D4.5 Report on Machine Learning integration algorithms

Deliverable Lead	UNITUS
Deliverable due date	2022/11/30
Status	FINAL
Version	V1.0

Co-financed by the Connecting Europe Facility of the European Union

DOCUMENT CONTROL PAGE

Title	D4.5
Creator	UNITUS
Publisher	Highlander Consortium
Contributors	Giovanni Chillemi, Federica Gabbianelli, Marco Milanesi, Daniele Pietrucci, Giovanni Vignali [DIBAF], Balasubramanian Chandramouli [CINECA]
Type	Report
Language	EN
Rights	Copyright “Highlander Consortium”
Audience	Public X
	Restricted
Requested deadline	M38

Index

1. Introduction

In the Highlander project, data (i.e. Big Data) from various sources are collected in the Highlander Data Lake. To obtain useful information for the end-user, this huge and complex datastore needs to be dissected and different analysis layers were implemented and tuned to achieve the goal. A Machine Learning prototype, capable of analyzing, in an automatic way, the new data coming in the Highlander Data Lake, and converting it into information, was set up.

The document will present each section of the developed pipeline, showing its application in the “IoT for animal wellbeing” DApOS as case (Activity 5 - DApOS 7), as an example.

2. Premises

The pipeline here presented is deposited in a private github repository (https://github.com/pixpit/Highlander_ML).

The pipeline is implemented using Python language. In the repository the scripts are in two formats: Python and Notebook. In this way the users could apply the pipeline in a HPC system or easily run in a Jupyter Notebook. Moreover, the Jupyter Notebook reports an example about how to run the pipeline. The pipeline has been uploaded as a package (called "HighlanderML") in the PyPI repository. In this way, after installing the dependencies, the user can easily install the modules and functions to perform the Machine Learning analysis on his/her local machine or an HPC system.

The scheme of the Machine Learning prototype was presented and reported in Highlander **MS7 (Figure 1)**.

Based on the final scope of the analyses, specific input data of different nature (i.e. climatic data, productive data, remote sensing, etc.) and origin are merged and analyzed together. The final scope is to obtain a validated Machine Learning model, through the identification of the best Machine Learning algorithm, the optimized algorithm parameters and an automatic selection of the most important features. All the main functions used in H2O ambient or the one here developed can work on multiple threads.

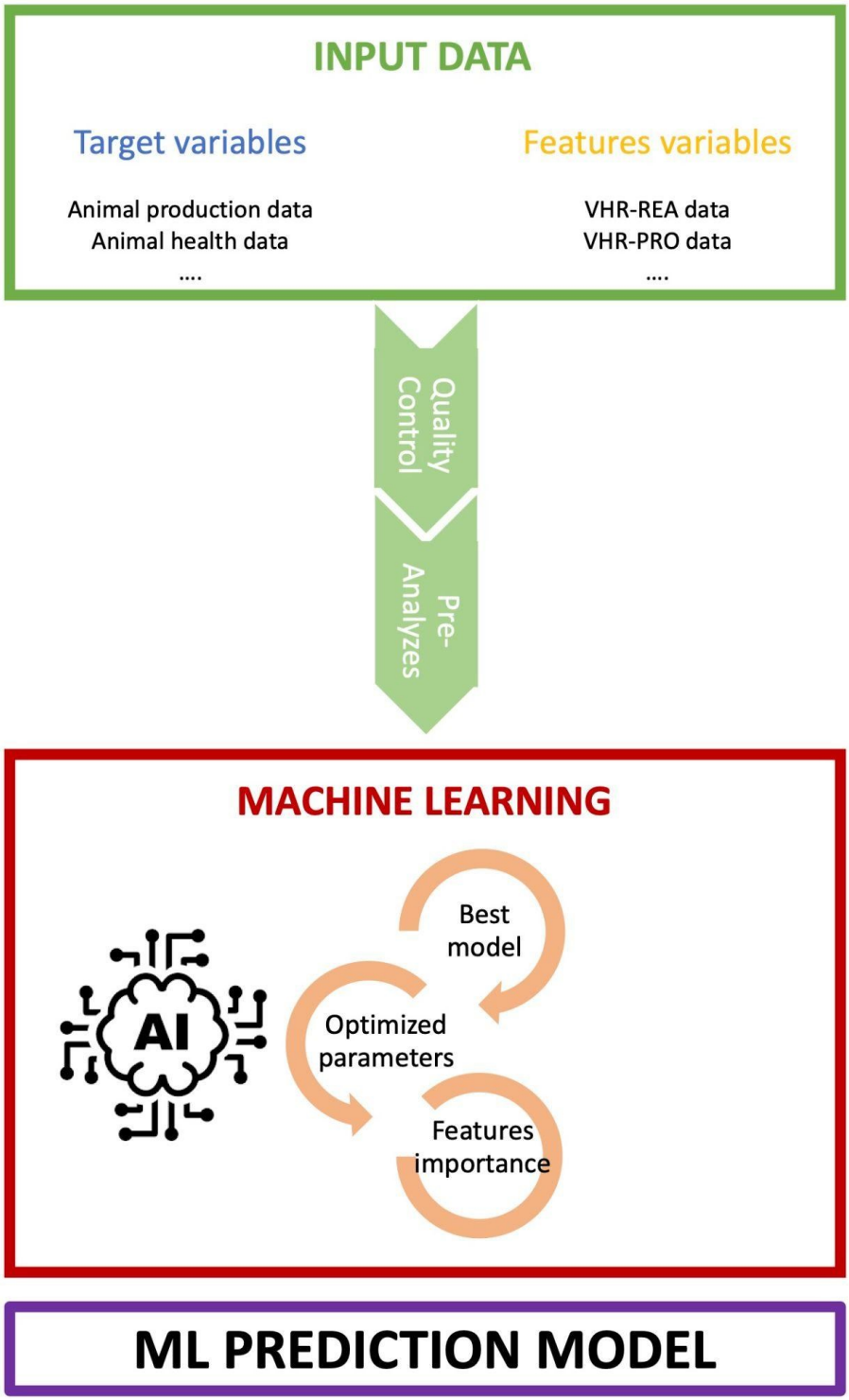

***Figure 1**: Schematic representation of the machine learning prototype from Highlander project.*

2.1 GitHub & Python Package Index repository

The GitHub repository, as mentioned before, contain the following file:

- "RFE_module.py" Python script, which includes the function to run "recursive feature elimination" analysis (RFE_highlander). This file is reported in the "highlander_scripts" directory;
- "H2O_module.py" Python script, containing the functions to run the Machine Learning (ML) analyses. This file is reported in the "highlander_scripts" directory. In particular, this module allows:
    - to search for the best algorithm to perform the prediction; the best algorithm is chosen using a set of different models (best_model)
    - to identify the best hyper-parameters, tuned using a grid search approach (grid_search);
    - to evaluate the relative importance of all the variables in the prediction and order them (varimp);
    - to train and test the model. For all the features, starting from the most important, the model accuracy is evaluated using the Mean Absolute Error metric (mae_nf).
    - to evaluate each feature contribution and explanation to the classification using the SHAP algorithm (shap).
    - to evaluate the feature impact in the model (`explain_model`)
- "Highlander_ML_notebook.ipynb" in the "Notebook" directory is a Jupyter Notebook script, already set to perform the analyses using an example dataset. The users could use it with any type of data. The same code is also reported in a python file code ("Highlander_ML_analysis.py")
- An example dataset to test the pipeline. It consists of 1000 records, 22 features (from F_1 to F_22) and one fictitious target variable (called "TARGET"). The file is stored in the "dataset" directory.
- The dependencies required to run the modules in a Python environment. The following Python packages are needed: pandas>=1.3.4; numpy>=1.21.2; scikit-learn>=1.0; h2o>=3.38.0.1. The list of packages is provided in the Github repository in the "dependencies" directory

The Python Package Index (PyPI) is a public repository of software for the Python programming language.

- In the GitHub repository is stored the setup.py file. This python script contains all the information to build a PyPI package. The setup.py file is used as input by the GitHub workflow software (using the "Publish Python Package" function) to create a package stored in the PyPI repository. This package is called "HighlanderML". The user can install the Highlander package using the pip command, with the following syntax: "pip HighlanderML". This installation provides all the dependencies, alongside with the modules stored in the "highlander_script" directory in the GitHub repository.

2.2 Pilot dataset

In this report, a pilot dataset was used to show how the pipeline works. As previously reported, we used the results from the "IoT for animal wellbeing" DApOS as case (Activity 5 - DApOS 7). In particular, the dataset refers to Pezzata Rossa Italiana cows reared in Friuli-Venezia Giulia. Milk data production was analyzed together with historical climatic data from the VHR-REA dataset (Raffa et al., 2021).

In detail, 2,332,083 records (i.e. functional control - FC) were analyzed after a quality control. The details about the quality control are reported in the Highlander M7 report. The residual of the milk production was analyzed (see Highlander M7 report for details about the procedure). In the pilot dataset, different climatic variables were evaluated. All the variables were measured up to 22 days before the measurement of the production of each animal. In detail, the following variables were analyzed: temperature, relative humidity, wind speed, cloud coverage and precipitation. For temperature and relative humidity, the mean, minimum and maximum values were computed for each day. In total, 198 variables were analyzed.

3. Recursive Feature Elimination

Recursive feature elimination (RFE) is a feature selection method that fits a pre-defined model and removes the weakest feature (or features) from the entire database, until the specified number of features is reached.

In our pipeline the Python script "RFE_module.py" performs a RFE using a DecisionTreeRegressor, as a model. The input dataset could be split to ease the analyses if the number of records is very high. However the RFE analyses could be repeated many times (the default parameter is 10, but is customizable) and the consensus among the replicates give the order of the features, from the most important to the least.

The function result is the ordered list of all the features from the original dataset (**Table 1**).

***Table 1**: The first 10 features (i.e. climatic variables) from the pilot dataset, ordered by the mean ranking value among 60 repetitions. In the table is reported the rank for the name of the climatic variable, the first 3 repetitions rank, the mean and standard deviation (STD) for the 60 repetitions.*

Climatic variables	Rank 0	Rank 1	Rank 2	*Rank …*	Mean	STD
T_MIN_mean_day_less_22	1	2	1		1.43	0.56
T_MAX_mean_day_less_1	2	2	1		1.83	1.56
WS_KMH_mean_day_less_4	3	26	3		10.55	10.51
WS_KMH_mean_day_less_5	12	5	16		13.92	10.09
WS_KMH_mean_day_less_1	6	13	7		14.00	9.99
WS_KMH_mean_day_less_9	4	20	4		19.35	12.97
WS_KMH_mean_day_less_2	19	16	32		19.52	11.70
WS_KMH_mean_day_less_22	8	22	19		20.27	14.10
WS_KMH_mean_day_less_16	22	3	31		22.92	14.95
WS_KMH_mean_day_less_8	10	6	27		23.37	14.86

Together with the order, the mean and standard deviation (SD) are given. The data could be plotted to understand the number of features to use in the subsequent steps (**Figure 2**).

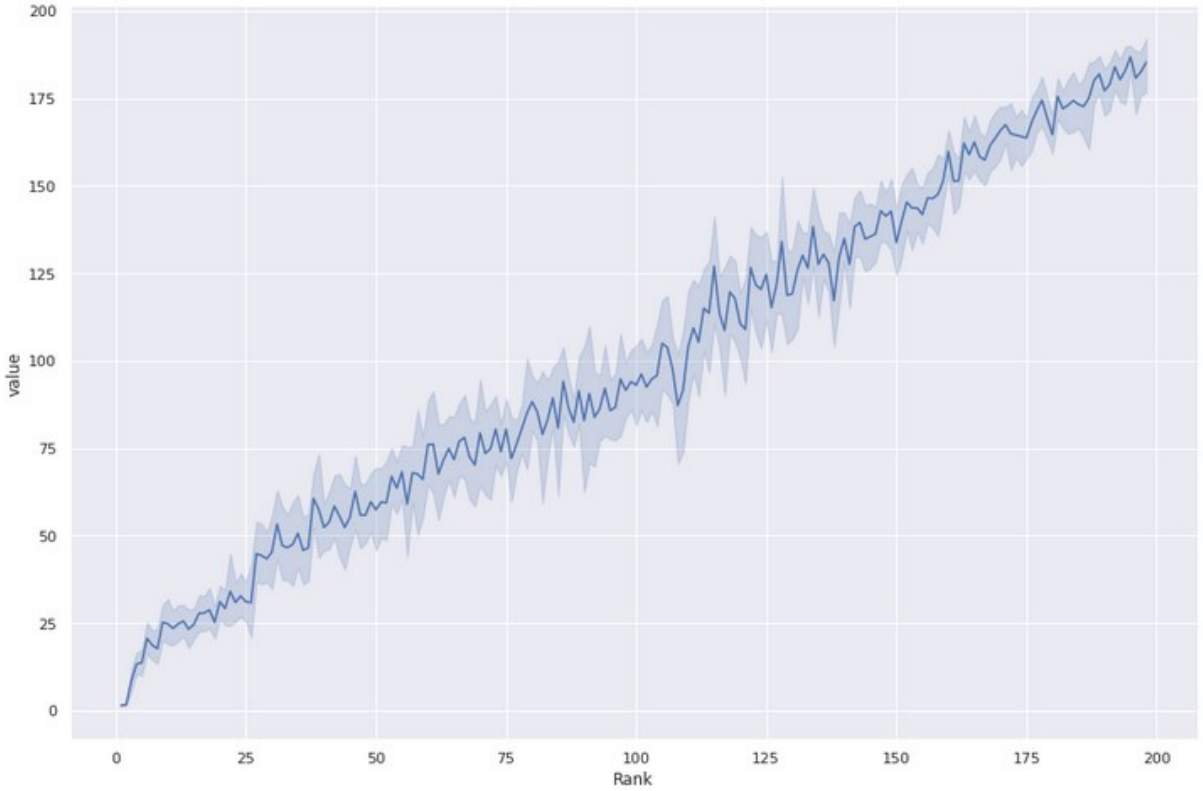

***Figure 2**: The plot of the ranking mean (solid blue line) and standard deviation (light blue shadow) for each feature among 60 repetitions in an increasing order.*

Using this information it is possible to identify how many features could have an effect on the investigated target variabile. in the case of the pilot, among the 198 features 100 were selected for the next steps.

4. Identification and test of the best Machine Learning algorithms

After the identification of the most important features associated with the target, a series of steps need to be performed to obtain a validated Machine Learning model. All the necessary functions were included in the "H2O_modules.py" Python script. Hereafter the single functions will be explained and an example of an application in the pilot dataset will be shown.

4.1 Identification of the best ML algorithm

First of all, the search for the best Machine Learning algorithm family is done using the H2O.ai and scikit-learn modules of Python. The function "best_model" was here implemented to ease the user experience. H2O tests different Machine Learning algorithms (for example, Random Forest, XGBOST, Gradient Boosting Machine - GBM), to identify the best one.

The function inputs are: 1) the dataframe (in which the rows represent the samples and the columns are the features plus the target variable) and 2) the name of the target variable. In addition, some H2O.ai parameters could also be set.

The output is a dataframe in which each row is a Machine Learning algorithm tested. A series of metrics are calculated and could be used to evaluate the model. The dataframe is sorted by RMSE (Root Mean Square Error) and the best model is the one reported in the first row (**Table 2**).

***Table 2**: The first 10 ML algorithm tested by H2O.ai from the pilot dataset. The best algorithm identified is "Gradient Boost Machine".*

Model id	Mean residual deviance	RMSE	MSE	MAE
GBM_grid_1_AutoML_1_20220923_210744_model_41	1489.58	38.59	1489.58	29.17
GBM_grid_1_AutoML_1_20220923_210744_model_5	1489.74	38.60	1489.74	29.18
GBM_grid_1_AutoML_1_20220923_210744_model_105	1491.36	38.62	1491.36	29.19
GBM_grid_1_AutoML_1_20220923_210744_model_49	1492.26	38.63	1492.26	29.22
XRT_1_AutoML_1_20220923_210744	1494.79	38.66	1494.80	29.24

Column legend: Root Mean Square Error (RMSE), Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Log-Error (RMSLE)

4.2 Model hyper-parameters optimization

In this step, the best ML algorithm previously selected by the function “best_model” is optimized. The optimization consists of tuning all parameters used by the algorithm. By tuning the parameters, the best prediction of the target variable can be ensured. This optimization is performed through a grid search approach, implemented in the “grid_search” function.

The inputs are the same as the previous function, namely the input dataframe and name of the target variable. Some default values are included in the functions, as a suggestion, but specific parameters are needed, depending on the previous function results.

The output is a dataframe in which columns report the tested hyper-parameters for each run and a metric to evaluate the run, in this case the MAE (Mean Absolute Error). Depending on the target variable nature, different metrics could be used. The dataframe is ordered using the MAE and, consequently, the best hyper-parameters are reported in the first row (Table 3).

***Table 3**: The first 5 hyper-parameters combinations tested. The best combination is reported in the first row and was used for the subsequent analyses.*

Col sample rate	Learn rate	Max depth	ntrees	Sample rate	Model ids	MAE
0.6	10	27.0	100.0	1.0	gbm_grid2_model_144	29.17
0.2	100	18.0	100.0	1.0	gbm_grid2_model_184	29.17
1.0	100	18.0	200.0	1.0	gbm_grid2_model_141	29.18
1.0	100	18.0	50.0	1.0	gbm_grid2_model_195	29.18
1.0	10	27.0	100.0	0.6	gbm_grid2_model_198	29.19

The values reported in the columns are the following: 1) “Col sample rate” (train accuracy)”, “Learn rate” (depth of the tree), “Max depth” (depht of the tree), “ntrees” (number of tree) and “Sample rate” (train accuracy): parameters used by the Gradient Boosting Machine algorithm; 2) “Model ids”, an unique identifier for each combination of parameters; 3) “MAE” (Mean Squared Error), the metric used to evaluate the performance of the models.

4.3 Identification of the most important features

Using the best algorithm and parameters previously identified, the complete dataset is analyzed with the objective to identify the most important features associated with the target variable. In this case, a feature importance approach was applied using the "varimp" function.

The output is a table with, for each variable, the relative and scaled importances, and the percentage of the importance (**Table 4**; **Figure 3**).

***Table 4**: The first 10 variables, ordered by their importance.*

Variable	Relative importance	Scaled importance	Percentage
T_MIN_mean_day_less_22	69807440.0	1.000	3.6
T_MAX_mean_day_less_1	62522792.0	0.896	3.2
WS_KMH_mean_day_less_4	30026174.0	0.430	1.5
WS_KMH_mean_day_less_1	25837002.0	0.370	1.3
WS_KMH_mean_day_less_5	25815808.0	0.370	1.3
WS_KMH_mean_day_less_2	25184130.0	0.361	1.3
RH_MIN_mean_day_less_2	25139532.0	0.360	1.3
WS_KMH_mean_day_less_9	23876186.0	0.342	1.2
T_MIN_mean_day_less_21	23448846.0	0.335	1.2

WS_KMH_mean_day_less_16	23101368.0	0.331	1.2

The columns report the following information: 1) "Variable", the name of the variable; 2) "Relative importance": the importance evaluated by the H2O package, the greater is the value the greatest is the variable importance; 3) "Scaled importance": the relative importance, scaled from 0 to 1; 4) "Percentage": the importance of the variable expressed as a percentage

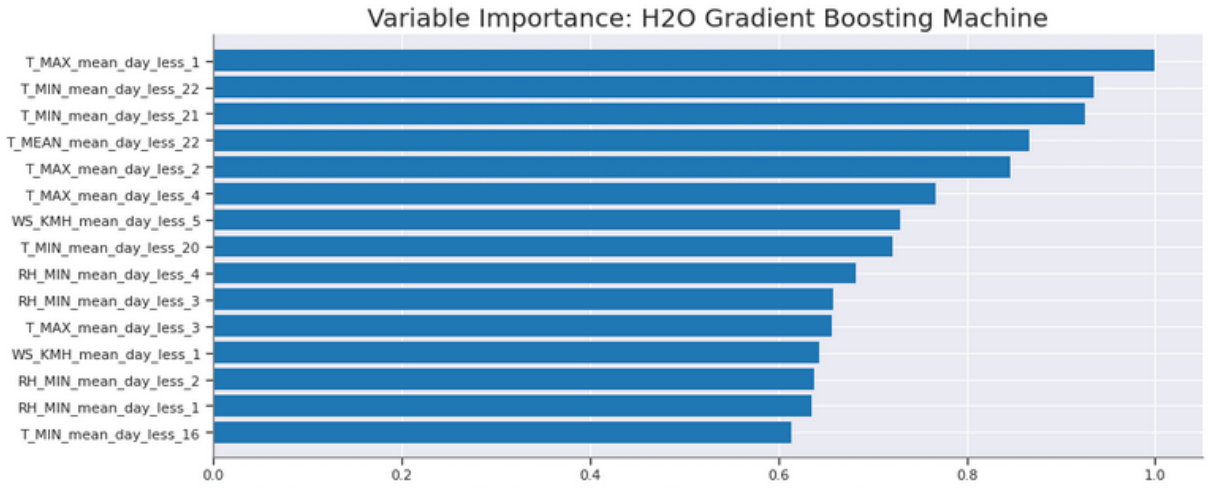

***Figure 3**: The plot of the first 10 variables, ordered by their importance. The x-axis represents the scaled feature importance.*

4.4 Identification of the most important features

To identify the minimal subset of the variable for the best algorithm and parameters previously identified, the previously ordered feature will be analyzed, from 2 (the n_feature_start parameter controls it) to the total number of features (the n_feature_end parameter control it), using the "mae_nf" function. The objective is to identify the number of variables that minimize the MAE, adding 1 or more variables (the step parameter controls it) at the time. In other words, the algorithm is trained and evaluated systematically, increasing the features at each step. Since the features are sorted using the var_imp function, it is possible to identify the minimal set of variables that can perform well as the dataset trained with all the variables. This process allows the elimination of the uninformative features, which are usually the ones with the lowest importance. The output is a table with the number of features tested and the corresponding MAE value (**Table 5**). To speed up the process, we did this analysis in two rounds: in the first one using a window step of 5, to find the zone where the MAE decreases, reaching a

plateau (**Figure 4**); in the second round, with a step of 1, to find the exact number of features to select (**Figure 5**). The final number of features chosen in the pilot dataset is 15.

***Table 5**: The table obtained with the "mae_nf" function. The MAE value is reported for each step, in this case from 2 to 100 features with a step of 5 value.*

MAE	Number of feature
29.49	2
29.44	7
29.30	12
29.24	17
29.22	22
29.20	27
29.19	32
29.18	37
29.17	42
29.17	47
29.17	52

The obtained values could be plotted (**Figure 4**) and used to identify the minimum, using for example the elbow method.

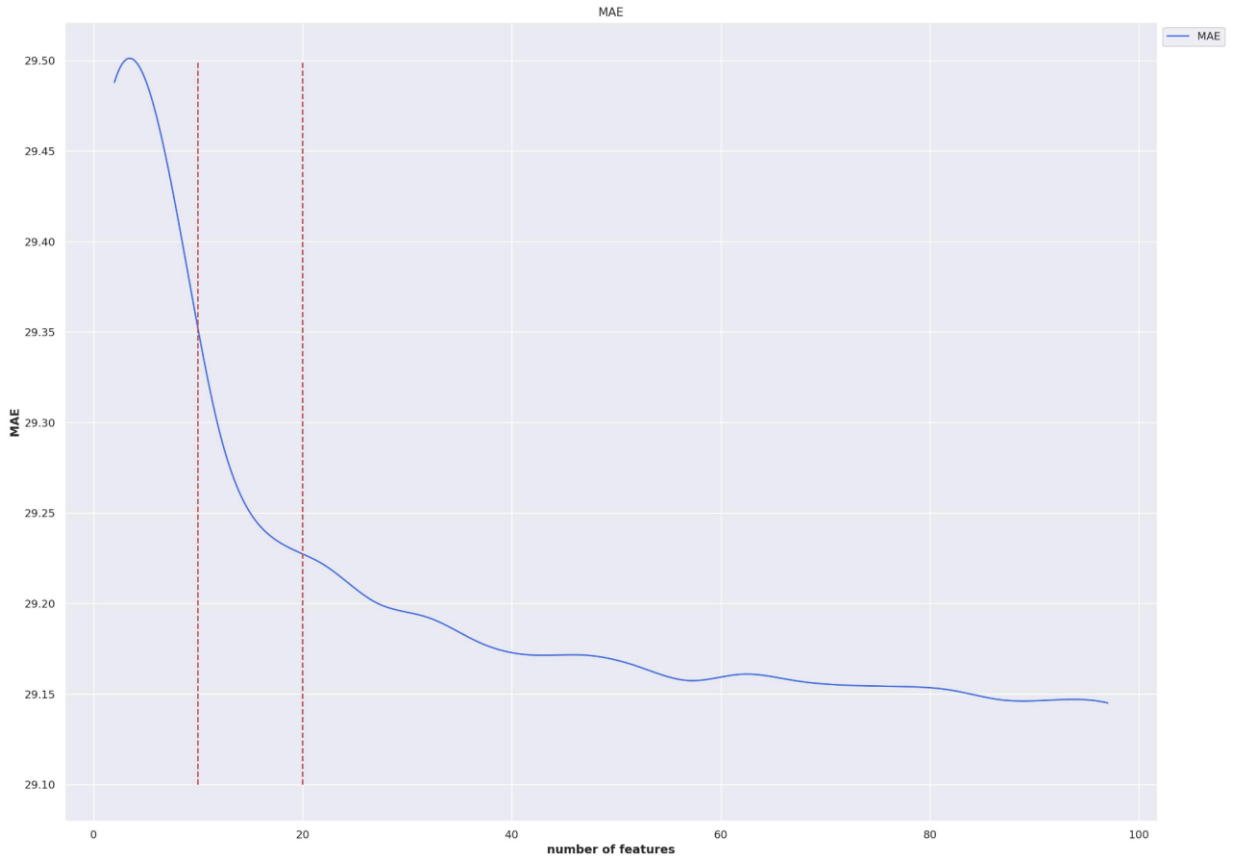

***Figure 4**: The plot of the MAE values from 2 to 100 with step 5, in the pilot dataset (i.e. the same reported in **Table 5**). The chosen window (red dashed line) for the next evaluation was from 10 to 20.*

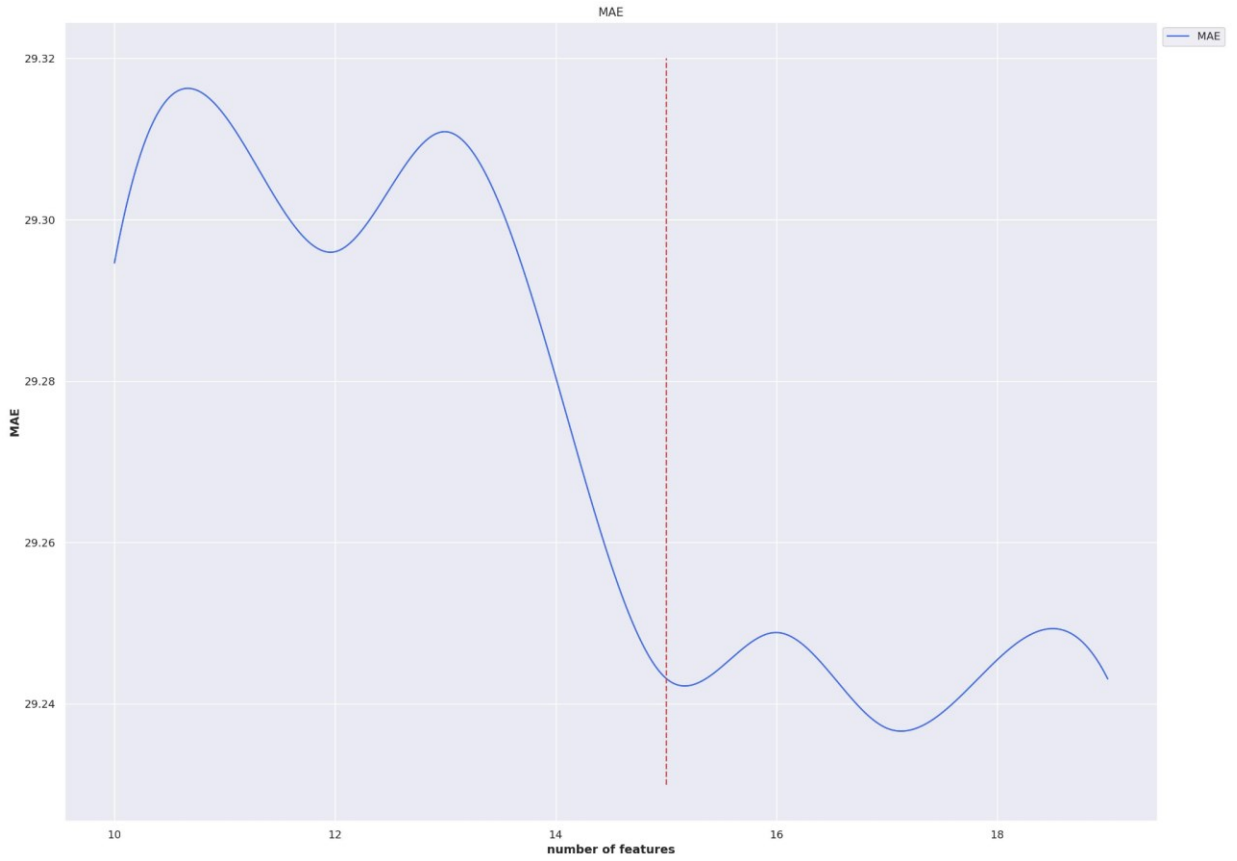

***Figure 5**: The plot of the MAE values from 10 to 20 (step 1) obtained in the pilot dataset. The chosen number of most important features is 15 (red dashed line).*

4.5 Evaluation of the identified model

Finally, a new dataset with only the most important features was created. In the case of the pilot, 15 variables were selected. This new dataset with the best algorithm and parameters is the result of this pipeline.

The “varimp” function could be used to evaluate the features importance. Compared with the previous analyses, the reduced number of features could influence the rank and relative importance (**Figure 6**).

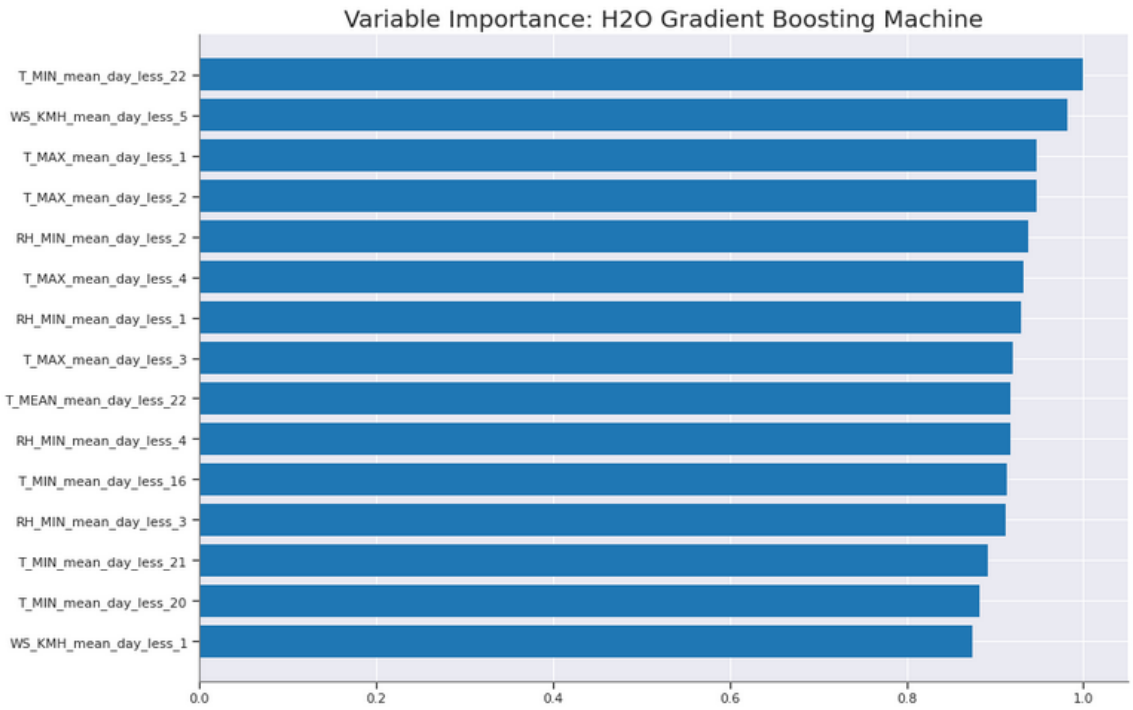

***Figure 6**: The plot of the 15* previously *selected variables, ordered by their importance. The x-axis represents the scaled feature importance.*

In addition, the SHAP algorithm (SHapley Additive exPlanations) was implemented in the "shap" function, to allow the explainability of each feature in the classification (Lundeberg et al., 2017). The function output is a plot, called "SHAP" or "swarm" plot (**Figure 7**).

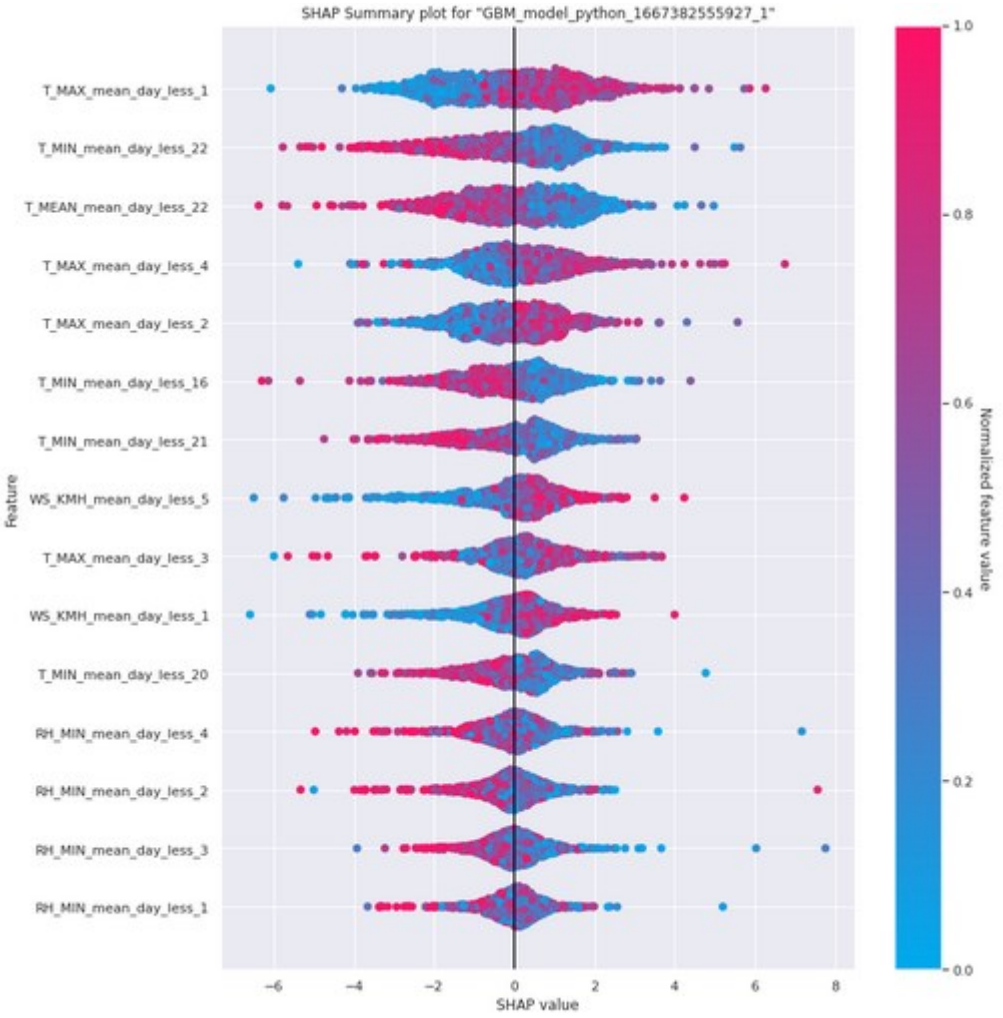

***Figure 7**: The SHAP graph with the feature explainability for the 15 features selected, ordered by their importance. The x-axis represents the feature, the y-axis represents the SHAP value. Each point is a sample (the residual of an animal production) and its color represents the normalized value for each feature. The values are normalized to compare different features with different units of measurement. The minimum value that the feature can assume is 0, while the maximum is 1. The normalized value is colored using a gradient color scale, from blue (0) to purple (1). If a point is associated with a positive SHAP value, then the feature contributes positively to the prediction. Conversely, if a point is associated with a negative SHAP value, then the feature contributes negatively to the prediction. For example, high values of "T_MAX_mean_day_less_1'' have a high positive contribution to the prediction, while low values have a high negative contribution to the prediction.*

Finally, some interesting and summarizing statistics about the identified model could be obtained using the "explain_model" function. In particular: Residual Analysis (Miles, 2014) (**Figure 8**), Variance Importance (Inglis et al., 2022), SHAP summary (Lundeberg et al., 2017),

Partial Dependence plots (**Figure 9**) and Individual Condition Expectation plots (Goldstein et al., 2015) (**Figure 10**) are produced as output.

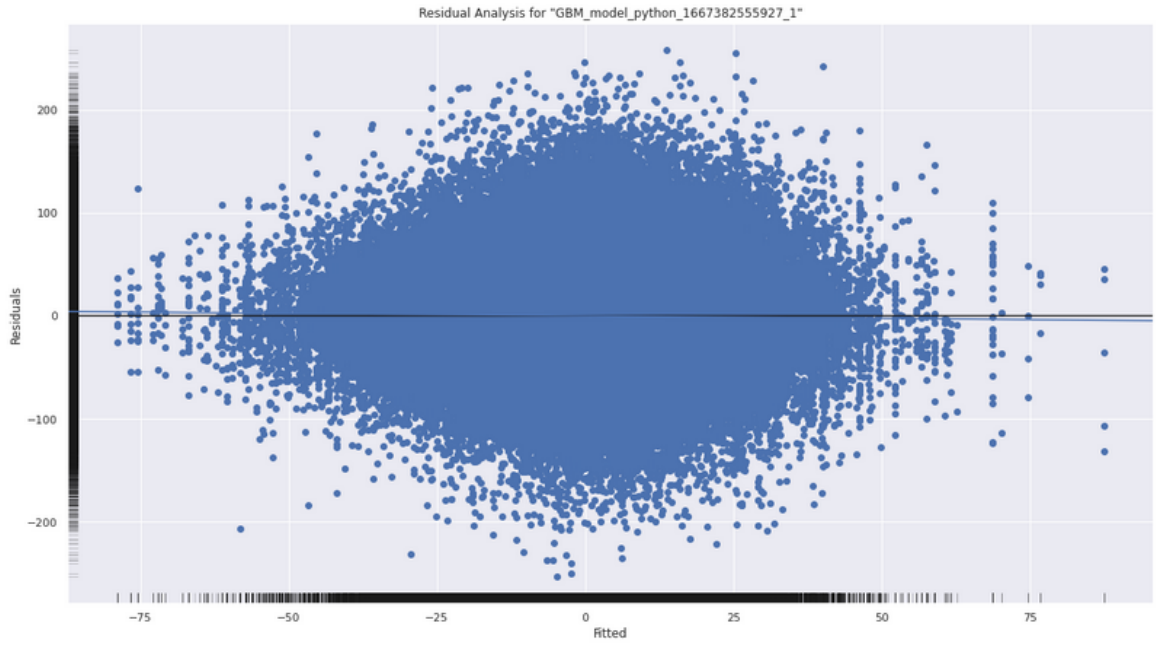

***Figure 8**: The Residual Analysis graph from the pilot dataset.*

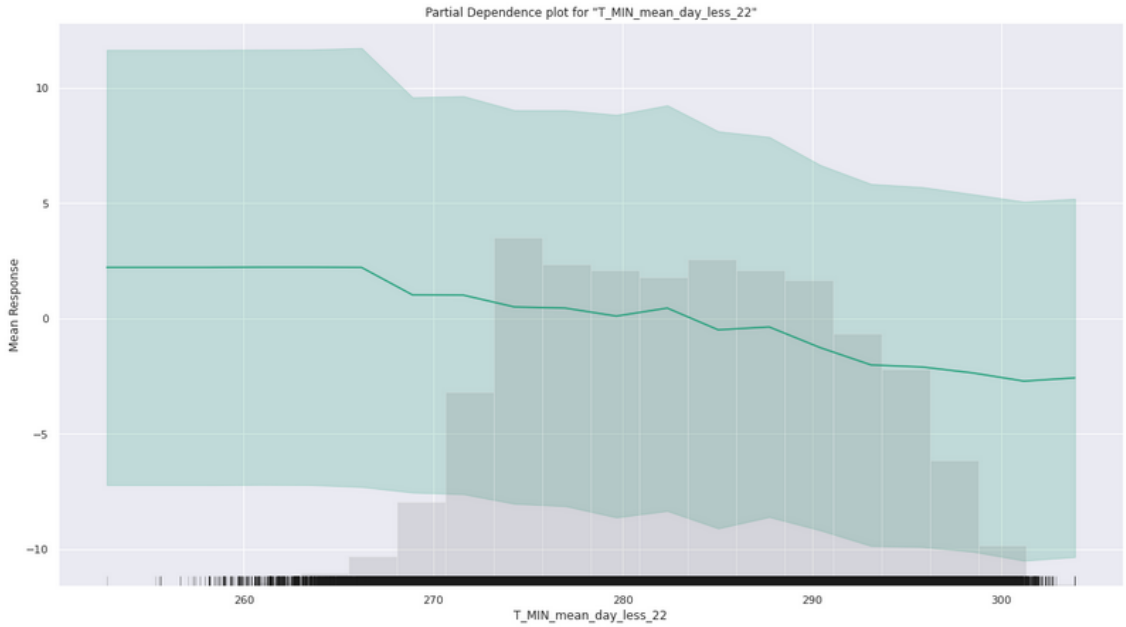

***Figure 9**: One of the Partial Dependence plot from the pilot dataset.*

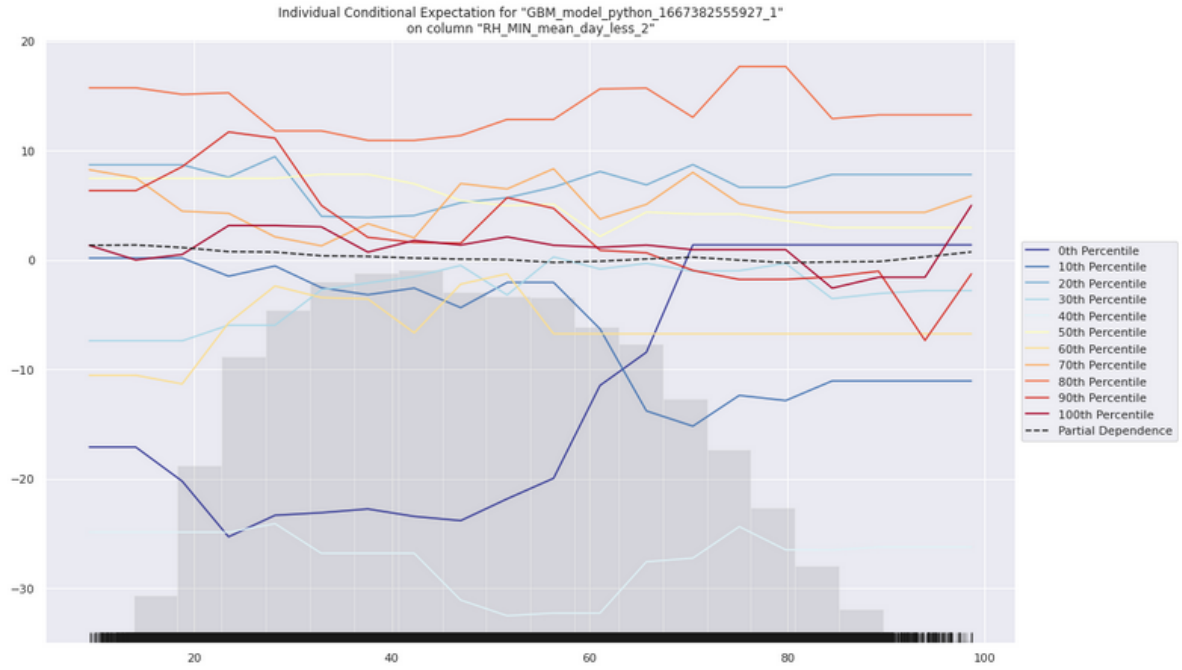

***Figure 10**: One of the Individual Condition Expectation plot from the pilot dataset.*

5 Conclusions

In this document we reported the Machine Learning integration algorithms developed in the Highlander project.

The pipeline is able to prepare the input data and identify the best Machine Learning model to analyze them. Once the best model is identified, it is optimized using a grid search approach. Finally, a feature selection is performed: the importance of each variable is evaluated, and the uninformative variables are removed. In this way, the model is simpler to run, and the most relevant variables can be studied to understand their role in the prediction. In addition, it is possible to perform some hypotheses of how the features (the climatic variables) can affect the target variable (the milk yield). In fact, the pipeline generates several graphs (i.e. residual plot, SHAP plot, Individual Conditional Expectation) that can help the user to interpret the results of the prediction.

These algorithms were applied in some of the Highlander DApOS. Here, as reported, was presented the results from a pilot analysis from “IoT for animal wellbeing” DApOS (number 7).

6 Bibliography

Goldstein, A., Kapelner, A., Bleich, J., & Pitkin, E. (2015). Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation. *Journal of Computational and Graphical Statistics*. https://doi.org/10.1080/10618600.2014.907095

Inglis, A., Parnell, A., & Hurley, C. B. (2022). Visualizing Variable Importance and Variable Interaction Effects in Machine Learning Models. *Journal of Computational and Graphical Statistics*, *31*(3), 766–778. https://doi.org/10.1080/10618600.2021.2007935

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, *2017-Decem*, 4766–4775.

Miles, J. (2014). Residual Plot. In *Wiley StatsRef: Statistics Reference Online*. https://doi.org/10.1002/9781118445112.stat06619

Raffa, M., Reder, A., Marras, G. F., Mancini, M., Scipione, G., Santini, M., & Mercogliano, P. (2021). VHR-REA_IT dataset: Very high resolution dynamical downscaling of ERA5 reanalysis over Italy by COSMO-CLM. *Data*, *6*(8). https://doi.org/10.3390/data6080088