nodes r (l) to the total number of points in the parent node q (Kuhn et al., 2013; Zhang and Ma, 2012). The process of splitting nodes continues until a stopping criterion is met, e.g., when no further samples are remaining, or the Gini-index of parent nodes is lower than the daughter nodes.

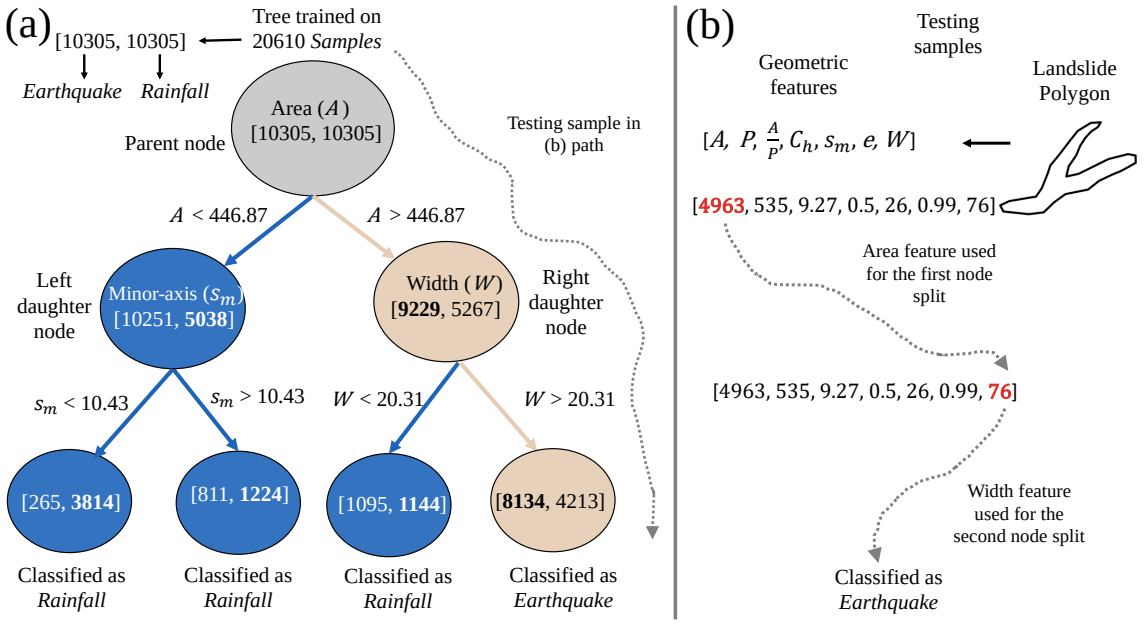

Figure S1. (a) The sample architecture of one of the trees of random forest classifier. The tree is trained on 20610 landslides samples with 10305 each earthquake and rainfall trigger class. Feature vector ($[A, P, C_h, W, s_m, \frac{A}{P}, e]$) represents landslide geometric property corresponding to each landslide sample. For illustration purposes, the tree is grown to only depth three. (b) testing sample of landslide tested on the tree shown in (a). The sample landslide polygon is classified as an earthquake.

The steps for constructing trees in the random forest are as follows:

(i) Select bootstrap samples of size N from training samples of size N.

(ii) Randomly select m variables among p variables for the node split.

(iii) Choose one variable among m variables that best split the node according to the Gini-index criterion.

(iv) Continue repeating steps (i) to (iii) until the stopping criterion is met.

For testing, each tree classifier predicts the class of testing sample independently, and the class with majority votes is the final outcome of random forest (Kuhn and Johnson, 2013; Pal, 2005; Arabameri et al., 2021; Belgiu and Drăguţ, 2016).

In random forest, bootstrapping training samples selection and random selection of features for a node split reduces the correlation between trees. This technique has proven to improve the predictive power of ensemble learning (Azar et al., 2014). In addition, random forest assigns each feature a score that provides its relative importance (Qi, 2012; Friedman et al., 2001). Features with low relative scores should be discarded as they are neutral to the model accuracy and increase the model complexity.

S2. Details of `Landsifier` library

`Landsifier` is a Python library we built with version 3.6 of Python and the code is available on GitHub: `https://github.com/kamalrana7843/Landsifier.git` (we published the Landsifier library under an open-source license in a way that it is accessible via the python terminal using the import command). On this link, prospective users can also find the list of Python packages used in the library. `Landsifier` contains three methods for landslide trigger classification, and these methods only use shapefiles from landslide inventories (two of these methods use 2D polygon shapes of landslides, while the third method uses the 3D shapes of landslides). This section describes various functions provided in the `Landsifier` library to implement the above methods and Figure S2 summarizes these functions in form of a flowchart. Also, Figure S3 shows a sample output of the `Landsifier`.

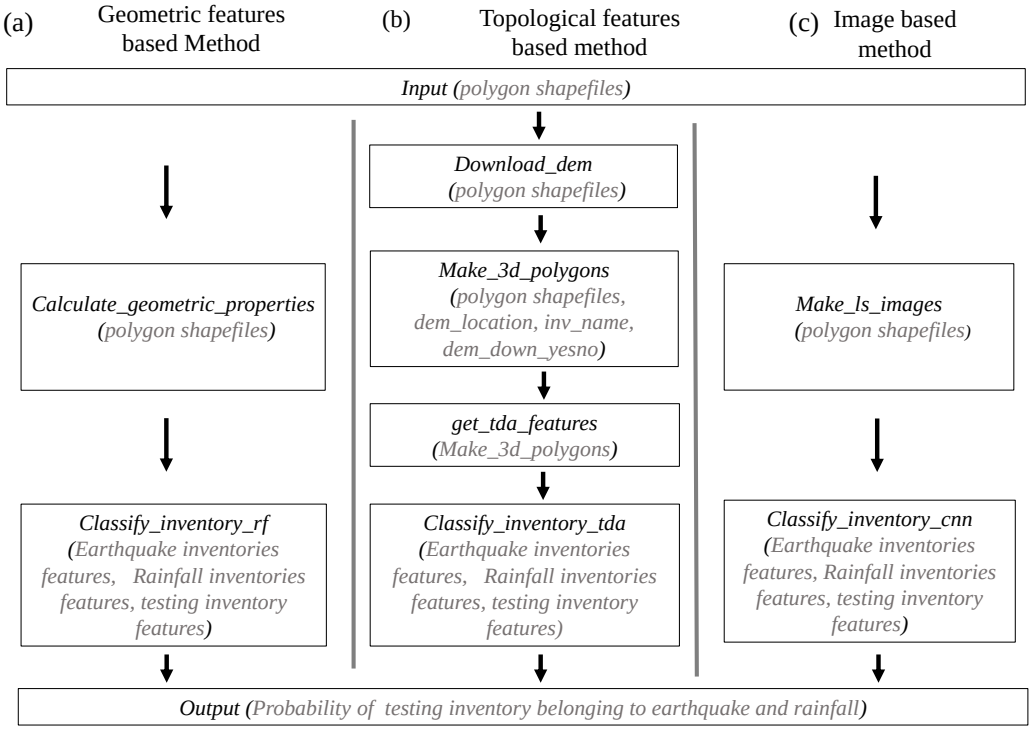

Figure S2. The figure shows the flowchart of implementations of all the three methods using functions and their variables used in the `landsifier` library. All three models use polygon shapefiles as an input to the model and provide the probability of landslide belonging to earthquake and rainfall as an output (a) geometric features based method (b) topological features based method (c) image-based method.

S2..1 Functions for geometric features based classification

Below we list functions to implement the geometric features-based classification, details of the method can be found above in section 3.1 of the main paper and in our publication (Rana et al., 2021). Note below we have described functions in a form that

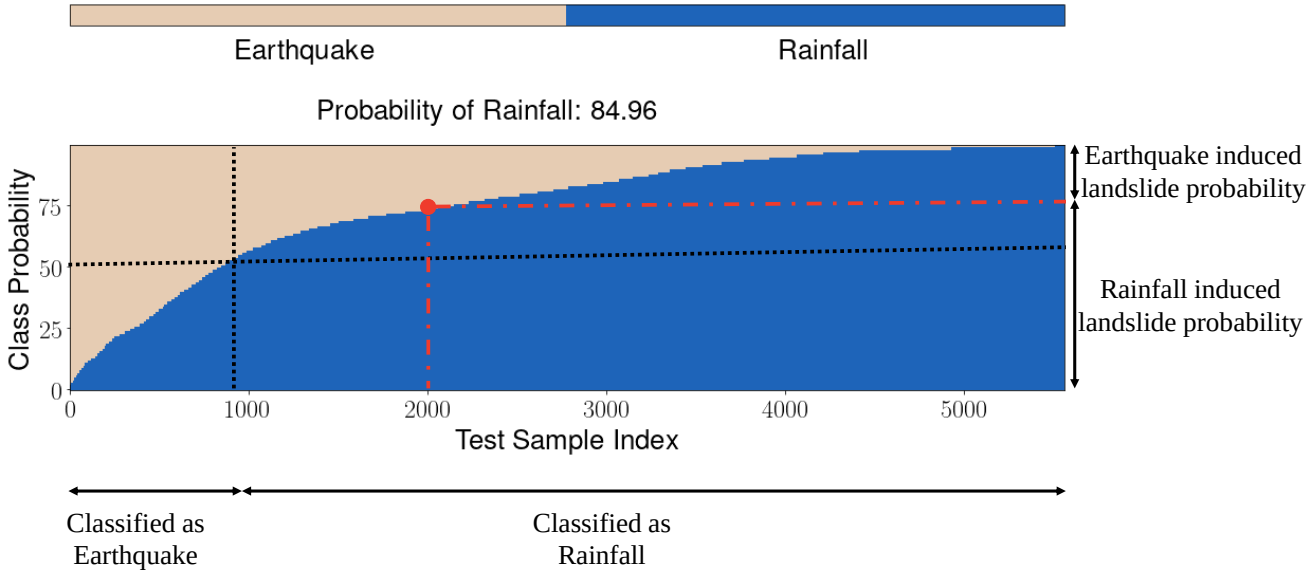

Figure S3. The output of the geometric feature-based method when Kumamoto inventory is testing inventory and the rest five inventories are used as training inventories. Each method in the `landsifier` will produce similar outputs. The y-axis in the plot shows the probability of landslides belonging to the earthquake and rainfall class, and the x-axis shows the sample index of landslides. For each landslide in the testing inventory, all the models give a probability of landslides belonging to earthquake and rainfall-induced classes. The predicted trigger of most of the testing landslides is the probable trigger of the testing inventory.

this method can be used for inventories with unspecified triggers, i.e., unknown ground truth.

`latlon_to_eastnorth(latlon_polydata)`: This function takes polygons data in $(longitude, latitude)$ coordinates as an input and provides polygon data in $(easting, northing)$ coordinates as output to the function. This function is used to get landslide polygons in $(easting, northing)$ coordinates when polygon data in shapely files are in $(longitude, latitude)$ coordinates.

`Calculate_geometric_properties (polygon_shapefile)`: As the name suggests, this function calculates the geometric properties of each of the landslide polygons present in shapefile. This function takes polygons shapefiles (`polygon_shapefile`) as input, converts polygon data into $(easting, northing)$ coordinates if required using the `latlon_to_eastnorth` function, and then provides the geometric properties of polygons as output to the function. For each landslide polygon it calculates a vector($[A, P, C_h, W, s_m, \frac{A}{P}, e]$) containing polygon geometric properties as output to the

function. All the geometric properties of the landslide polygon are calculated using the *shapely* package in Python.

`classify_inventory_rf (earthquake_inventory_features, rainfall_inventory_features, test_inventory_features)` : This function takes the earthquake-triggered inventories ($earthquake\_inventory\_features$), rainfall-induced inventories ($rainfall\_inventory\_features$) and testing inventories ($test\_inventory\_features$) geometric features as the input. Within the function, it trains the random forest algorithm on training data containing equal samples of the earthquake and rainfall-induced class. The output of the function is the probability of testing landslides belonging to each trigger class.

S2..2 Functions for topological features based classification

Below we list functions to implement the topological features-based classification, details of the method can be found above in section 3.2 of the main paper. Note below we have described functions in a form that this method can be used for inventories with unspecified triggers, i.e., unknown ground truth.

`download_dem (polygon_shapefile)`: This function takes shapefile of landslide polygons as an input and downloads the *Shuttle Radar Topography Mission* digital elevation model (DEM) of resolution 30 meters corresponding to inventory region (Farr et al., 2007). It takes the bounding box over the entire inventory location and calculates the `(minimum latitude, minimum longitude)` and `(maximum latitude, maximum longitude)`. Using the elevation package in Python, it downloads the DEM data of a region bounded by `minimum latitude, minimum longitude, maximum latitude, and maximum longitude` coordinates. This function downloads all the tiles (one tile constitutes $1^\circ \times 1^\circ$ region in both latitude and longitude) of the inventory region and combines all the tiles into one file corresponding to one inventory.

`make_3d_polygons (polygon_shapefile, dem_location, inv_name, dem_down_yesno )`: This function takes landslide polygon shapefiles (`polygon_shapefile`), DEM path location (`dem_location`), inventory name (`inv_name`) and Boolean parameter (`dem_down_yesno`) as input and provides 3D point cloud data of landslides as output. This function carries out several tasks. First, it downloads the DEM data corresponding to the whole inventory region in path location (`dem_location`) with inventory name (`inv_name`) using `download_dem` function if `dem_down_yesno` is True. If users already have DEM corresponding to inventory in path location (`dem_location`) with inventory name (`inv_name`) then (`dem_down_yesno`) is False. Then corresponding to each landslide polygon it interpolates the DEM data around the bounding box of the polygon. Using the shapely package, the function removes all the interpolated data outside the outline of the landslide polygon and takes elevation data only within the landslide.

`get_tda_features (three_d_data)`: This function takes the 3D shape of landslides point cloud data (`three_d_data`) as an input and provides machine learning features corresponding to each 3D landslides as an output to function. This func-

tion calculates the persistence diagram using Vietoris Rips persistence for each 3D landslide, and then using the persistence diagram, it calculates the following TDA metrics: persistence entropy, average lifetime, number of points, betti curve based measure, persistence landscape curve based measure, Wasserstein amplitude, Bottleneck amplitude, Heat kernel-based measure, and landscape image-based measure corresponding to each homology dimension–0, 1, and 2. These TDA metrics are used as a feature space for the machine learning algorithms.

`classify_inventory_tda (earthquake_inventory_features, rainfall_inventory_features, test_inventory_features)` : This function takes training earthquake inventory's `(earthquake_inventory_features)`, training rainfall inventory's `(rainfall_topological_features)` and testing inventory's `(test_inventory_features)` TDA based features as input to function. Inside the function, it first selects the top 10 features with the highest feature importance using training data. It then combines an equal number of training earthquake and rainfall samples to avoid any class imbalance problem. It trains the random forest algorithm on training data and predicts the probability of testing landslides belonging to each trigger class.

S2..3 Functions for image based classification

Below we list functions to implement the image-based classification, details of the method can be found above in section 3.3 of the main paper. Note below we have described functions in a form that this method can be used for inventories with unspecified triggers, i.e., unknown ground truth.

`increase_resolution_polygon (poly_data)` : This function takes a single polygon coordinates data (`poly_data`) in $(easting, northing)$ coordinates as input and increases the number of points between any two adjacent vertexes of the polygon within the function. This function is useful in creating smooth binary scale landslide polygon images. The output of the function is landslide polygons coordinates data having multiple points between the adjacent vertex of polygons.

`make_ls_images (polygon_shapefile)`: This function takes polygon shapefiles (`polygon_shapefile`) as an input and provides landslide polygon images as an output to the function. It creates $N \times N$ ($N = 64$ in our case) pixel image with binary values of 0 or 255 for each pixel. This function first increase the number of data points in polygons using `increase_resolution_polygon` function and then takes a bounding box of polygon and transforms the coordinates of polygons by subtracting polygon $(minimum\_easting, minimum\_northing)$ value from each point in the polygon. Then divide each point in polygon $(easting, northing)$ value by resolution of pixels (desired spatial distance between any two adjacent horizontally or vertically pixels) and convert them into nearest integers. Then for each pixel (x, y) the value of the pixel is 255 if there exists a point in the polygon with coordinates (x, y) otherwise the value of the pixel is 0. This function also removes those landslides having length and width of bounding box greater than 180 meters as the image of a polygon has some

restrictions on maximum landslide polygon it can have (resolution of pixels (3 meters) $\times$ N = 192 meters).

`train_augment (train_data, train_label)`: This function takes input training landslides data `(train_data)` and training labels `(train_label)` as input. The main idea behind using `train_augment` function is to augment the training data as CNN is data extensive algorithm. It rotates each image by 90°, 180°, 270°and flip the image vertically and horizontally to increase the number of training samples. The output of the function is augmented training data and labels.

`classify_inventory_cnn (earthquake_inventory_images, rainfall_inventory_images, test_inventory_images)`: This function takes training earthquake inventory images `(earthquake_inventory_images)`, training rainfall inventory images `(rainfall_inventory_images)` and testing inventory images `(test_inventory_images)` as input to the function. Within the function, it combines an equal number of training earthquake and rainfall samples to avoid any class imbalance problem and then augments the training data by using the `train_augment` function. Then it trains the CNN algorithm on augmented training data and predicts the probability of testing landslides belonging to each of the trigger classes.

Code availability. The source code and future updates are available in the GitHub repository (`https://github.com/kamalrana7843/Landsifier.git`).

Data availability. The landslide inventories used in this paper are publicly available by Geospatial Information Authority (GSI) and the National Research Institute for Earth Science and Disaster Resilience (NIED). The 30 meters SRTM DEM data used is also publicly available by NASA and downloadable via (`https://www2.jpl.nasa.gov/srtm/`).

Author contributions. All authors contributed to the writing and reviewing of the manuscript. KR developed the code. NM and UO interpreted the results and supervised the work.

Competing interests. The authors declare that they have no conflict of interest.

Acknowledgements. This project is supported by Co-PREPARE project (No: 57553291) by German Academic Exchange Service (DAAD). KR acknowledges support from RIT's Steven M. Wear Endowed graduate fellowship and NM acknowledges support through RIT's FEAD grant. UO acknowledges funding from the Research Focus Point "Earth and Environmental Systems" of the University of Potsdam.

References

Arabameri, A., Chandra Pal, S., Rezaie, F., Chakrabortty, R., Saha, A., Blaschke, T., Di Napoli, M., Ghorbanzadeh, O., and Thi Ngo, P. T.: Decision tree based ensemble machine learning approaches for landslide susceptibility mapping, Geocarto International, pp. 1–35, 2021.

Azar, A. T., Elshazly, H. I., Hassanien, A. E., and Elkorany, A. M.: A random forest classifier for lymph diseases, Computer methods and programs in biomedicine, 113, 465–473, 2014.

Barnett, I., Malik, N., Kuijjer, M. L., Mucha, P. J., and Onnela, J.-P.: EndNote: Feature-based classification of networks, Network Science, 7, 438–444, 2019.

Belgiu, M. and Drăguţ, L.: Random forest in remote sensing: A review of applications and future directions, ISPRS journal of photogrammetry and remote sensing, 114, 24–31, 2016.

Biau, G.: Analysis of a random forests model, The Journal of Machine Learning Research, 13, 1063–1095, 2012.

Biau, G. and Scornet, E.: A random forest guided tour, Test, 25, 197–227, 2016.

Breiman, L.: Random forests, Machine learning, 45, 5–32, 2001.

Chaudhary, A., Kolhe, S., and Kamal, R.: An improved random forest classifier for multi-class classification, Information Processing in Agriculture, 3, 215–222, 2016.

Farr, T. G., Rosen, P. A., Caro, E., Crippen, R., Duren, R., Hensley, S., Kobrick, M., Paller, M., Rodriguez, E., Roth, L., et al.: The shuttle radar topography mission, Reviews of geophysics, 45, 2007.

Friedman, J., Hastie, T., and Tibshirani, R.: The elements of statistical learning, vol. 1, Springer series in statistics New York, 2001.

Kuhn, M. and Johnson, K.: Applied predictive modeling, vol. 26, Springer, 2013.

Kuhn, M., Johnson, K., et al.: Applied predictive modeling, vol. 26, Springer, 2013.

Kursa, M. B.: Robustness of Random Forest-based gene selection methods, BMC bioinformatics, 15, 8, 2014.

Liaw, A., Wiener, M., et al.: Classification and regression by randomForest, R news, 2, 18–22, 2002.

Okun, O. and Priisalu, H.: Random forest for gene expression based cancer classification: overlooked issues, in: Iberian conference on pattern recognition and image analysis, pp. 483–490, Springer, 2007.

Pal, M.: Random forest classifier for remote sensing classification, International journal of remote sensing, 26, 217–222, 2005.

Qi, Y.: Random forest for bioinformatics, in: Ensemble machine learning, pp. 307–323, Springer, 2012.

Rana, K., Ozturk, U., and Malik, N.: Landslide Geometry Reveals its Trigger, Geophysical Research Letters, 48, e2020GL090 848, 2021.

Rodriguez-Galiano, V. F., Ghimire, B., Rogan, J., Chica-Olmo, M., and Rigol-Sanchez, J. P.: An assessment of the effectiveness of a random forest classifier for land-cover classification, ISPRS Journal of Photogrammetry and Remote Sensing, 67, 93–104, 2012.

Zhang, C. and Ma, Y.: Ensemble machine learning: methods and applications, Springer, 2012.