Figure 1: GRDC Pilot App

Machine Learning for Mapping Soil Constraints in 3D: GRDC Pilot App

This is a GRDC pilot App for machine learning to transform sparse soil measurements and surface measurements into 3D cube prediction of soil properties and their uncertainties. One of the key features is the probabilistic 3D modeling (cubing), which is data-driven approach and performed via Gaussian Process Priors with a spatial 3D kernel plus multiple mean functions to take into account a diverse range of additional covariates (e.g., from satellite data, surface or climate measurements). The App is build with Python (see Mlsoil) and includes multiple options via a graphical user interface (GUI) or as a settings file. The four main functionalities are: - Feature importance calculation - Model evaluation and ranking - Soil predictions in 3D including prediction uncertainties - Available water capacity predictions based on soil constraints

Author: Sebastian Haan (Sydney Informatics Hub at The University of Sydney)

Example I/O

Input: sparse soil measurements at irregular and uncertain depth plus surface measurements of multiple covariates (DEM, Rain, NDVI, Soiltype, Slope etc.)

Output: Soil properties (e.g., ESP, pH, EC) and their uncertainties as 3D cubes.

Table of Contents

Functionality

The core features are:

- Probabilistic prediction of 3D soil distributions based on sparse measurements
  - Prediction of spatial covariance via Gaussian Process Regression (GPR) with sparse 3D kernels
  - Predictions possible at any scale and resolution
  - Generation of uncertainty maps for each depth
  - Multiple options for covariate-dependent mean function of GP:
    - Power-Transformed Bayesian Linear Regression (BLR+GP)
    - Bayesian Neural Networks (BNN+GP)
    - Random Forest (RF+GP)
  - Input Uncertainties taken into account:
    - estimation of measurement uncertainties
    - uncertainties of measurement position (e.g. depth intervals)
  - Global GP hyperparameter optimisation
- Automatic training and evaluation of multiple models (BLR+GP, BNN+GP, RF+GP)
  - 10-fold cross-validation
  - residual error analysis
  - evaluation: RMSE, R squared, predicted uncertainty accuracy
  - ranking of models based on RMSE
- Feature Importance Calculation
  - Significance of Bayesian Linear Regression Coefficients
  - Random Forest Permutation test
- Prediction of soil constraints and available water capacity (AWC)
- Multiple output formats:
  - image maps
  - csv tables
  - geolocation-referenced tif
- Optional support for prediction over volume rather than point predictions
- Graphical User Interface (GUI) as well as support for python scripting
- Available as one-click executable App for Windows 10 and MacOS

Installation And Requirements

Executables for Windows and MacOS

The App is available as executable windows 10 (`Dig3D.exe`) and MacOS app (`Dig3D.app`). For MacOS please download Dig3d_App_MacOS, unzip the folder and click `Dig3D` in folder MacOS. This will open the GUI to select options and to run the App.

It is also recommended to download the example data (project_example_Uah) in the same folder as the App. This will allow you to test the app on some example field data and to check the required input format specifications for soil data.

Python Installation

MLsoil requires Python 3 (tested with Python 3.7)

For installation it is recommended to setup a virtual environment. This can be done, e.g., via conda: (replace ENV_NAME with your environment name)

```
conda create --name ENV_NAME python=3.7.11
```

for windows add env name to path:

```
set PATH=C:\Anaconda\envs\ENV_NAME\Scripts;C:\Anaconda\envs\ENV_NAME;%PATH%
```

activate environment:

```
conda activate ENV_NAME
```

and install packages:

```
conda env update -n ENV_NAME --file environment_OS.yaml
```

To deactivate environment use `conda deactivate`.

Alternatively use virtualenv, e.g. on a mac, to enable python framework for the GUI:

```
pip install virtualenv
env PYTHON_CONFIGURE_OPTS="--enable-framework" python3.7 -m venv ENV_NAME
source ENV_NAME/bin/activate
pip install -r requirements_pipfreeze.txt
```

The dependencies are defined in the requirement files and are included in the repository (see `environment_macOS.yaml`, `environment_win10.yaml`, `requirements_pipfreeze_windows10.txt`)

For custom installation, the main required packages are:

- matplotlib
- seaborn
- scipy
- pandas
- numpy
- scikit-learn
- tqdm
- pyyaml
- pyvista (for 3D visualisation)
- wxpython (for gui)
- gooey (for gui)

Required geospatial libaries:

- geopandas>=0.7.0

- rasterio
- shapely
- geos
- gdal

For bayesian neural network:

- tensorflow
- tensorflow_probability

GETTING STARTED

The easiest way to get the App running and to test its functionalities is to start the App via the executable `Dig3D.exe` or `Dig3D.app` (MacOS). This will open a GUI window with some predefined settings (here for the sample data Uah as demonstration). To familiarize yourself with the settings and input data specifications, it is advisable to check the sample data and test multiple run options.

Manual via Concol and Python

Once the software package and dependencies are installed, you can launch the GUI via python:

```
python MLsoil/run_gui.py
```

or run the App via settings file:

1) Specify settings and filenames for input data in settings yaml file (see example settings file in repository.)
2) run script run_mlsoil.py in folder MLsoil with command line argument of name of settings yaml file, e.g.:

```
python run_mlsoil.py NAME_OF_SETTINGS_FILE.yaml
```

Options and Customization

Target Soil Properties

The app has been tested given the following soil measurements (see soil example data for Uah): - Sodicity, i.e., Exchangeable Sodium Percentage (ESP) - pH value (pH) - Electrical conductivity (EC)
Given that the underlying machine learning model is a pure data-driven approach, no prior knowledge is required as input. Thus, in principle any soil property can be modeled if measurements are provided.

Feature Importance Tests

The importance and significance of factors (here covariates) can be used for, e.g., feature selection, and multiple methods exists to determine their importance. This App applies two methods for estimating feature importance: 1) Bayesian Linear Regression via significance given by the ratio of correlation coefficient divided by its standard deviation, 2) Random Forest permutation test (note that permutation test has multiple advantages over Random Forest impurity-based feature importance. However, permutation and impurity test assume that features are not correlated with each other). The features significance is presented in a ranked bar chart (see section Output).

Model Testing via Crossvalidation

Multiple models are automatically tested on the input soil data and evaluated in terms of their performance (see for more details section Output). The residual analysis is performed on the validation data with a split in train/validation ratio of 90/10 and 10 cross validations. The residual errors defined

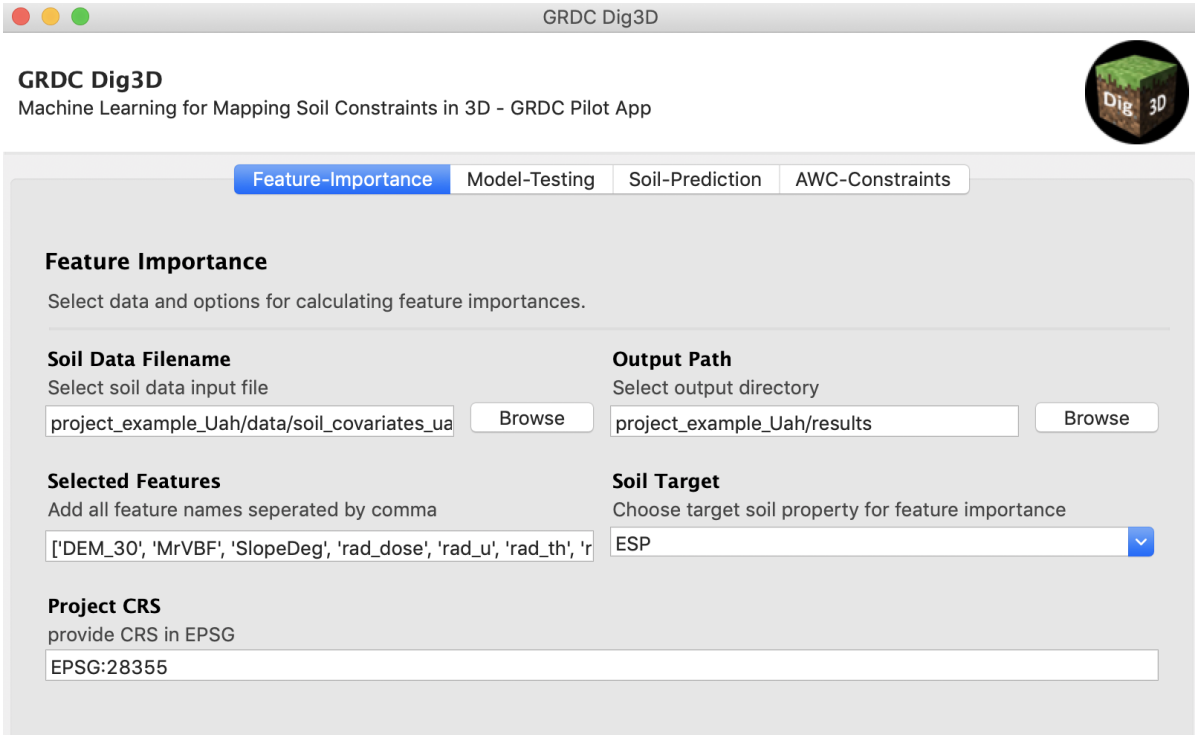


Figure 2: Feature Importance Configuration

as the predicted value minus the ground truth for each data point in the validation set. The lower the Root-Mean-Square-Error (RMSE) value, the better is the prediction. To test whether the predicted uncertainties are consistent with the residual error of the prediction, we calculate theta which is the ratio of the residual squared divided by the predicted variance. The following three mean function models are included in conjunction to the Gaussian Process.

Mean Model Functions

- Bayesian Linear Regression: Before performing linear regression, the App standardizes the data and applies a feature-wise power transform scaler via scikit-learn implementation. Power transforms are a family of parametric, monotonic transformations that are applied to make data more like normal distributed. This is useful for modeling issues related to heteroscedasticity (non-constant variance), or other situations where normality is desired. In detail, the Yeo-Johnson transform [@Yeo:2000] is applied, which support both positive or negative data. After the feature-wise scaling of data a first Bayesian Ridge regression is performed using scikit learn implementation `BayesianRidge`. The results of the coefficients and their uncertainty are used to select only significant features (with the ratio correlation coefficient divided by standard deviation larger than one). Then a second Bayesian Ridge regression is made using only the selected features and the final model and coefficients are stored, with non-significant coefficients set to zero. For more implementation details see `blr.py`.
- Probabilistic Neural Network: The probabilistic neural network is implemented by building a custom tensorflow probability model with automatic feature selection for sparsity. For implementation details see `bnn.py`. This method requires a feature-wise standard scaler.
- Random Forest: The scikit-learn Random Forest model implementation is applied. No data scaler required. Prediction uncertainties are currently estimates by using the standard deviation and

Confidence Intervals of all decision trees. For more implementation details and hyper parameter settings see `rf.py`.

Soil Predictions

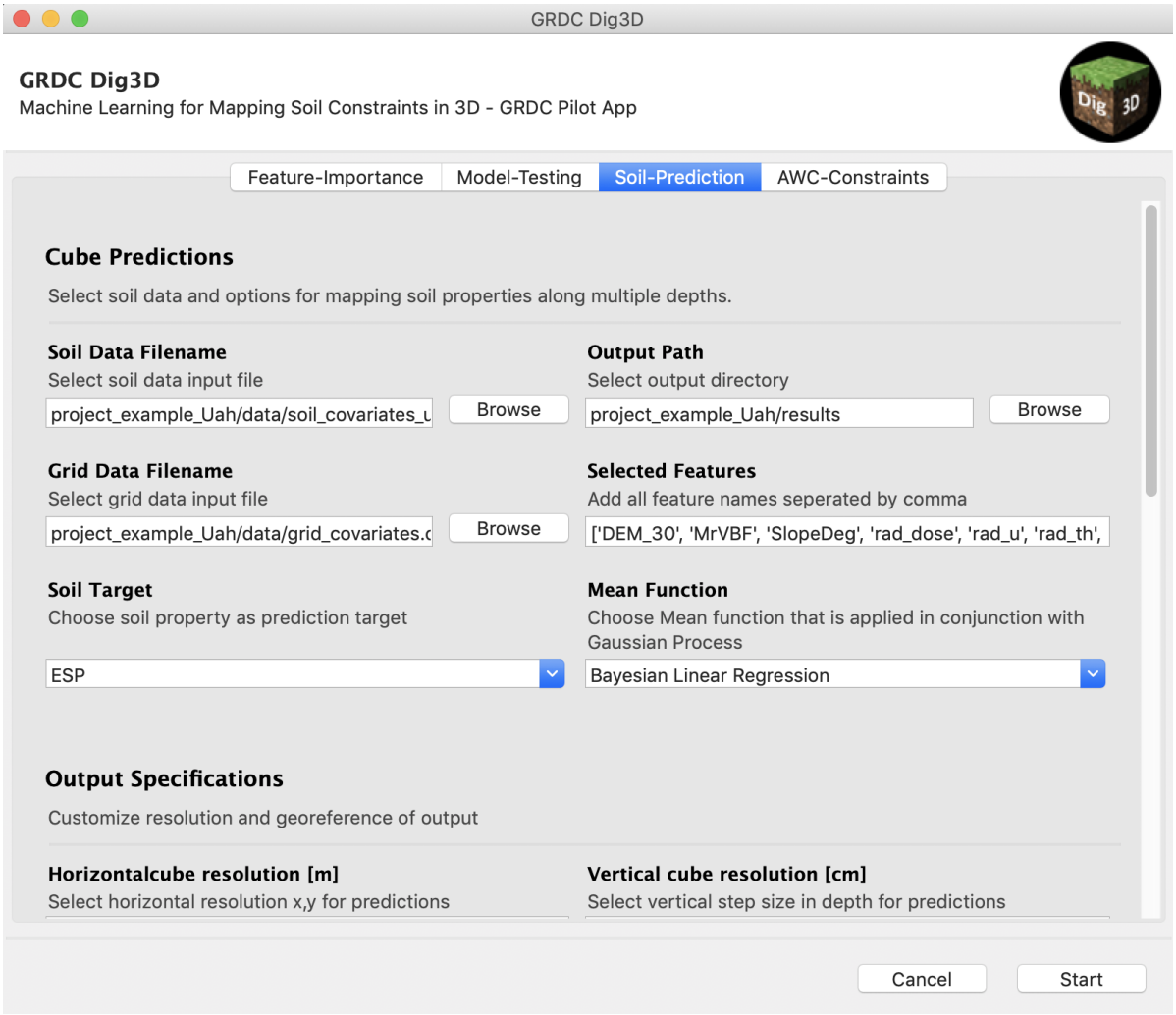


Figure 3: Cube generation Configuration

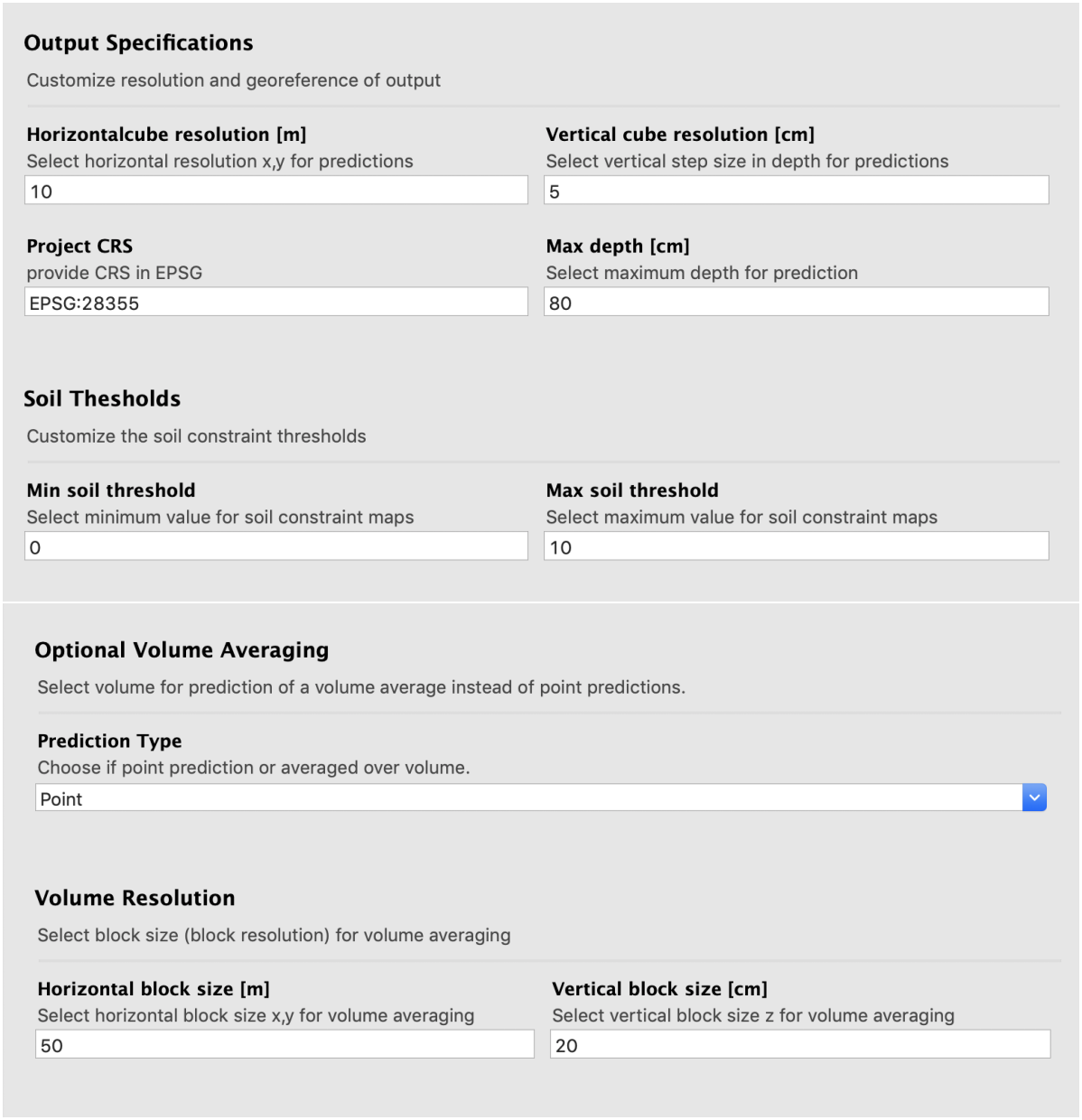
Predictions and their uncertainties are generated at any selected resolution (ideally should match resolution of the covariate grid). The required input files are: - soil measurement file including the target soil property (ESP, ph, or EC), xy coordinates, lower and upper depth interval, and other covariates such as DEM, NDVI, radiation, EM measurements) - the grid data file that includes the covariates for the prediction The selected features should match the feature names in the soil and grid data. An example of the data files and their specification is provided in the folder `project_example_Uah`.

The user can select for prediction between the three models (BLR+GP, BNN+GP, RF+GP), which can be based on certain preferences (default BLR+GP) or the result of the model testing (see previous section).

The user can select the minimum and maximum threshold for soil properties (see options `Soil Thresholds`), which creates the corresponding depth constraint map for the predicted soil proper-

ties.

If predictions should be averaged over a certain volume (e.g., to reduce prediction uncertainty) rather than point predictions (default setting), the user can choose in the settings `Optional Volume Averaging` the prediction type `Volume` and the corresponding size of the volume block (horizontal and vertical resolution) for averaging. The volume averaging method takes into account spatial covariance between points for prediction of the mean and uncertainty.



AWC Constraints

The estimation of the available water capacity (AWC) integrated over depth is based on the depth constrain maps for the corresponding soil target which is part of the soil prediction output (see previous section). The required input files are: - AWC Data: a cube of the available water in .tif format, where each band represents the available water capacity at 1 cm depth intervals - The depth constrain maps

which are generated as part of the soil predictions.

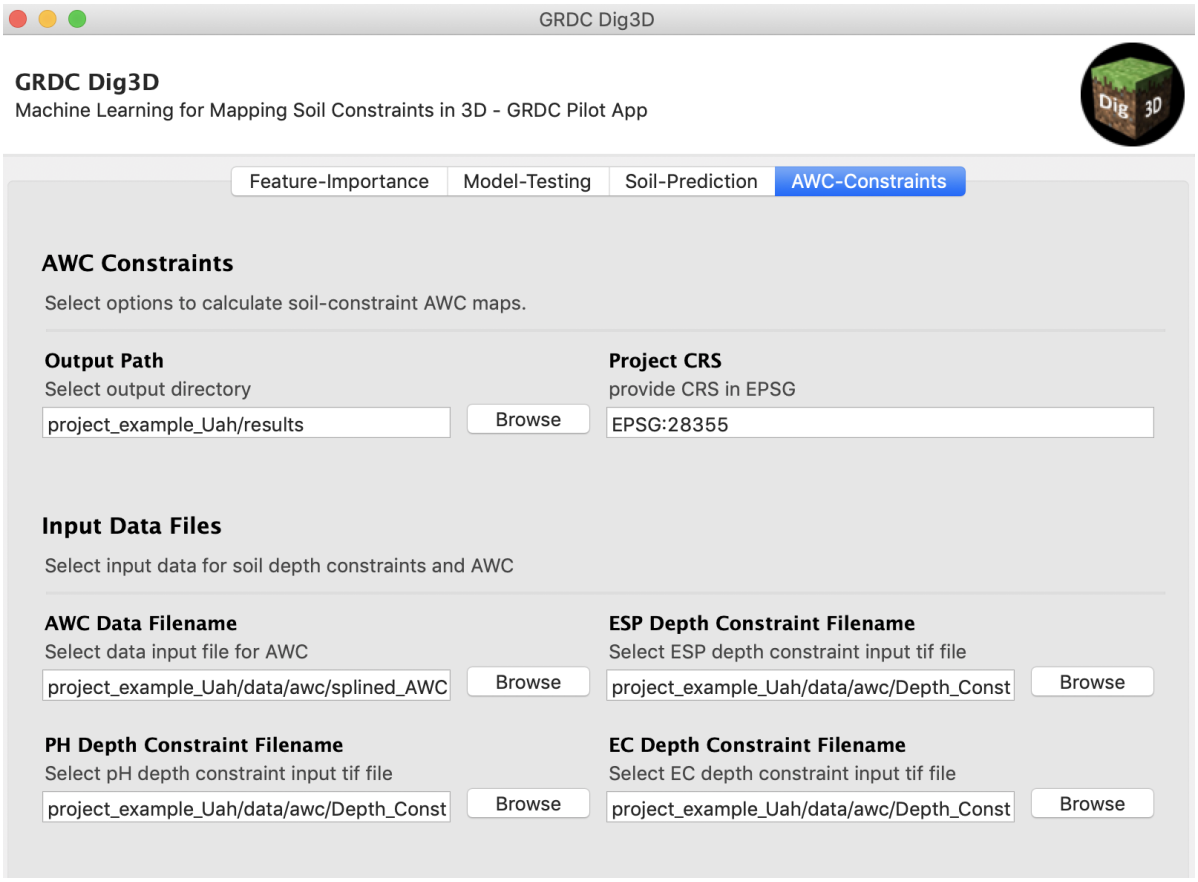


Figure 4: Volume Averaging Option

Results and Output Files

In this use case scenario, the following soil measurements are used (see data in `project_example_Uah`):

- soil measurements of ESP, pH, and EC at positions x,y (Easting, Northing) and upper and lower depth, including a range of surface covariates extracted for the x,y positions (e.g. DEM_30, 'SlopeDeg', 'rad_dose', 'rad_k', 'NDVI_5', 'NDVI_50', 'NDVI_95', 'EM_top','EM_sub', 'RED_5','RED_50','RED_95').
- surface covariate grid data of the same covariates at grid positions x,y (here at 10 m resolution)
- optional: available water capacity (AWC) file: .tif file with multiple bands, where each band represents the available water capacity (in mm) at 1 cm depth intervals.

Some of the main results are shown below for feature-importance, model selection, soil predictions, and AWC-Constraints.

Attribution and Acknowledgments

This software was developed by the Sydney Informatics Hub, a core research facility of the University of Sydney, as a machine learning pilot App for GRDC.

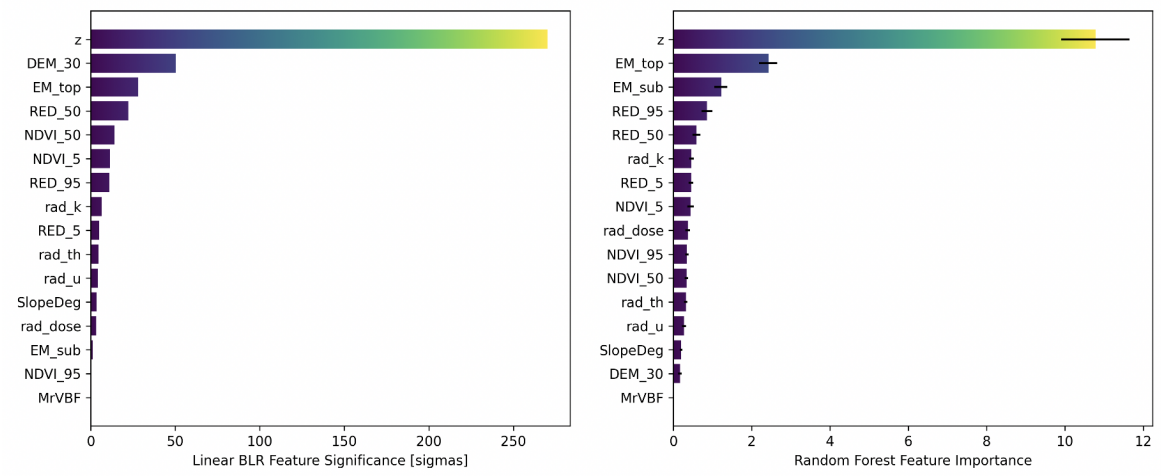


Figure 5: Example feature importance results for ESP. The left panel shows the BLR significance given by the ratio of correlation coefficient to standard deviation. The right panel shows the results of the Random Forest permutation importance.

Acknowledgments are an important way for us to demonstrate the value we bring to your research. Your research outcomes are vital for ongoing funding of the Sydney Informatics Hub.

If you make use of this software for your research project, please include the following acknowledgment:

"This research was supported by the Sydney Informatics Hub, a Core Research Facility of the University of Sydney."

Project Contributors

Key project contributors to this project are:

- Dr. Sebastian Haan (Sydney Informatics Hub, The University of Sydney): Main developer and machine learning scientist
- Prof. Brett Wheelan (Sydney Institute of Agriculture, The University of Sydney): Project lead of GRDC project
- Dr. Liana Pozza (Sydney Institute of Agriculture, The University of Sydney): Project research associate, software testing and data mapping
- Dr. Patrick Filippi (Sydney Institute of Agriculture, The University of Sydney): Project research associate, data extraction and mapping

License

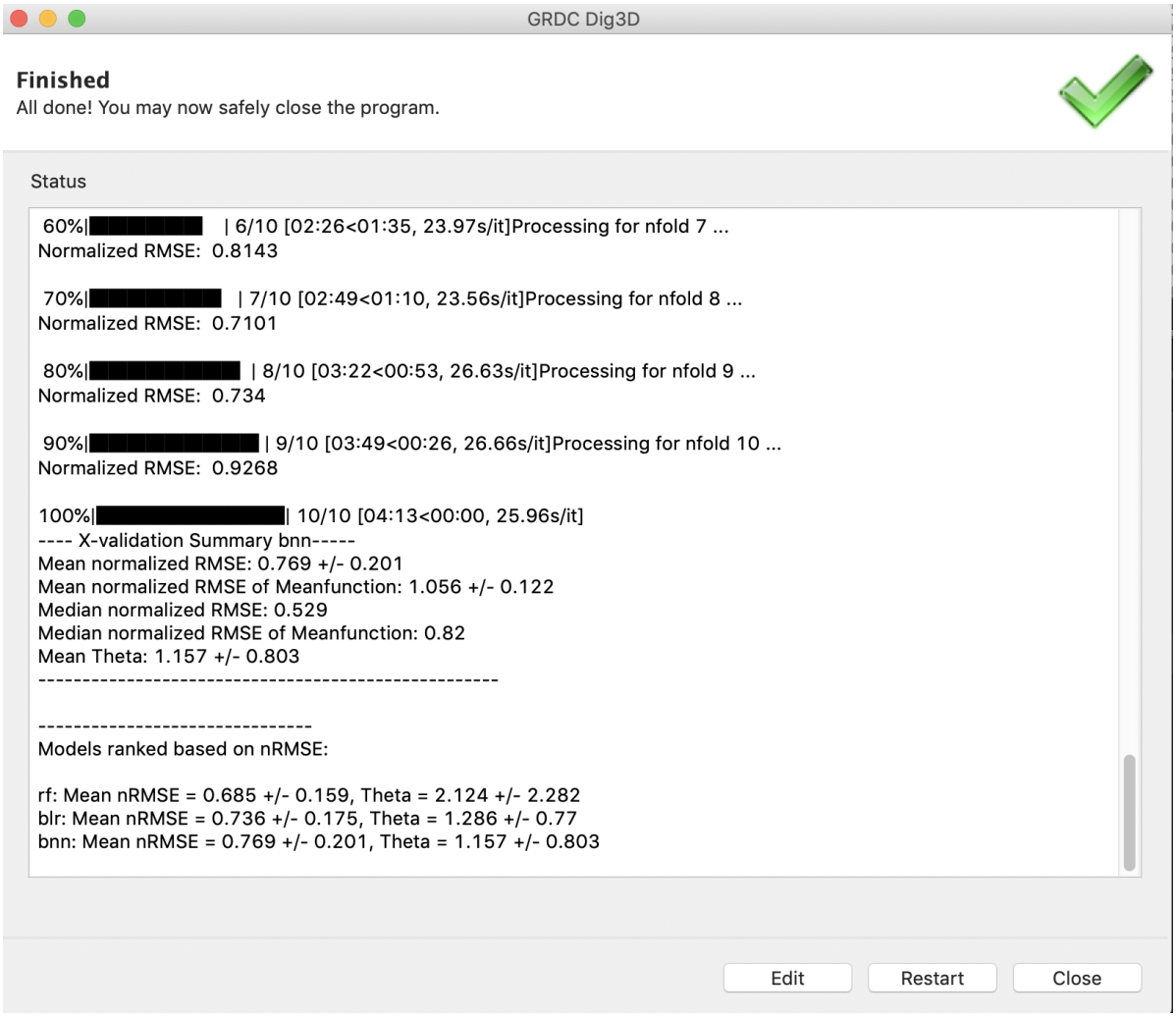


Figure 6: Example cross-validation model ranking for ESP after 10-fold cross-validation run.

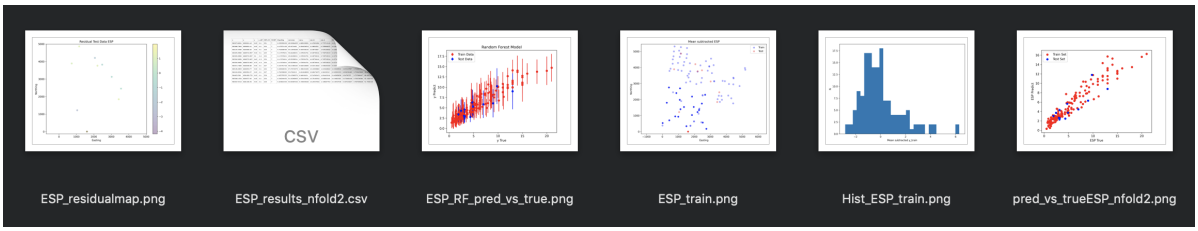


Figure 7: Overview plots for one cross-validation set out of ten

nfold	RMSE	nRMSE	RMEDIANSE	nRMEDIANSE	Theta	RMSE_fmean	nRMSE_fmean	RMEDIANSE_fmean	nRMEDIANSE_fmean
1	2.77	1.01	1.68	0.61	1.08	2.11	0.77	1.26	0.46
2	1.38	0.43	1.29	0.40	0.33	1.51	0.47	1.04	0.32
3	1.90	0.51	1.22	0.32	0.72	2.96	0.79	2.28	0.60
4	3.40	0.66	2.58	0.50	1.40	3.73	0.72	2.59	0.50
5	2.30	0.80	1.12	0.39	1.61	2.50	0.86	1.16	0.40
6	1.46	0.80	0.93	0.51	0.47	1.66	0.91	0.88	0.49
7	2.73	0.61	2.17	0.49	1.10	2.49	0.56	1.66	0.37
8	4.70	0.86	1.88	0.35	3.11	4.53	0.83	2.77	0.51
9	4.09	0.91	3.03	0.67	2.20	4.19	0.93	2.69	0.60
10	3.25	0.67	2.60	0.54	1.60	3.54	0.73	2.62	0.54

Figure 8: Example BLR cross-validation results for ESP for 10 cross-validation sets. The columns are defined as Root-Mean-Squared-Error (RMSE), nRMSE (RMSE/standard deviation of training data), Root-Median-Squared-Error (RMEDIANSE), Theta (the mean of ratios of the squared error to predicted variance) for the complete model and indicate with suffix `_fmean` for the results of mean function only

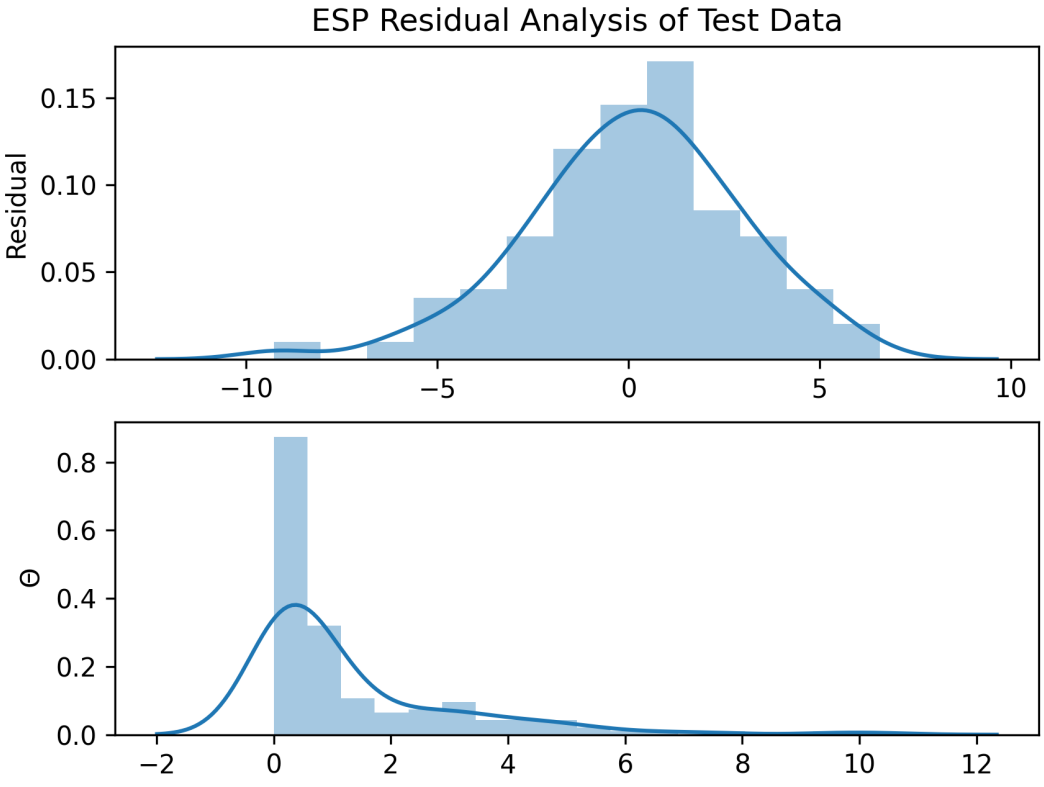


Figure 9: Example residual plot for cross-validation results for ESP. Shown are the residual error for test data (difference between model prediction and data that was unseen by model) at the top and theta (mean ratio of the squared error to predicted variance) at the bottom.

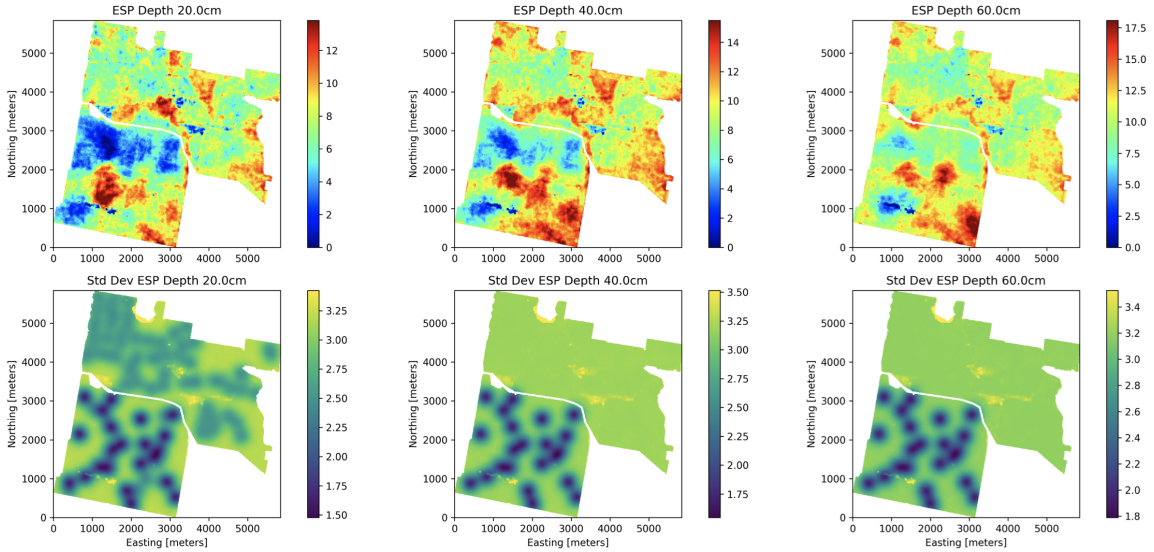


Figure 10: Results maps (top) and uncertainty (bottom) of ESP predictions at depths of 20, 40, and 60 cm.

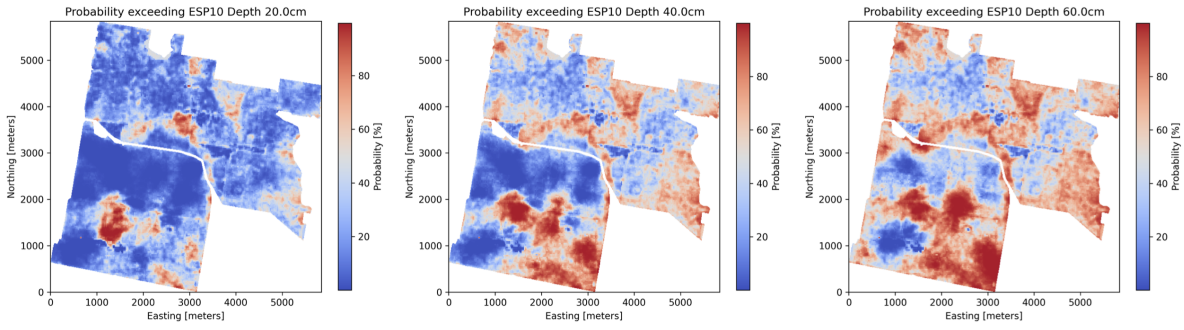


Figure 11: Probability maps of exceeding soil threshold ESP=10 at depths of 20, 40, and 60 cm depths.

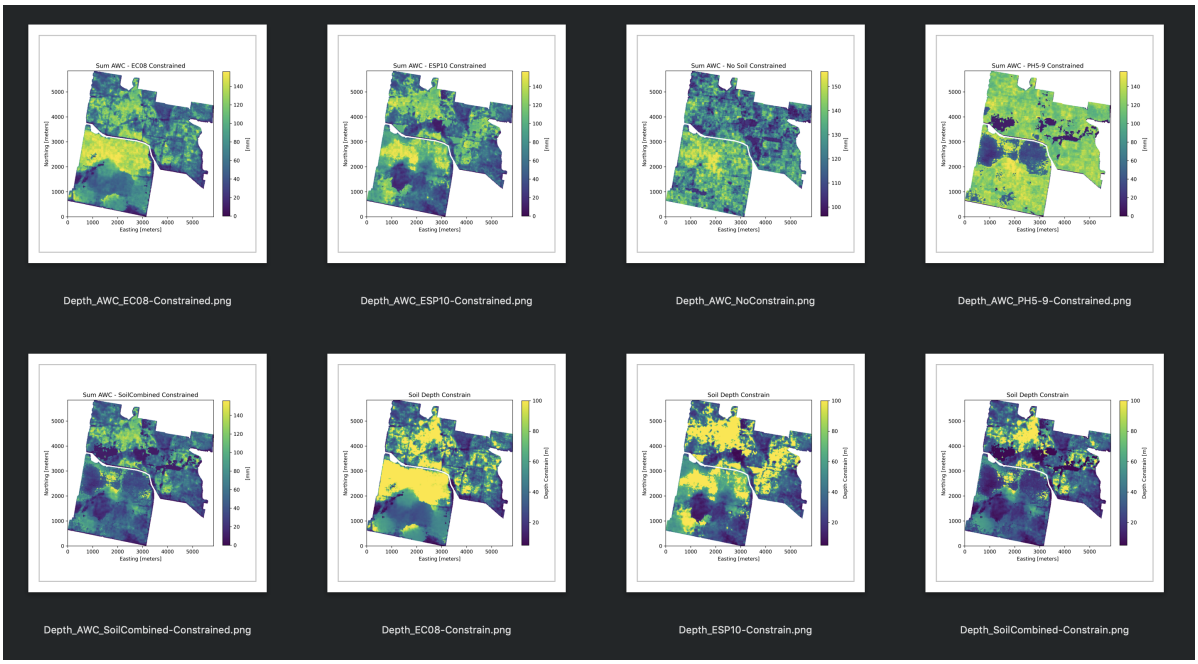


Figure 12: Example result overview of AWC soil constraint maps.