

Susceptibility and hazard mapping framework SHIRE – User Manual

Version 1.0, July 2024

[Ann-Kathrin Edrich*](#), [Anil Yildiz](#), [Julia Kowalski](#)

Methods for Model-based Development in Computational Engineering,
RWTH Aachen University

* Corresponding author (edrich@mbd.rwth-aachen.de)

Abstract

This manual describes the capabilities and usage of the susceptibility and hazard mapping framework SHIRE for landslide susceptibility and hazard mapping. The document focuses on the application and individual steps necessary to run SHIRE. For more information on the background and technical implementation refer to the publications below, specifically the associated PhD Thesis. If questions remain, please write to the corresponding author. The framework was developed and tested on a MacBook Pro, 2.3 GHz Quad-Core Intel Core i7 and 16 GB RAM using Python version 3.7.11.

This manual is an addition to the following publications:

- Edrich, A. K., Yildiz, A., Roscher, R., Bast, A., Graf, F., & Kowalski, J. (2024). A modular framework for FAIR shallow landslide susceptibility mapping based on machine learning. *Natural Hazards*, 1-30.

To be published:

- Edrich, A. K., Yildiz, A., Roscher, R., & Kowalski, J. (2024). Incorporation of time-dependent precipitation information into Random Forest-based landslide hazard mapping for early warning.
- Edrich, A. (2024) Machine learning based landslide susceptibility assessment: Challenges of study design and opportunities for early warning [Thesis]

Contents

| | | |
|----------|--------------------------------------|-----------|
| 1 | SHIRE | 3 |
| 2 | Initial installation | 4 |
| 3 | Preparations | 4 |
| 3.1 | GUI and Plain version | 4 |
| 3.1.1 | Data collection | 4 |
| 3.1.2 | Absence locations sampling | 4 |
| 3.1.3 | Keys-file | 4 |
| 3.1.4 | Data summary-file | 4 |
| 3.2 | Plain version-specific | 5 |
| 3.2.1 | Settings-file | 5 |
| 4 | Running SHIRE | 8 |
| 4.1 | GUI version | 8 |
| 4.2 | Plain version | 11 |
| 4.3 | Checking user input | 11 |
| 5 | Output files | 12 |

| | | |
|----------|--|-----------|
| 6 | Validation | 13 |
| 6.1 | Validation and Quality control included in SHIRE | 13 |
| 6.1.1 | Landslide inventory and geospatial datasets | 13 |
| 6.1.2 | RF input datasets | 13 |
| 6.1.3 | Model validation | 14 |
| 6.2 | Recommended validation by user | 14 |

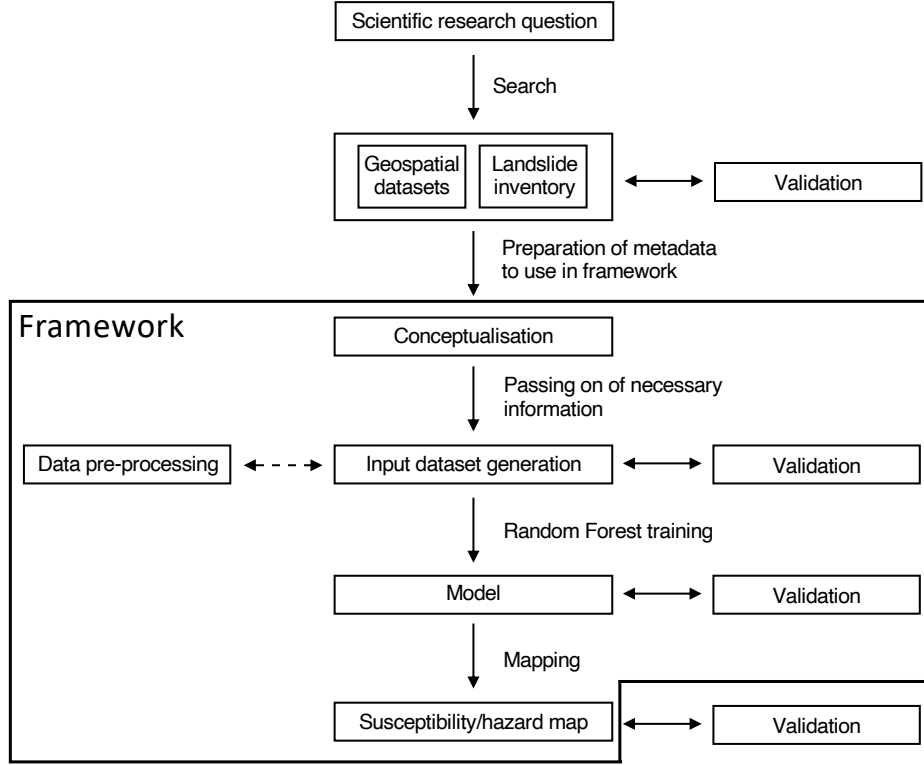


Fig. 1: Schematic flow chart of SHIRE (Edrich et al., (2024))

1 SHIRE

SHIRE (Susceptibility and **H**azard mappIng **f**RamEwork) is a Python-based framework designed to streamline and facilitate future Random Forest (RF) Classifier-based landslide susceptibility and hazard mapping (LSHM) studies without the need for secondary software like GIS. SHIRE produces binary landslide susceptibility or hazard maps. Developed as part of the [KISTE project](#), SHIRE was initially presented in the publication "[A modular framework for FAIR shallow landslide susceptibility mapping based on machine learning](#)" by Edrich et al. (2024). The framework is publicly available and openly accessible on [GitLab](#).

Figure 1 illustrates the structure of SHIRE. The framework aims to provide pre-implemented solutions for repetitive steps in LSHM. The main steps include: (1) conceptualisation, (2) RF input dataset generation, which includes data pre-processing, and (3) map generation using the RF model. Additionally, SHIRE incorporates quality control measures and outputs products that can be used for validation (see section 6.1). Users are responsible for formulating the scientific objective, collecting the necessary datasets, and performing the primary quality control of the datasets. SHIRE is explicitly not designed to relieve the user of the responsibility of ensuring the quality of the final mapping product. Caution and thorough quality control and validation are advised.

SHIRE is available in two versions: (1) GUI version and (2) Plain version. Both versions have the same capabilities, but the GUI version allows conceptualisation through a user interface, which may facilitate the setting of various variables. In the Plain version, the settings are defined in a Python script.

2 Initial installation

Clone the git repository to your local machine. It is recommended to set up a virtual environment. Then install the packages in **requirements.txt**.

3 Preparations

3.1 GUI and Plain version

3.1.1 Data collection

Geospatial datasets must be collected prior to running SHIRE and should be available locally as either TIFF or netCDF files. It is crucial to identify the no-data values for each dataset and determine whether the data is categorical or continuous. Note that SHIRE does not include an assessment of dataset quality; this should be conducted before generating a landslide susceptibility or hazard map (see section 6.2).

Additionally, a landslide inventory containing the spatial coordinates of the instances and individual IDs is required. Any extra information in the inventory will be disregarded. The inventory should be provided as a CSV file, with the column names for the ID, longitude and latitude coordinates unambiguously identifiable.

3.1.2 Absence locations sampling

Absence locations (non-landslide locations) must be sampled and provided by the user in either CSV or netCDF format. The CSV file should include two columns for the longitude and latitude values of the absence locations. The netCDF file should contain the same information, stored as two separate variables.

It is recommended to sample more absence locations than initially intended for integration into the training dataset (TD). This is because the cleaning process, which removes instances containing no-data values, might otherwise disrupt the desired ratio of presence to absence locations. Any extra absence locations will be removed in a final step, ensuring that the original inventory's excess instances do not impact the resulting TD. However, to avoid unnecessarily increasing computational demands, the number of additional absence locations should be carefully considered. Randomizing both the sampling process and the order of absence locations in the dataset is also advised to prevent introducing bias when additional instances are removed.

3.1.3 Keys-file

The **keys_to_include.csv** file specifies which features to include in the TD and/or PD generation or add/remove from an existing TD and/or PD. This CSV file contains a single column labeled *keys_to_include*, with each row listing the name of a feature. These feature names must be a subset of those listed in the *keys* column of the **data_summary.csv** file (see section 3.1.4). A template for this file is available in the Git repository.

3.1.4 Data summary-file

The **data_summary.csv** file provides an overview of the available geospatial datasets. It includes the following columns: *path*, *keys*, *no_value*, and *categorical*. Each column must be completed for SHIRE to successfully perform its tasks. A template for this file is available in the Git repository. Table 1 describes the content of the data summary file in more detail.

Table 1: Overview of parameters to provide in the **data_summary.csv** file

| Name | Content | Comment |
|-------------|--|--|
| path | Local path where the dataset is stored | Full path including file extension |
| keys | Feature name | Names shouldn't include '/' |
| no_value | No data values | If none set <i>None</i> , else comma-separated list of integers without spaces |
| categorical | True for categorical, otherwise False | Boolean |

3.2 Plain version-specific

3.2.1 Settings-file

The **settings.py** file contains the general setting for the RF input dataset generations, data pre-processing and mapping. It is the *Plain version*'s way of assembling the information gathered in the GUI presented in Section 4.1. The content of **settings.py** can be found in Table 2.

Table 2: Overview of parameters to define in the **settings.py** file

| Name | Content | Type | Comment |
|-------------------------|--|---------|--|
| training_dataset | TD generation | Boolean | Call TD generation module |
| preprocessing | Pre-processing approach | string | Choose cluster, interpolation or no_interpolation |
| train_from_scratch | Generation of TD from scratch | Boolean | True for TD generation from scratch, else False |
| train_delete | Add/delete feature(s) | Boolean | train_from_scratch=False and train_delete=True deletes feature, train_from_scratch=False and train_delete=False adds feature |
| prediction_dataset | PD generation | Boolean | Call PD generation module |
| pred_from_scratch | Generation of PD from scratch | Boolean | True for PD generation from scratch, else False |
| pred_delete | Add/delete feature(s) | Boolean | train_from_scratch=False and train_delete=True deletes feature, train_from_scratch=False and train_delete=False adds feature |
| map_generation | Mapping | Boolean | Call map generation module |
| training | Model training | Boolean | True for model training, else False |
| prediction | Map generation | Boolean | True for map generation, else False |
| crs | Coordinate reference system | String | Metadata |
| no_value | Common no data value | Integer | To replace dataset-specific no data values, e.g. -999 |
| random_seed | Random seed value | Integer | |
| resolution | Resolution of final map | Integer | in meters |
| path_ml | Path for storing parameters/files | String | |
| size | Ratio for VD | Float | Value between 0 and 1 |
| path_train | Path to store TD | String | |
| path_landslide_database | Path to landslide dataset | String | |
| ID | ID column in landslide inventory | String | |
| landslide_database_x | Name of longitudes column in landslide inventory | String | |
| landslide_database_y | Name of latitudes column in landslide inventory | String | |
| path_nonls_locations | Path to absence locations dataset | String | |
| num_nonls | Number of non-landslide locations for TD | Integer | |

| Name | Content | Type | Comment |
|-------------------------|--|-----------------|---|
| nons_database_x | Longitudes column/variable in absense location inventory | String | |
| nons_database_y | Longitudes column/variable in absense location inventory | String | |
| bounding_box | Bounding box of the AOI | List | $[y_{max}, y_{min}, x_{max}, x_{min}]$ |
| path_pred | Path to prediction dataset | String | |
| not_included_pred_data | Features in PD to drop before mapping | List of Strings | |
| not_included_train_data | Features in TD to drop before training | List of strings | |
| num_trees | Number of decision trees | Integer | see options for <i>scikit-learn's</i> RF Classifier |
| criterion | RF criterion | String | |
| depth | Number of nodes | Integer | |
| model_to_save | Name of the model for saving | String | Typically equal to model_to_load |
| model_to_load | Name of the model for loading | String | Typically equal to model_to_save |
| model_database_dir | Path to models | String | |
| parallel | Parallel prediction | Boolean | |

Mapping

4 Running SHIRE

4.1 GUI version

SHIRE can be launched by running `shire.py` and the GUI opens automatically. The python package `tkinter` which has been used for implementing the GUI is known to cause issues with certain editors. Therefore, it is recommended to run SHIRE from the command line.

The first window to open is the interface to define the general settings (see Figure 2).

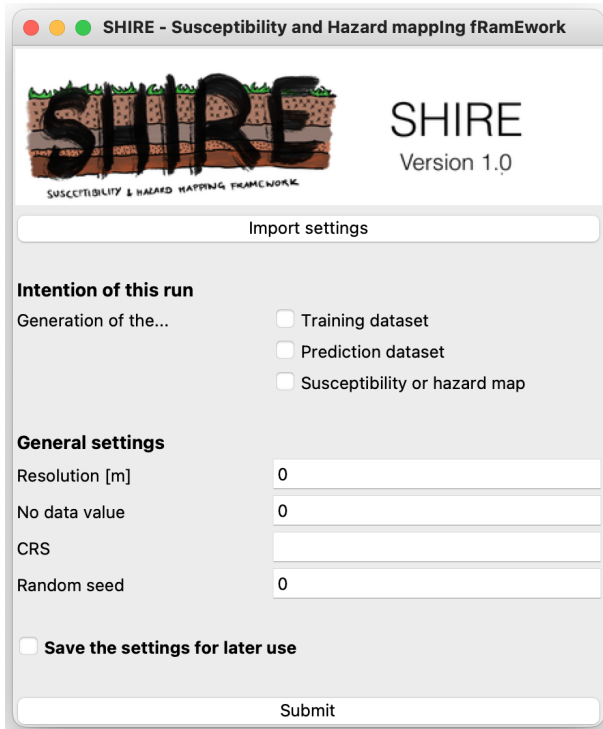


Fig. 2: General settings GUI

In this initial step, properties are set that are relevant for all modules included in SHIRE. Depending on the boxes ticked for *Intention of this run*, the next window opens. If *Training dataset* is ticked the window shown in Figure 3 opens, if *Prediction dataset* is ticked window 5 opens and if *Map generation* is ticked window 6 opens. If several are ticked, its always training dataset first, then prediction dataset and then map generation.

Import settings: Import previously saved settings

Intention of this run: Tick the boxes associated with training and prediction dataset generation or map generation according to individual demands

General settings: Add desired resolution in m (integer), No data value (integer), coordinate reference system (string) and random seed (integer)

Save the settings for later use: Tick if settings shall be saved as pickle file for later use

Submit: Submit settings to proceed. A temporary pickle file is saved which is deleted at the end

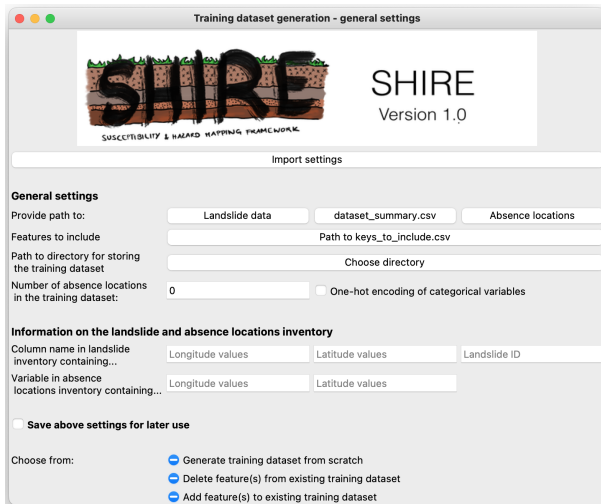


Fig. 3: Training dataset GUI

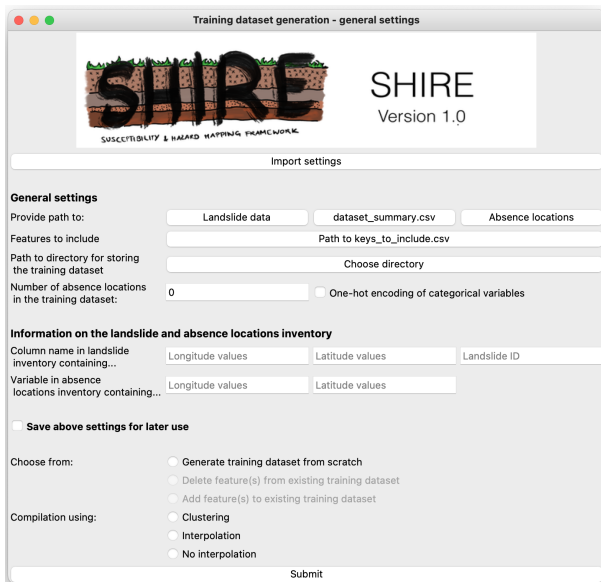


Fig. 4: Training dataset GUI if **Generate training dataset from scratch** is ticked

Import settings: Import previously saved settings

Provide path to: Provide the path to the landslide inventory, data_summary.csv file, and absence locations database

Features to include: Provide the path to the keys_to_include.csv file

Path to directory for storing the training dataset: Choose directory for storing the TD after generation

Number of absence locations in the training dataset: Provide the number of non-landslide locations (integer)

One-hot encoding of categorical variables: If ticked the categorical variables are one-hot encoded, otherwise ordinal encoding is used

Column name in landslide inventory containing...: Provide column names that contain the landslide IDs, longitude and latitude values (strings)

Variable in absence locations inventory containing...: Provide variable names of longitude and latitude values (strings)

Save the settings for later use: Tick if settings shall be saved as pickle file for later use

Choose from: Click on the goal of the run. Only one option is possible. Depending on the choice, the window extends differently

Compilation using: Click on the pre-processing step of your choice. Only one option is possible (Figure 4)

Submit: Submit settings to proceed. A temporary pickle file is saved which is deleted at the end



Fig. 5: Prediction dataset GUI

Import settings: Import previously saved settings

Path to summary of geospatial data: Provide the path to the data_summary.csv file

Features to include: Provide the path to the keys_to_include.csv file

Path to directory for storing the training dataset: Choose directory for storing the PD after generation

One-hot encoding of categorical variables: If ticked the categorical variables are one-hot encoded, otherwise ordinal encoding is used

Area of Interest: Provide bounding box coordinates of the area of interest in decimal degrees (float)

Save the settings for later use: Tick if settings shall be saved as pickle file for later use

What do you want to do?: Click on the aim of the run. Only one option is possible

Submit: Button appears after choosing the aim of the run. Submit settings to proceed. A temporary pickle file is saved which is deleted at the end of the run

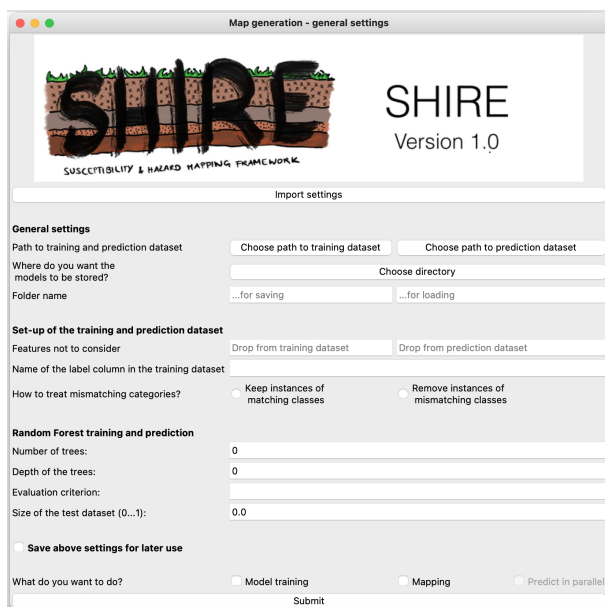


Fig. 6: Map generation GUI

Import settings: Import previously saved settings

Path to training and prediction dataset: Provide the path to the TD and PD

Where do you want the models to be stored?: Provide the path to the directory where the model shall be stored

Folder name: Provide the name of the model for saving the trained model and name of the model to load for mapping (string). Typically identical

Features not to consider: Comma-separated list of features (no spaces!) that shall be dropped from the TD before model training and PD before mapping

Name of label column in training dataset: Name column containing target variable in training dataset (string)

How to treat mismatching categories?: Choose how do handle mismatches of the one-hot encoded TD and PD

Random Forest training and prediction: Specify number of decision trees (integer), depth of the trees (integer), evaluation criterion (string) and the size of the Test dataset (float between 0 and 1)

Save the settings for later use: Tick if settings shall be saved as pickle file for later use

What do you want to do?: Choose if model training and/or mapping shall be conducted. Both options can be chosen independently (make sure to adapt **Folder name** accordingly!). If mapping is chosen, there is the option to run the RF prediction in parallel

Submit: Submit settings to proceed. A temporary pickle file is saved which is deleted at the end

After submitting the information on all desired steps, SHIRE runs the respective modules and continuously updates a log file called **shire.log**. This log can be consulted to review the specifics of a certain run or find errors and warnings in failed runs. Careful: New runs append to existing log file!

4.2 Plain version

Initialising SHIRE using the *Plain version* only requires running **shire.py**. Please make sure that **settings.py** has been adapted according to the individual needs before running the framework.

4.3 Checking user input

After SHIRE has been launched, the user input is automatically assessed for its completeness and validity. The results are stored in the **check_user_input.log** file. If any errors are detected

the run is cancelled.

5 Output files

Table 3 provides an overview of SHIRE's output files.

Table 3: Overview of SHIRE's output files. The output files are if not otherwise specified equal for GUI and Plain version

| Step | Name |
|-------------------------------|---|
| Training dataset generation | training.csv <ul style="list-style-type: none"> • Directory: user defined • Format: csv • Content: TD • Comment: - |
| | data_combined_training_<resolution>.nc <ul style="list-style-type: none"> • Directory: same as training dataset • Format: netCDF4 • Content: cropped and interpolated datasets after data pre-processing • Comment: filename depends on user defined resolution |
| | data_combined_training_<resolution>.pkl <ul style="list-style-type: none"> • Directory: same as TD • Format: pickle • Content: metadata on data pre-processing • Comment: filename depends on user defined resolution |
| Prediction dataset generation | prediction.nc <ul style="list-style-type: none"> • Directory: user defined • Format: netCDF4 • Content: PD • Comment: - |
| | data_combined_prediction_<resolution>.nc <ul style="list-style-type: none"> • Directory: same as PD • Format: netCDF4 • Content: cropped and interpolated datasets after data pre-processing • Comment: filename depends on user defined resolution |
| | data_combined_prediction_<resolution>.pkl <ul style="list-style-type: none"> • Directory: same as prediction dataset • Format: pickle • Content: metadata on cropping and interpolating • Comment: filename depends on user defined resolution |
| Model training | saved_model.pkl <ul style="list-style-type: none"> • Directory: model folder • Format: pickle • Content: RF model • Comment: - |
| | model_params.pkl <ul style="list-style-type: none"> • Directory: Model folder • Format: pickle • Content: training parameters and quality metrics • Comment: - |
| | feature_importance.csv |

| Step | Name |
|---------|--|
| Mapping | <ul style="list-style-type: none"> • Directory: model folder • Format: csv • Content: RF feature importance output • Comment: - |
| | prediction_results.csv <ul style="list-style-type: none"> • Directory: model folder • Format: csv • Content: prediction results on all individual locations within the area of interest • Comment: table format consisting of longitude, latitude and prediction |
| | pos_prediction_results.csv <ul style="list-style-type: none"> • Directory: model folder • Format: csv • Content: locations with a predicted susceptibility to landslides • Comment: table format consisting of longitude, latitude and prediction |
| | neg_prediction_results.csv <ul style="list-style-type: none"> • Directory: model folder • Format: csv • Content: locations without a predicted susceptibility to landslides • Comment: table format consisting of longitude, latitude and prediction |
| | prediction.nc <ul style="list-style-type: none"> • Directory: model folder • Format: netCDF4 • Content: susceptibility or hazard map • Comment: - |
| | |
| | |
| | |
| | |
| | |
| | |

6 Validation

Figure 1 shows that validation and quality control take place throughout the whole mapping pipeline to ensure the reliability of the final product. Some validation steps are included in SHIRE, others, that are domain-specific, require scientific knowledge or are necessary to conduct before running the framework. In the following, first the validation and quality control measures included in SHIRE are presented and afterwards recommendations are listed for validation procedures to be conducted by the user.

6.1 Validation and Quality control included in SHIRE

6.1.1 Landslide inventory and geospatial datasets

In the beginning of the TD generation process, the landslide inventory is investigated for missing or nan values which are consequently removed. There is no quality control of the geospatial datasets included in SHIRE.

6.1.2 RF input datasets

The RF input datasets need to be cleaned from no data instances. SHIRE scans the input datasets for these instances and removes them. This though might obscure the ratio of presence to absence locations which has been pre-determined by the user before running SHIRE. To ensure that the ratio, respectively the number of absence locations remains the same, SHIRE removes or adds absence locations.

SHIRE first checks whether a 1:1 ratio is intended by the user, meaning the number of absence locations provided in **settings.py** or through the GUI is identical to the number of presence locations before cleaning. In this case additional absence locations that are included in the TD are removed to ensure the same number of absence as presence locations. This thereby takes into consideration also if presence locations were removed due to no data values. If not enough absence locations were sampled or too many removed due to no data values, SHIRE produces an error.

If a different ratio of presence to absence locations is desired, the number of absence locations is reduced to the amount specified in **settings.py** or through the GUI.

6.1.3 Model validation

One of the most common model validation approaches is the evaluation of the test dataset. SHIRE computes the mean squared error, the mean absolute error, F1 and F2 score as well as the ROC curve and stores them in the **model_params.pkl** file. The assessment of the sufficiency of the accuracy is up to the user.

Scientific consistency is another way of identifying model flaws. The RF inherently computes the feature importance which can be evaluated for its consistency with domain knowledge. The export of the feature importance is integrated in SHIRE. The assessment of the feature importance, as the assessment depends e.g. on the integrated features and area of interest, and therefore has to be performed manually.

6.2 Recommended validation by user

Quality of input datasets

The user is responsible to ensure the quality of the geospatial datasets and landslide inventory.

Scientific consistency

Using the **feature_importance.csv** file which contains the feature importance ranking of the RF model. This ranking has to be evaluated manually for its scientific consistency. Missing scientific consistency is an indication for model flaws. Please be aware that the feature importance ranking for strongly correlated features is not reliable. See [Edrich et al. \(2024\)](#) for an example or a manual feature importance assessment. Visual assessment of the resulting map for flaws and contradictions with domain knowledge can provide further trust in the final product.

References

Edrich AK, Yildiz A, Roscher R, Kowalski J (2024) A modular framework for FAIR shallow landslide susceptibility mapping based on machine learning. Natural Hazards <https://doi.org/10.1007/s11069-024-06563-8>