

Subscribe to DeepL Pro to edit this document.  
Visit www.DeepL.com/Pro for more information.

Deliverable A3.4

Release Date:16/09/2020

[**Tool Used**](#_rz3evkmp7bez) **2**

[ARFF format](#_tnh20bk30v3e) 3

[Weka and Java - Advantages](#_7r0s3ohnwnlz) 4

[HW eSW Requirements](#_v5szzobd7q2e) 5

[Weka and Data Mining](#_srvxnuyalcwc) 6

[Forecasting plugin](#_6wu1cra9jkb) 7

[**Operations performed and results**](#_feibwod2fmm6) **11**

[Creating the file to analyze](#_7oe09fk3myf5) 11

[Detection of outliers and zero values](#_c8mwm7bg35v1) 13

[Weather labeling](#_hx5h1uxme6cf) 16

[**Data analysis**](#_owrab8hvekor) **19**

[SIMOreg](#_y3xxqe187rfn) 19

[Random Forest](#_rc7sd56ac63l) 20

[Perceptron Multilayer](#_jgxv7cqitwbn) 23

[Temperature forecast from numerical values](#_ylpb3ueavole) 24

[Forecast with labeling insertion for weather conditions](#_l60qtlbbk1z9) 25

[RandomForest](#_qnl2kmc0fxbz) 25

[SMO Classify](#_qjqy456gz2oh) 26

[Multilayer perceptron 6nodes](#_v60cuvtv3plh) 27

[Consumption forecast Vs numerical weather features](#_ixs4896s9wp2) 28

[RF](#_9nncft66cw9w) 28

[SIMoreg](#_e0wcpxk4dpym) 29

[MPL](#_wprb0tnrwjkl) 30

[Consumption vs weather labeling](#_4lru7snvugr8) 32

[RF](#_cypwrnw598xs) 32

[SMO](#_htr2tnnrbjmo) 33

[MPL Classify](#_5kr586pmdrwj) 33

[**Appendix**](#_fymhw32y4qjw) **35**

The following will describe the activities carried out on the starting dataset and the software used for the evaluation of the forecast model.

In this phase we adopted, as will be described below, a predictive tool that would allow to test as many models as possible in order to identify the most performing one.

# Tool Used

Weka (Waikato Environment for Knowledge Analysis) is a collection of open-source machine learning algorithms that allow to perform preprocessing, classification, clustering and regression on data sets in a supervised and unsupervised way. In the execution of the available algorithms Weka allows the setting of parameters in order to customize the execution.

*"The programme aims to build a state-of-art facility for developing techniques of machine learning and investigating their application in key areas of the New Zealand economy. Specifically we will create a workbench for machine learning, determine the factor that contribute towards its successful application in the agricultural industries, and develop new methods of machine learning and ways of assessing their effectiveness."*

The choice made fell on this tool for the considerable opportunities offered:

**Portability:** Weka is written and implemented in Java, an interpreted language, so in order to run the Weka program it is sufficient that the Java Runtime Environment (JRE) is installed on the computer.

**Memory management:** since a Java program runs inside a virtual machine, memory management is controlled and managed by an appropriate module, called Garbage Collection.

**Object-oriented**.

**Community Documentation:** Weka has a large community working on its development and implementation of new plugins. It is also very easy to find materials for the study and for its use in complex areas.

**API:** Weka's code was written according to JSR-000073 or Java Data Mining API Specification Request in order to ensure an easy and effective integration between a generic application and its engine.

**Different modes of use.** The Weka software can be used in different ways: from the command line, through the graphical interface or by inserting in Java project. All well documented.

## ARFF format

The dataset format for Weka is **.ARFF**, Attribute Relationship File Format.

ARFF is a text format that shows the structure of a database. An arff file basically contains two distinct sections: header and data.

***The header*** contains the fields related to the name of the dataset and the attributes it contains, for example:

@relation YB2\_19\_20

@attribute Fulltime DATE "yyyyy-MM-dd HH:mm:ss"

@attribute tons numeric

@attribute ampere numeric

@attribute numeric consumption

@attribute Hour numeric

@attribute Numeric dates

@attribute 'Day of Week' numeric

@attribute Month numeric

@attribute Year numeric

@attribute dayname {Tuesday,Wednesday,Thursday,Friday,Monday}

@attribute temp numeric

@attribute feels\_like numeric

@attribute temp\_min numeric

@attribute temp\_max numeric

@attribute pressure numeric

@attribute numeric humidity

@attribute wind\_speed numeric

@attribute clouds\_all numeric

@attribute weather\_main {Clouds,Clear,Rain,Thunderstorm,Mist,Fog,Drizzle}

In the data section you will find the actual values of the measurements made or the quantities observed. In our case:

@data

'2019-07-1610:00:00',140.5,11.35,1,10,16,1,7,2019,Tuesday,25.66,26.03,20,28.33,997,50,1.5,28,Clouds

'2019-07-1611:00:00',305.9,20.03,3,11,16,1,7,2019,Tuesday,26.83,27.59,22.22,30.56,1001,50,1.5,37,Clouds

'2019-07-1612:00:00',362.8,23.8,3.5,12,16,1,7,2019,Tuesday,27.3,27.8,24.44,29.44,998,50,2.1,32,Clouds

'2019-07-1613:00:00',375.8,24.14,3.4,13,16,1,7,2019,Tuesday,27.86,28.73,25,30.56,995,48,1.5,25,Clouds

## Weka and Java - Advantages

As stated above, there are many reasons to rely on software such as Weka in the preliminary stages of building a predictive model. It should also be considered that Weka can also be used in more advanced stages, such as the production of the algorithm, in fact it is possible to integrate a Weka jar package into the Java project that is being implemented, like any other library.

After integrating the jar into the project you can import the classes to use all the functionality that Weka is able to provide. In this way you can implement a Data Mining application by instantiating in the code objects such as classifiers, filters and all the other tools inside Weka and use the different available algorithms associated with them without having to implement the code from time to time.

Below is an example of using Weka within a Java project, where you try to use a Bayesian classifier on numerical, vectorial and boolean variables and return the test results.

At the beginning of the class you import the weka classes you want to use and then call and instantiate them as usual in Java.

//Example of using Weka software in a Java program

import weka.classifiers.Classifier;

import weka.classifiers.Evaluation;

import weka.classifiers.bayes.NaiveBayes;

import weka.core.Attribute;

import weka.core.FastVector;

import weka.core.Instance;

import weka.core.Instances;

public class Test{

public static void main(String[] args) throws Exception{

// Declare two numeric attributes

Attribute Attribute1 = new Attribute("firstNumeric");

Attribute Attribute2 = new Attribute("secondNumeric");

// Declare a nominal attribute along with its values

FastVector fvNominalVal = new FastVector(3);

fvNominalVal.addElement("blue");

fvNominalVal.addElement("gray");

fvNominalVal.addElement("black");

Attribute Attribute3 = new Attribute("aNominal", fvNominalVal);

// Declare the class attribute along with its values

FastVector fvClassVal = new FastVector(2);

fvClassVal.addElement("positive");

fvClassVal.addElement("negative");

Attribute ClassAttribute = new Attribute("theClass", fvClassVal);

// Declare the feature vector

FastVector fvWekaAttributes = new FastVector(4);

fvWekaAttributes.addElement(Attribute1);

fvWekaAttributes.addElement(Attribute2);

fvWekaAttributes.addElement(Attribute3);

fvWekaAttributes.addElement(ClassAttribute);

// Create an empty training set

Instances isTrainingSet = new Instances("Rel", fvWekaAttributes, 10);

// Set class index

isTrainingSet.setClassIndex(3);

// Create the instance

Instance iExample = new Instance(4);

iExample.setValue((Attribute)fvWekaAttributes.elementAt(0), 1.0);

iExample.setValue((Attribute)fvWekaAttributes.elementAt(1), 0.5);

iExample.setValue((Attribute)fvWekaAttributes.elementAt(2), "gray");

iExample.setValue((Attribute)fvWekaAttributes.elementAt(3), "positive");

// add the instance

isTrainingSet.add(iExample);

Classifier cModel = (Classifier)new NaiveBayes();

cModel.buildClassifier(isTrainingSet);

// Test the model

Evaluation eTest = new Evaluation(isTrainingSet);

eTest.evaluateModel(cModel, isTrainingSet);

// Print the result à la Weka explorer:

String strSummary = eTest.toSummaryString();

System.out.println(strSummary);

// Get the confusion matrix

double[][] cmMatrix = eTest.confusionMatrix();

for(int row\_i=0; row\_i< cmMatrix.length; row\_i++){

for(int col\_i=0; col\_i< cmMatrix.length; col\_i++){

System.out.print(cmMatrix[row\_i][col\_i]);

System.out.print("|");

}

System.out.println();

}

}

}

## HW eSW Requirements

The following matrix shows which minimum version of Java is required to run a specific version of Weka. The latest official versions of Weka require Java 8 or higher. It should be noted that in cases using a Windows machine with a HiDPI display, you may need to use Java 9 or higher to avoid problems with inappropriate resizing of Weka's graphical user interfaces.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Weka** | **Java 1.4** | **Java 5** | **Java 6** | **Java 7** | **Java 8 or later** |
| <3.4.0 | ☑ | ☑ | ☑ | ☑ | ☑ |
| 3.4.x | ☑ | ☑ | ☑ | ☑ | ☑ |
| 3.5.x | <3.5.3 | ☑ | ☑ | ☑ | ☑ |
| 3.6.x |  | ☑ | ☑ | ☑ | ☑ |
| 3.7.x |  | 3.7.0 | <3.7.14 | ☑ | ☑ |
| 3.8.x |  |  |  | <3.8.2 | ☑ |
| 3.9.x |  |  |  | <3.9.2 | ☑ |

***Tab. 1***

In addition to the Java specifications, Weka does not need other special requirements, obviously the available ram must be proportionate with the amount of data under analysis, however, we recommend machines with a ram not less than 32G.

## Weka and Data Mining

For data mining weka provides numerous algorithms and techniques that allow a great versatility from the operational point of view, versatility that adds to its portability due to the Java language.

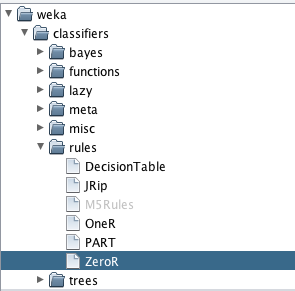
In addition, Weka can be used:

* through a graphical user interface (GUI)
* from command line
* as an importable library in Java applications.

Weka made a classification saves the generated models in .model extension files allowing easy reuse at a later time. By default, without adding plugins, dataminig offers the following features:

* Data preprocessing.
* Selection of attributes.
* Classification.
* Regression.
* Clustering.
* Associative rules.

We report, in Fig. 1, as an example, a screenshot of the algorithms available in Weka:



***Fig. 1 Weka models example***

## Forecasting plugin

In order to be able to better analyze and predict the data obtained from the Ema Control platform, a plugin developed specifically for time series analysis has been integrated into Weka, a plugin that allows you to analyze a series of data points depending on time or better in succession.

Time series forecasting is a process of using a model to generate forecasts based on known past events. Time series data have a natural time order, which differs from typical data mining/machine learning applications where each point is an independent example to be learned and the order of the data points within a data set does not matter.

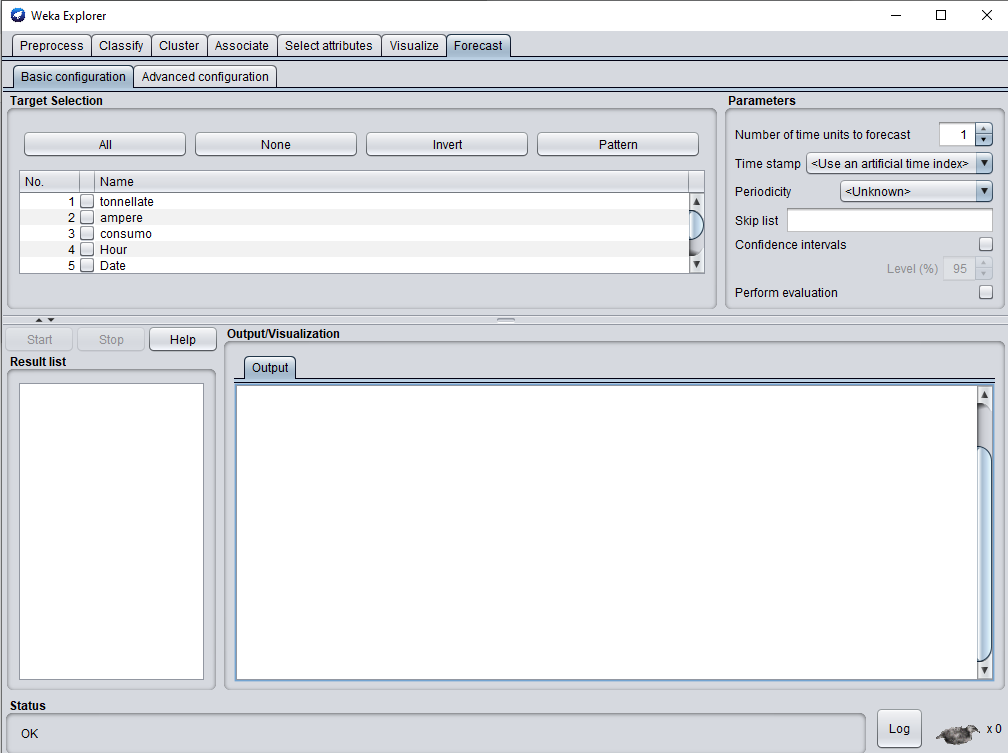
In Weka (> = 3.7.3) you can incorporate this time series analysis tool that allows you to develop, evaluate and visualize forecast models. This environment takes the form of a tab added in Weka's graphical user interface "Explorer" and can be installed through the package manager.

In particular, the plugin used adopts a machine learning/data mining approach to model time series by transforming the data into formats that the learning algorithms can process. The modeling environment of the "core" time series is available as free open source software and can be installed following the instructions given on the site by the developers[[1]](#footnote-1).

The plugin allows you to use a basic version and an advanced version, which can be integrated with each other.

**Basic Configuration** (Fig .2), below is a description of the functionalities present in this configuration.

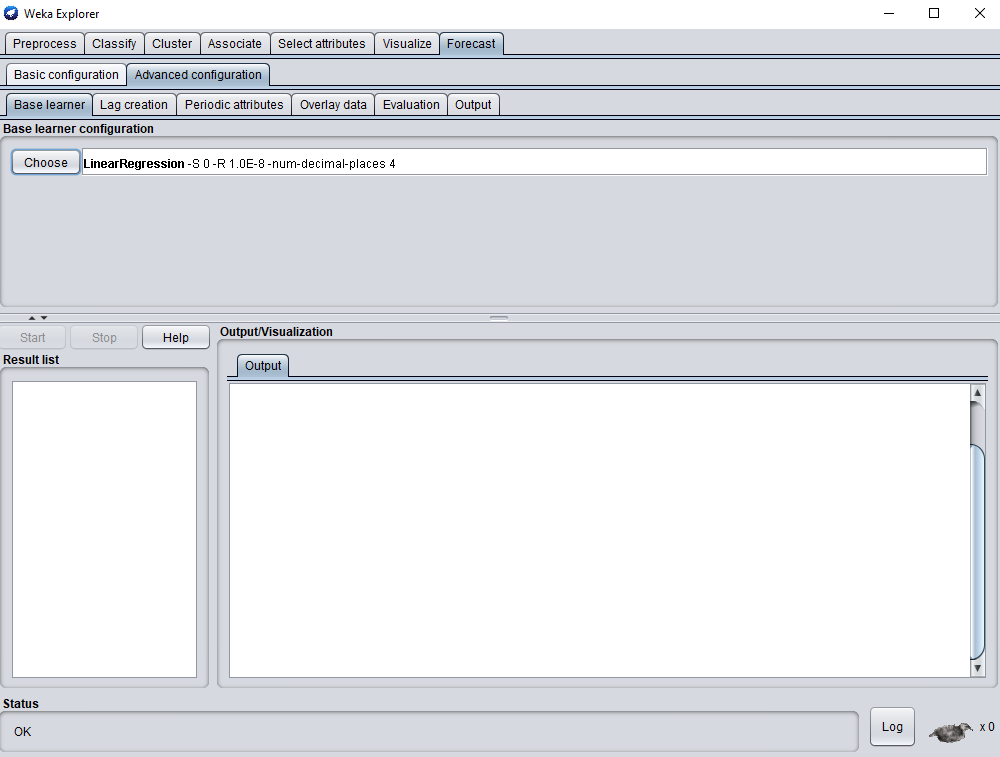
* The menu box allows you to select the target under analysis;
* The *Number of units to forecast* indicator defines for how many "time steps" the forecast will be produced. The default setting is 1, which means that the system will make a single forecast 1 step ahead.
* The *Timestamp* indicator, allows the user to select which, if any, is the field in the data contains the timestamp. If there is a date field in the data, the system automatically selects it. If there is no date field in the data, the *"<Use an artificial time index>"* option is selected automatically. Note that the timestamp is an undated numeric field (because the system cannot distinguish it from a potential target). The user also has the option to select *"<None>" from the* drop-down box to indicate to the system that no timestamp (artificial or not) should be used.
* The *Periodicity* drop-down menu allows the user to specify the periodicity of the data. If a date field has been selected as a timestamp, the system can use heuristics to automatically detect the periodicity - *"<Detect automatically>" will be* set by default. If the timestamp is not a date, the user can explicitly tell the system what the periodicity is or select *"<Unknow>" if it is not* known.
* The *Skip List bar,* allows the user to specify time periods that should not be considered as an increase in timestamp compared to the modeling, forecasting and visualization process (in our case, for example, the nighttime hours of plant closure). The heuristics used to automatically detect the periodicity cannot cope with these "holes" in the data, so the user must specify a periodicity to be used and provide time periods that should not be considered as increments in the Skip List text field. Strings such as "weekend", "sat", "tuesday", , specific dates (with an optional formatting string) such as "2011-07-04@yyyy-MM-dd"and integers (which are interpreted differently depending on the specified periodicity) can be entered in the field.
* The checkbox and the numeric field *Confidence intervals* allow the user to enter the confidence limits on the forecast in progress.
  + The default confidence level is 95%. The system uses predictions made for known target values in the training data to set confidence limits. Therefore, a confidence level of 95% means that 95% of the actual target values are within the range. Note that the confidence intervals are calculated for each step forward independently, i.e. all one step forward predictions on the training data are used to calculate the one step forward confidence interval, all two step forward predictions are used to calculate the two step forward interval and so on.
* *Perform evaluation*, instructs the system to perform a prevision evaluation using the training data. That is, once the model has been trained on the data, it is applied to make a prediction at any time (in order) by passing through the data. These forecasts are collected and summarized, using various metrics, for each expected future time step, i.e. all one-step forecasts are collected and summarized, all two-step forecasts are collected and summarized, and so on. This allows the user to see, to a certain extent, how the furthest predictions in time compare with the closest ones in time.



***Fig .2 Basic Configuration Forecast Plugin***

**Advanced configuration (Fig .3)**, by selecting this tab the user has full control over the forecast. It is, in fact, possible to choose the model, the parameters, the creation of lagged variables, the creation of variables derived from a date and time, the specification of the "overlay" data, the evaluation options and the control of the output created. Each of these features has a dedicated secondary panel, described as follows and shown in the next figure.

* **Base learner**, is the tab that provides control over which learning algorithm to use to model the time series and allows the user to configure specific parameters for the selected learning algorithm.



***Fig .3 Advanced Configuration Forecast Plugin***

* **Lag creation**, allows the user to control the lag on variables.
* **Periodic attributes**, allows the user to customize the periodic attributes derived from the timestamp. This functionality is only available if the data contains a date and time stamp. If the timestamp is a date, some default values are set automatically
* **Overlay data** , allows the user to specify, if any, an overlap between the data.
* **Evaluation**, allows the user to select the evaluation mode of the algorithm used. The available metrics are:

1. Mean absolute error (MAE)
2. Mean squared error (MSE)
3. Root mean squared error (RMSE)
4. Mean absolute percentage error (MAPE)
5. Direction accuracy (DAC)
6. Relative absolute error (RAE)
7. Root relative squared error (RRSE)

* **Output**, allows you to indicate what type of output you want to have if numeric or graphical.

# Operations performed and results

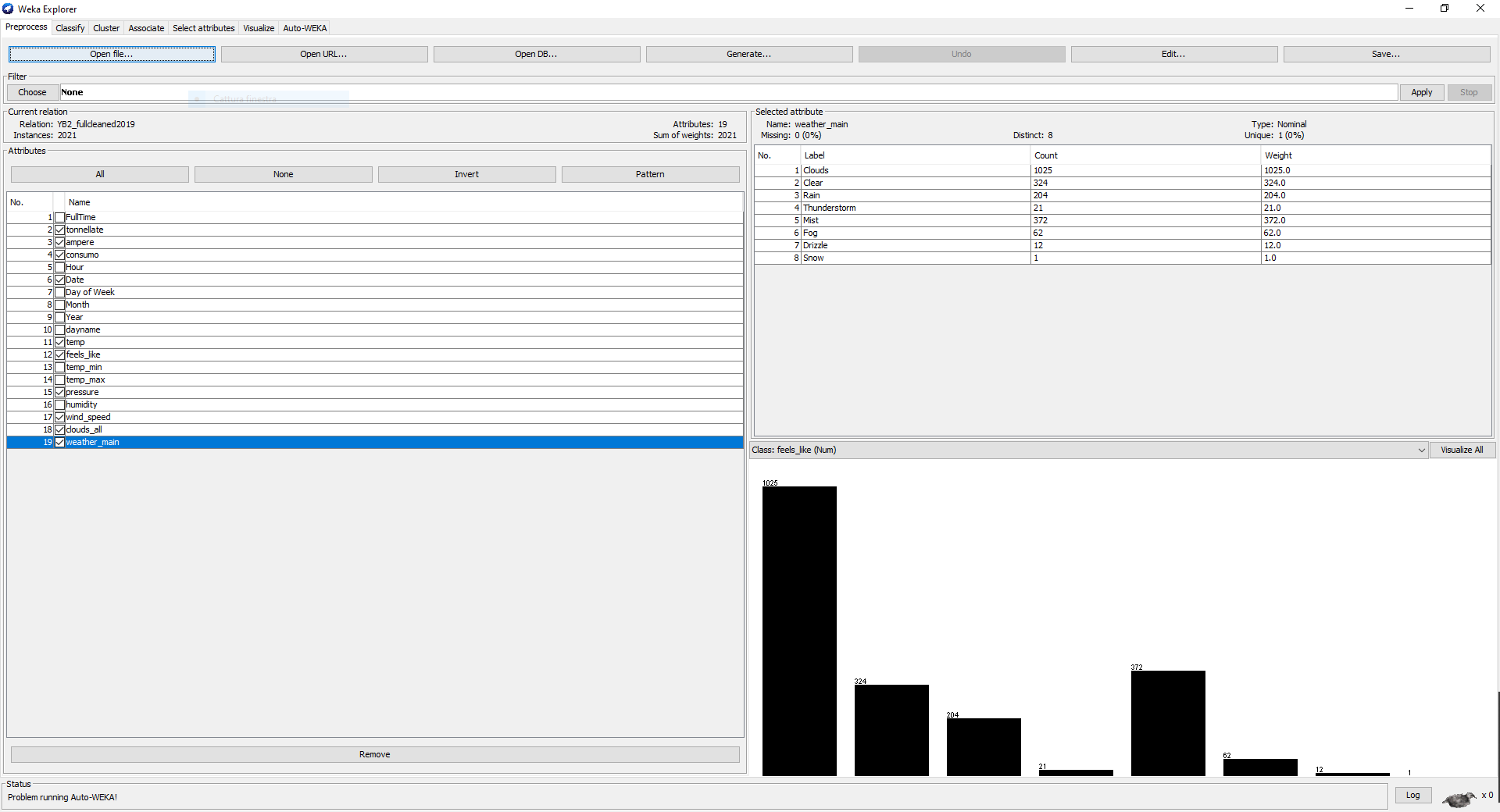
## Creating the file to analyze

To use weka, as described above, it is necessary to convert the dataset from .csv to .arff format. In addition, in order to detect any periodicity, seasonality, the data for the year 2019 and 2020 for the "YB2" plant have been combined. For the union and the transformation in .arff of the two datasets it is possible to use two roads:

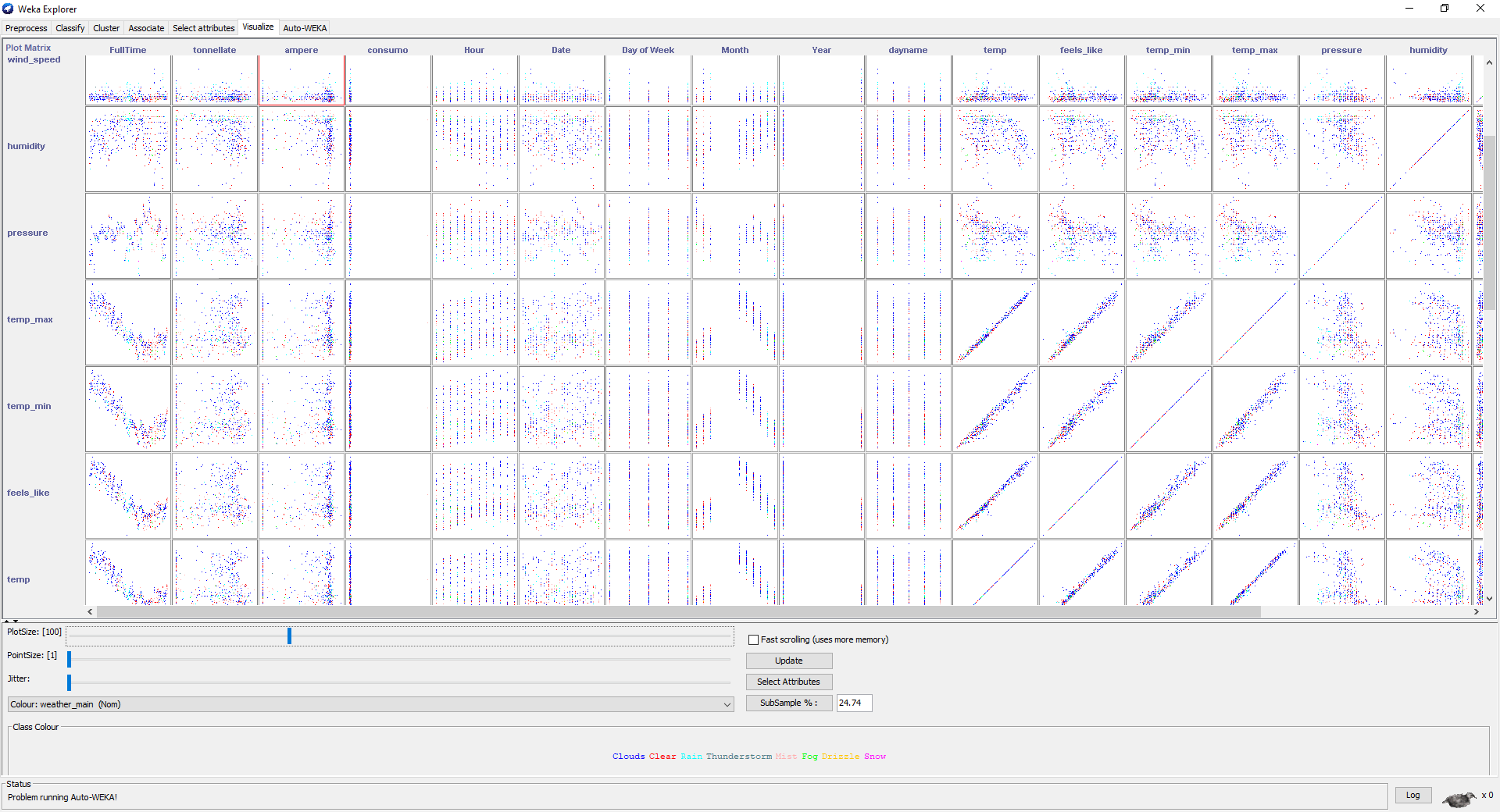
1. manually merge the two .csvs, and then import the Weka using the **AfterViewer** tool for csv conversion → arff (in the conversion pay particular attention to replacing the ";" with ",")
2. automatically use a script in python that allows the conversion csv → arff and then merge the two files (in appendix there is an example code for the two operations)

The first solution is the simplest but the second one involves the possibility to convert files at runtime and to process them in real time without having to intervene manually.

After loading the dataset, Weka automatically shows the available attributes from which you can select features by displaying statistics and classes (Fig .4). Moreover, by clicking "visualiza all" (Fig .5) it is possible to see the relative distributions of each parameter with respect to the others.

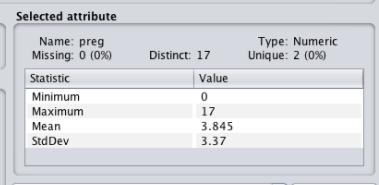
******

***Fig .4 Dataset "YB2" in weka***

******

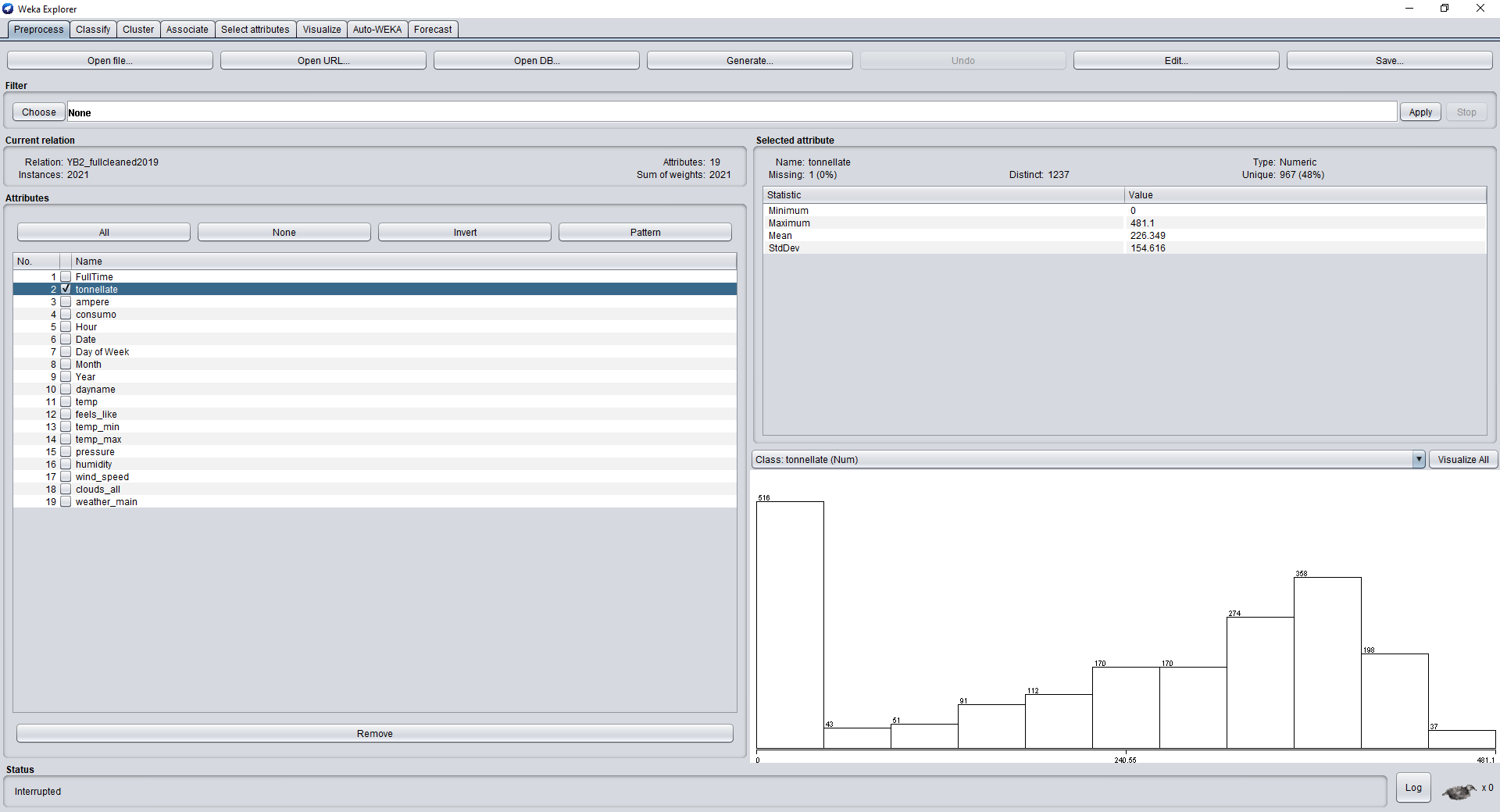
***Fig .5 Comparison of all the variables of the Dataset "YB2" in weka***

In addition, by selecting each variable in the box on the left it is possible to obtain (Fig.6) a series of information related to that variable: name, type, % of null values, minimum and maximum value, etc..



***Fig .6***

## Detection of outliers and zero values



***Fig .7 "YB2 Dataset".***

From the data analysis carried out also during the construction of the dataset it is evident a continuous oscillation of the values, around a reference value and the presence of many null values that prescind from closing times, holidays and non-working days.

Below are some days in which abnormal trends were found, on these days should be questioned the operators of the plant in order to understand if the anomalies are due to HW malfunctions or unexplained management of the plant or weather conditions.

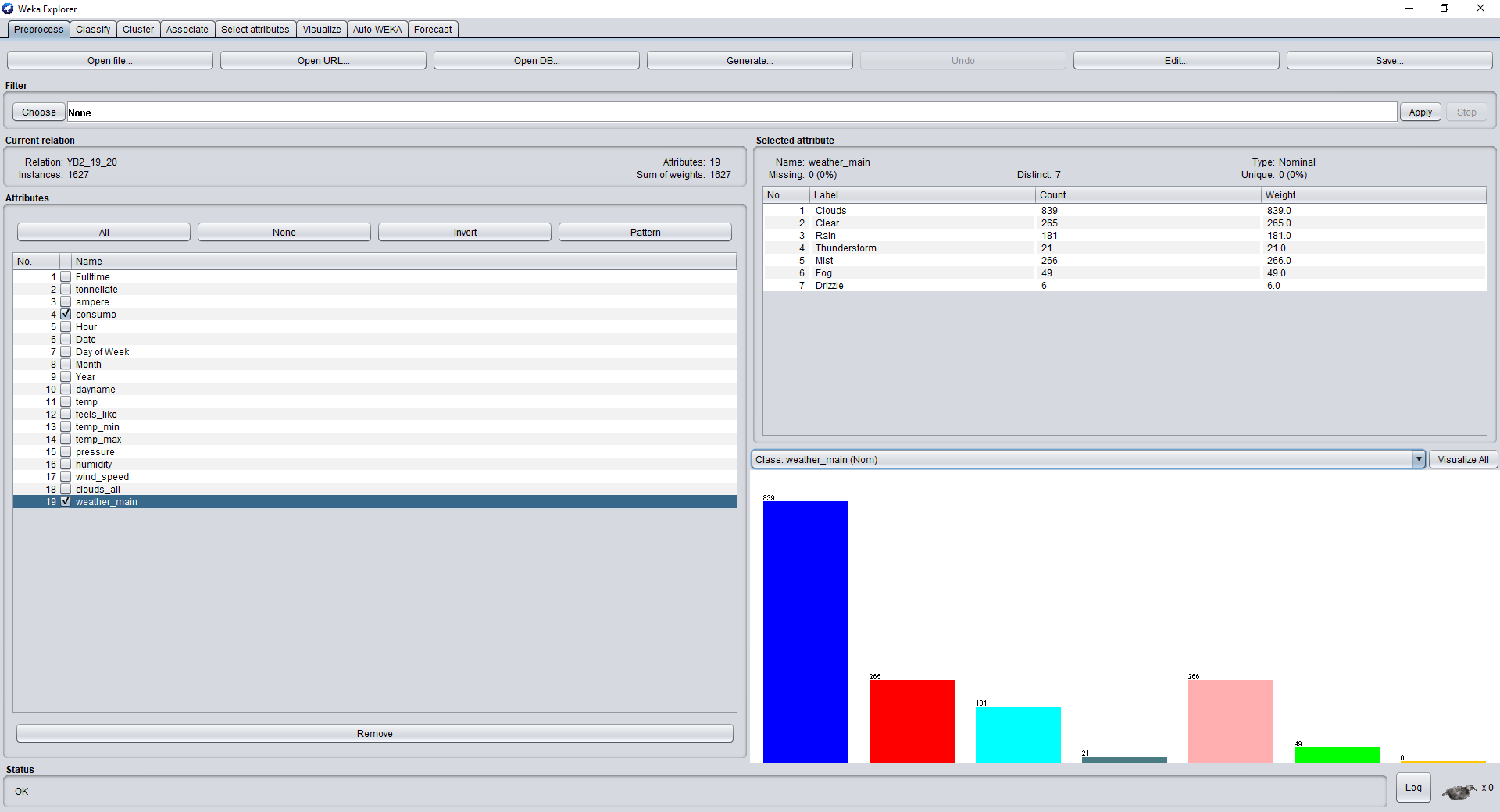
|  |  |  |
| --- | --- | --- |
| **Date/Time** | **Data** | **Possible Causes** |
| August 16, 2019 from 14:00:00 pm | all null | * activity interruption * HW malfunction * weather change from clear to cloudy |
| 2019-08-20 12:00:00 to 2019-08-21 17:00:00:00 | all null | * activity interruption * HW malfunction * cloudy |
| 2019-08-23 06:00:00 to 2019-08-26 06:00:0 | all null | * activity interruption * HW malfunction * cloudy |
| 2019-09-05 from 06:00:00 to 11:00:00:00 | all null | * activity interruption * HW malfunction * meetings * weather change from clear to cloudy |
| from 2019-09-13 15:00:00 (Friday) to 2019-09-16 17:00:00 (Monday) | all null | * activity interruption * HW malfunction * variable weather |
| 2019-09-30 (Monday) | all null | * activity interruption * HW malfunction * variable weather |
| 2019-10-02 (Wednesday) | all null | * activity interruption * HW malfunction * variable weather |
| From 2019-10-15 14:00:00 to 2019-10-24 16:00:00:00 | Ampere and consumption reading problems.  Following 0 alternating and a value absolutely out of altitude for the consumption (240.3) the situation stabilizes 2019-10-25 11:00:00 | * malfunction * variable weather |
| from 2019-10-30 06:00:00 (Wednesday) 2019-10-30 17:00:00:00 | all null | * activity interruption * HW malfunction * cloudy |
| 2019-11-04 06:00:00 (Monday) for half day | all null | * activity interruption * HW malfunction * meetings * variable weather |
| from 2019-11-06 16:00 (Wednesday) until 7am the following Friday | all null | * activity interruption * HW malfunction * variable weather |
| from 2019-12-05 06:00:00 to 2019-12-05 17:00:00 (Thursday) | all null | * activity interruption * HW malfunction * peaceful |
| 2019-12-13 06:00:00 all day (Friday) | all null | * activity interruption * HW malfunction * rain |
| Tuesday 2019-12-24 06:00:00 all day long | all null | * activity interruption * HW malfunction * peaceful/cloudy |
| from 2020-01-15 13:00:00 to 2020-01-16 17:00:00:00 | some parameters are null and others are not | * HW malfunction * cloudy |
| from 2020-02-13 06:00:00 close until 2020-02-17 16:00:00 | all null | * activity interruption * HW malfunction * Covid 19 * variable weather |
| 2020-02-20 11:00:00 to 2020-02-26 06:00:00:00 | all null | * activity interruption * HW malfunction * Covid 19 * variable weather |
| 2020-03-09 06:00:00 to 2020-03-10 17:00:00:00 | all null | * activity interruption * HW malfunction * Covid 19 * variable weather |

***Tab .2 "Data problems"***

From this post processing of the dataset some considerations on the data are evident:

* It would seem that there is a correlation between the closure of the plants and the cloudiness condition, the weather conditions should be verified in a timely manner and should be clustered (as will be shown later) in order to correlate correctly with consumption trends;
* covid conditions probably became apparent prior to the declared suspension of activities;
* measurement malfunctions exist, probably due to the sensor technology used.

## Weather labeling



***Fig .8 Weather Labeling***

In Fig .8 you can see which colors are associated with the weather conditions described in the variable "Weather\_main" of the dataset. Below are the values used for the classification. Also note that in Fig .9 are reported the trend of the measurement of consumption, amperes and tons in relation to the weather condition.

***blue→ cloud***

***red→ serene***

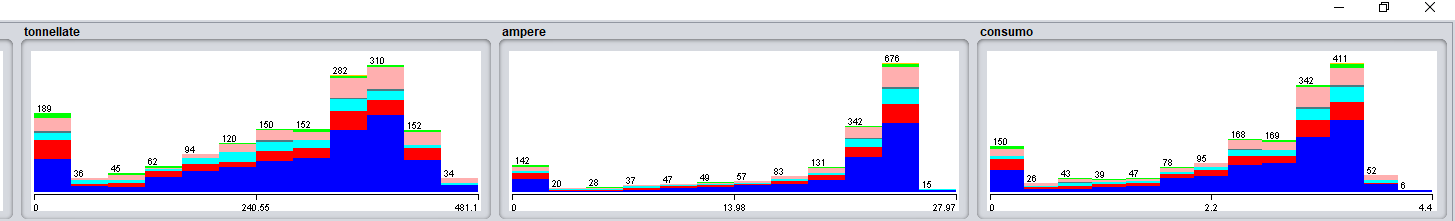
***blue→ rain***

***green→ rain and wind***

***pink→ fog***

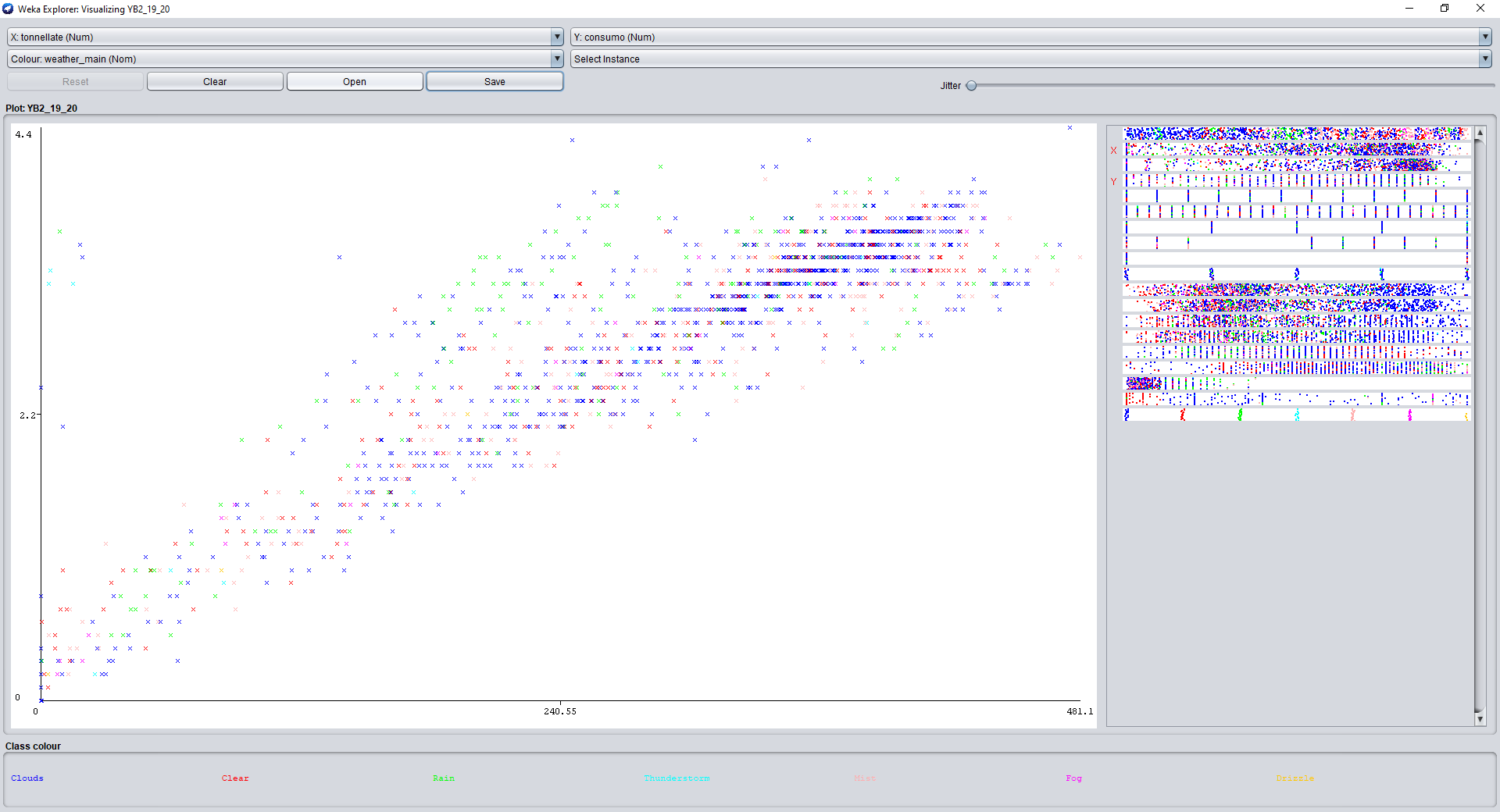
***fluo green→ intense fog***

***yellow → drizzle***

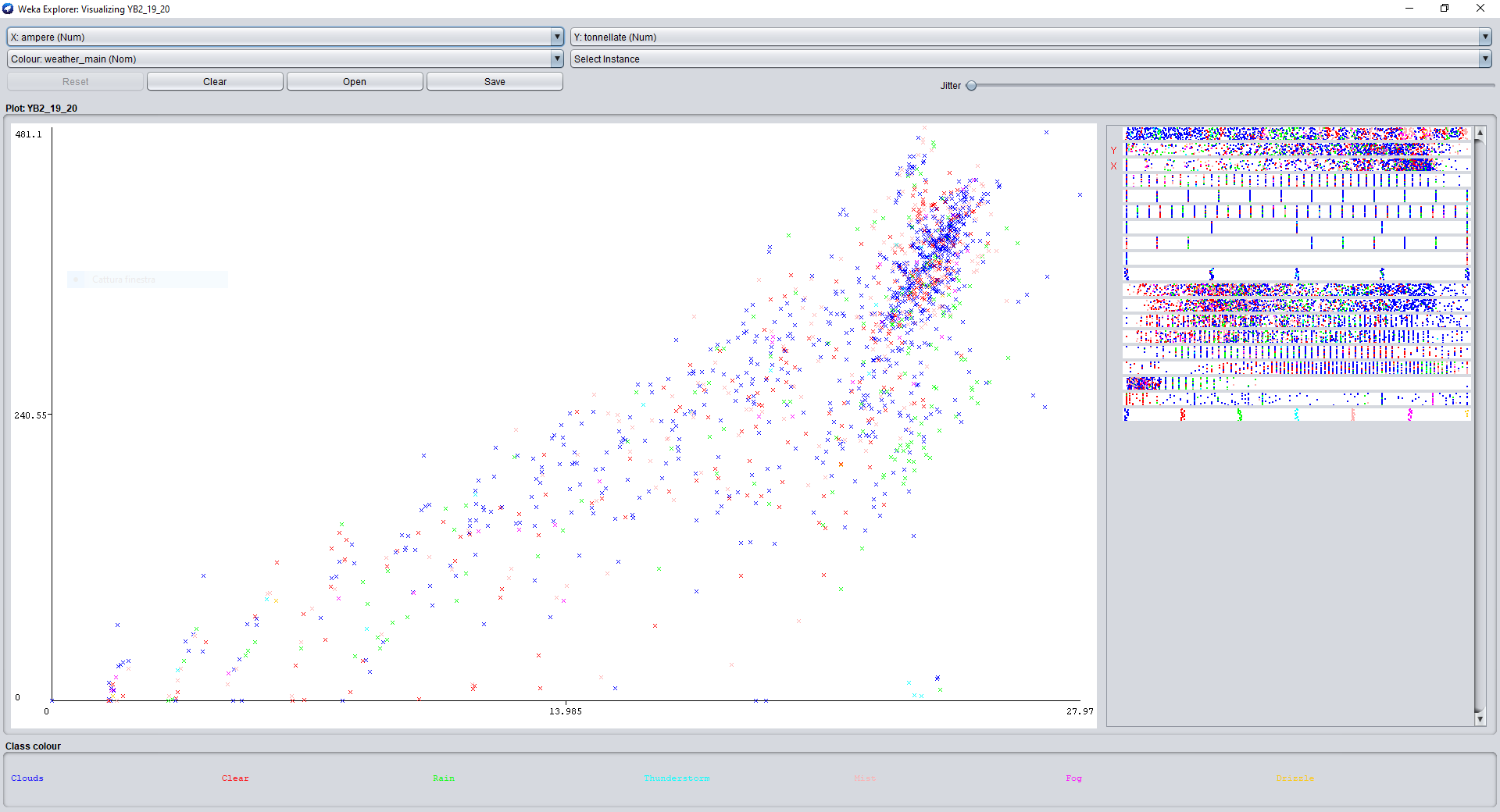


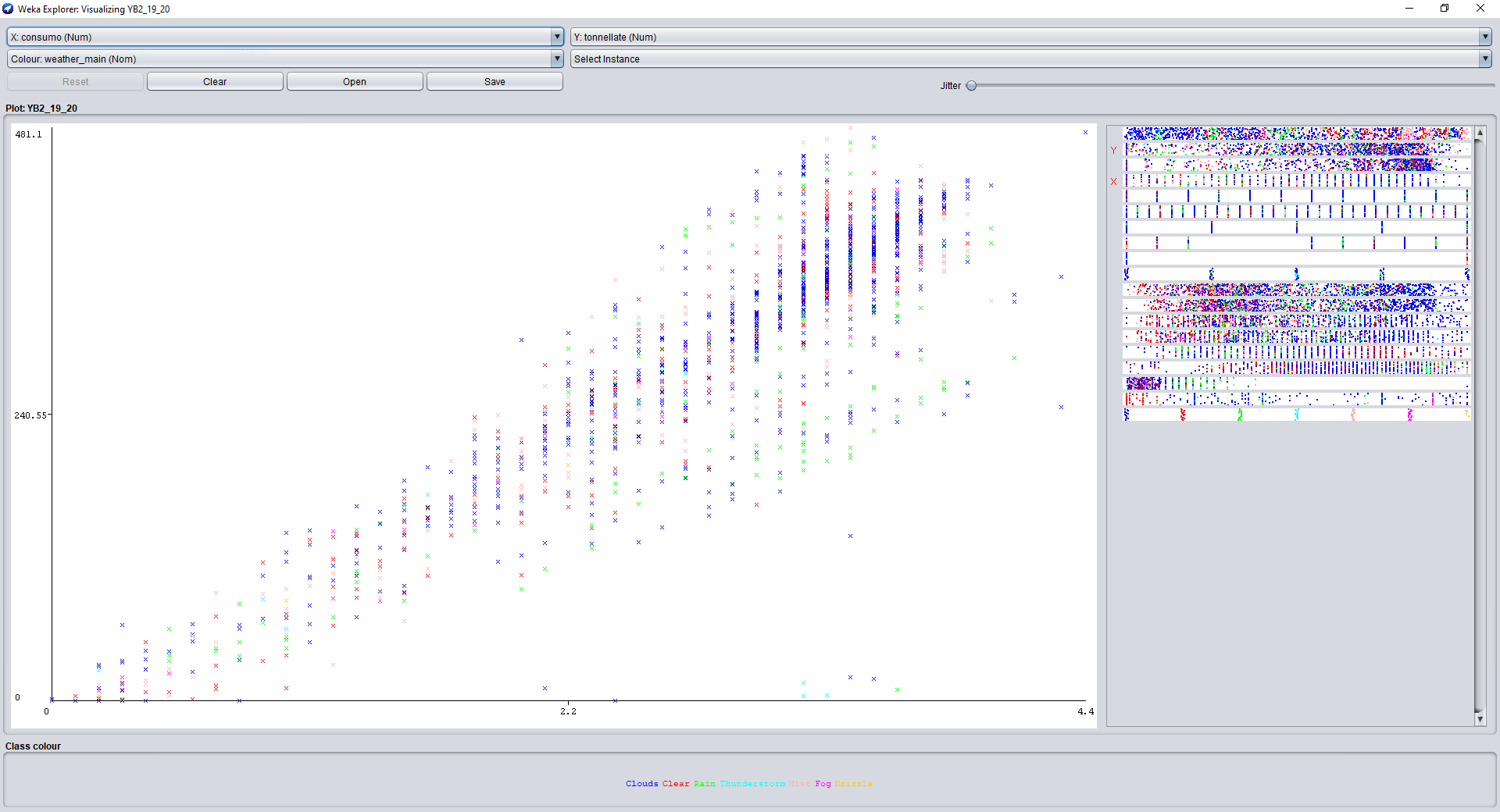
***Fig .9 Consumption Vs weather, Ampere Vs weather, Tons Vs weather***

As it is imaginable the consumption-ampere-tonne variables are closely correlated, as shown in Fig 10,11 and 12.



***Fig .10 Ampere vs Consumption***

***Fig .11 Tons vs Amperes***



***Fig .12 Tons vs Consumption***

The variable "Consumption" from the studied dataset is affected by fewer outliers and contains fewer null values, so in the following one it will be used as target.

Zero values from initial dataset:

**Consumption→ 3968**

**ampere→ 4207**

**tons→ 4119**

# Data analysis

In the following we present the models used and the results obtained. The data have been analyzed considering the correlation between the weather variables and the consumption values produced by the plant.

We report the results of the forecast obtained on daily times from *06:00:00 to 17:00:00, obtained with a ski for the night hours and with unitary lag*.

The choice of models was made based on the results obtained by using another weka plugin, "Auto-Weka", which allows, among other things, the comparison between different models according to the expected features.

The following predictions have been made

* a temperature forecast based on the exclusive use of numerical meteorological data.
* a temperature forecast based on the use of numerical weather data and weather labeling. This forecast has been implemented to verify the forecasting effectiveness in the inclusion of a label.
* a consumption forecast based on the use of numerical weather parameters
* a consumption forecast based on the use of weather labs

## SMOreg[[2]](#footnote-2)

SMOreg (Sequential Minimal Optimization) is a machine learning algorithm for SVM (Supporting Vector Machines) that implements approximations and can also be used to make forecasts on historical series. Weka uses PolyKernel or Polynomial Kernel as its kernel.

LSTM (Long Short Term Memory) neural networks are considered the most suitable types of neural networks for the prediction from historical series but to use them it is necessary to have a long time series and more complex HW resources. Having a limited number of data and wanting to use a classic supervised algorithm that could reach acceptable levels of accuracy and with a computational cost much lower than that of an LSTM.

It should be remembered that:

* SMOreg returns even negative results and does not allow the constraint of positivity (constraint present in our case).
* Where it is possible to use a variable labeled on time, it is better to use SMO, which adapts well to textual variables and allows you to insert positivity constraints.

## Random Forest[[3]](#footnote-3)

Normally this type of algorithm is not used for the analysis of time series, but by adapting some parameters it is possible to obtain valuable information from this type of models (many examples of use are reported in the literature) especially when both has to do with datasets with considerable variability and you want to make short-term predictions (as in the case of Ema Control). In general Random Forest is an improvement of Bagging. It uses a tree learning algorithm that selects, at each division of the candidate in the learning process, a random subset of the characteristics, for the analysis of the historical series is fundamental the choice of parameters and the definition of the *ntree* and the number of variables sampled in each subdivision candidate.

With weka you can configure some model parameters, as shown in Fig .13:

|  |  |
| --- | --- |
|  |  |
| ***Fig .13 Random Forest configuration*** | |

In addition to the parameters for bagging, as shown in the previous figure there is a key parameter that is the number of attributes to consider for the split point. In Weka this parameter is controlled by numFeatures attribute, which is set to 0 by default.

## Perceptron Multilayer[[4]](#footnote-4)

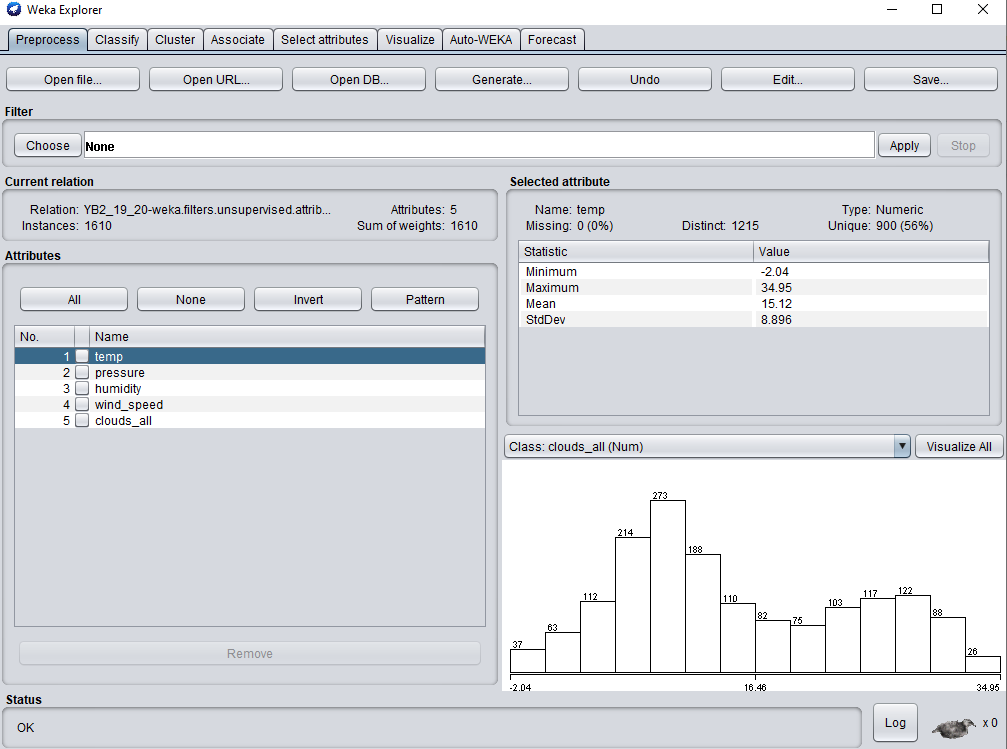
Also in this case it is possible to set the parameters related to the model in a quite punctual way, as shown in Fig .14. In particular Weka helps considerably in the creation of "characteristic" vectors, based on lag and in the a priori definition of the training and testing dataset.

## 

***Fig .14 MLP configuration***

## Temperature forecast from numerical values

The features used as first approach, are exclusively the weather features not labellized (numeric only), as shown in Fig.15. Below we report the results obtained, from what we have seen in the case in which labeling systems are not used, the best algorithm is the Random Forest even if it presents remarkable values for the errors.



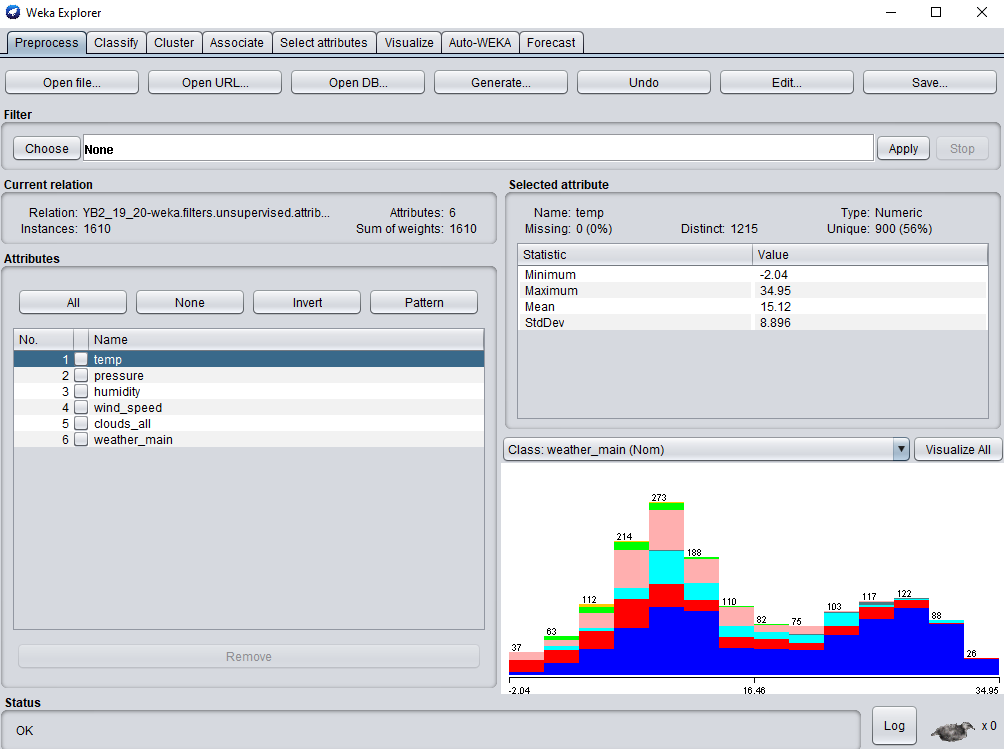
***Fig .15***

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **Mean Absolute Error** | **Root Mean Squared Error (RMSE)** |
| Random Forest | 3.75 | 5.22 |
| Smoreg | 6.52 | 7.97 |
| MLP | 5.92 | 7.63 |

## 

## 

## Forecast with labeling insertion for weather conditions



***Fig .16***

Below are the results obtained using the models previously shown. The most relevant result is the one related to the use of the SMO algorithm, from which one would have expected a better classification of the Random Forest, which instead as reported below has not been found. A reason for this trend could be dictated by the type of weather classification inserted. In the course of the project the nomenclature used and the conditions in which it is inserted should be verified. In particular, it has been noted that by decreasing and defining in a more univocal way the weather variables introduced, better results will be obtained.

#### RandomForest

Specifically the decision tree adopted the performance has been calculated using metric evaluation including parameters such as accuracy, precision, FP rate, TP rate, F-measure and ROC area.

Time taken to test model on test split: 0.03 seconds

=== Summary ===

Correctly Classified Instances 433 82.159 %

Incorrectly Classified Instances 114 17.841 %

Kappa statistic 0.6869

Mean absolute error 0.0876

Root mean squared error 0.2088

Relative absolute error 40.3008 %

Root relative squared error 67.93 %

Total Number of Instances 547

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class

0.861 0.116 0.892 0.861 0.876 0.744 0.948 0.950 Clouds

0.925 0.018 0.915 0.925 0.920 0.903 0.995 0.973 Clear

0.641 0.031 0.732 0.641 0.683 0.646 0.903 0.590 Rain

0,000 0,000 ? 0,000 ? 0.925 0.040 Thunderstorm

0.605 0.104 0.484 0.605 0.538 0.458 0.876 0.545 Mist

0.600 0.021 0.522 0.600 0.558 0.542 0.980 0.536 Fog

0.000 0.002 0.000 0.000 0.000 -0.004 0.853 0.114 Drizzle

Weighted Avg. 0.792 0.083 ? 0.7 92 ? ? 0,941 0,831

=== Confusion Matrix ===

a b c d e f g <-- classified as

248 0 6 0 32 2 0 | a = Clouds

0 86 3 0 4 0 0 0 | b = Clear

11 3 41 0 8 1 0 | c = Rain

2 0 0 0 0 0 0 | d = Thunderstorm

15 5 4 0 46 5 1 | e = Mist

1 0 2 0 5 12 0 | f = Fog

1 0 0 0 3 0 | g = Drizzle

#### SMO

=== Summary ===

Correctly Classified Instances 394 72.0293 %

Incorrectly Classified Instances 153 27.9707 %

Kappa statistic 0.5455

Mean absolute error 0.2103

Root mean squared error 0.3113

Relative absolute error 111.1051 %

Root relative squared error 101.2644 %

Total Number of Instances 547

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class

0.892 0.317 0.758 0.892 0.820 0.592 0.796 0.740 Clouds

0.978 0.031 0.867 0.978 0.919 0.904 0.983 0.862 Clear

0.156 0.006 0.769 0.156 0.260 0.317 0.735 0.283 Rain

0,000 0,000 ? 0,000 ? 0 .681 0.006 Thunderstorm

0.474 0.115 0.400 0.474 0.434 0.335 0.808 0.321 Mist

0,000 0,000 ? 0,000 ? 0.867 0.179 Fog

0,000 0,000 ? 0,000 ? 0.459 0.007 Drizzle

Weighted Avg. 0.720 0.189 ? 0.720 ? ? 0,822 0,621

=== Confusion Matrix ===

a b c d e f g <-- classified as

257 0 2 0 29 0 0 0 | a = Clouds

1 91 0 0 1 0 0 0 | b = Clear

40 4 10 0 10 0 0 | c = Rain

2 0 0 0 0 0 0 | d = Thunderstorm

29 10 1 0 36 0 0 0 | e = Mist

9 0 0 0 11 0 0 0 | f = Fog

1 0 0 0 3 0 0 0 | g = Drizzle

#### Multilayer perceptron 6 nodes

=== Evaluation on test split ====

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances 415 75.8684 %

Incorrectly Classified Instances 132 24.1316 %

Kappa statistic 0.6236

Mean absolute error 0.0965

Root mean squared error 0.2253

Relative absolute error 50.9624 %

Root relative squared error 73.3006 %

Total Number of Instances 547

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class

0.882 0.208 0.825 0.882 0.852 0.678 0.920 0.928 Clouds

0.957 0.020 0.908 0.957 0.932 0.918 0.986 0.920 Clear

0.609 0.039 0.672 0.609 0.639 0.595 0.878 0.652 Rain

0,000 0,000 ? 0,000 ? 0.926 0.035 Thunderstorm

0.421 0.0 91 0.427 0.421 0.424 0.332 0.833 0.386 Mist

0.050 0.013 0.125 0.050 0.071 0.057 0.855 0.174 Fog

0,000 0,000 ? 0,000 ? 0 .358 0.007 Drizzle

Weighted Avg. 0.759 0.131 ? 0.759 ? ? 0,908 0,782

=== Confusion Matrix ===

a b c d e f g <-- classified as

254 0 9 0 25 0 0 0 | a = Clouds

0 89 0 0 4 0 0 0 | b = Clear

14 4 39 0 6 1 0 | c = Rain

2 0 0 0 0 0 0 | d = Thunderstorm

28 5 8 0 32 3 0 | e = Mist

9 0 2 0 8 1 0 | f = Fog

1 0 0 0 3 0 | g = Drizzle

## 

***Fig.17 Temperature trend last 6h forecast***

## Consumption forecast Vs numerical weather features

As before, we report the results of the models obtained using numerical weather variables and consumption as a target.

#### RF

=== Run information ===

Test mode: split 66.0% train, remainder test

=== Classifier model (full training set) ====

RandomForest

=== Summary ===

Correlation coefficient 0.6654

Mean absolute error 0.8046

Root mean squared error 1,053

Relative absolute error 90.9273 %

Root relative squared error 93.4846 %

Total Number of Instances 547

=== Evaluation on training data ===

Target 1-step-ahead 2-steps-ahead 3-steps-ahead 4-steps-ahead 5-steps-ahead 6-steps-ahead 7-steps-ahead 8-steps-ahead 9-steps-ahead 10-steps-ahead 11-steps-ahead 12-steps-ahead

=============================================================================================================================================================================================================

consumption

N 1595 1594 1593 1592 1591 1590 1589 1588 1587 1586 1585 1584

Mean absolute error 0.2432 0.4075 0.4769 0.5226 0.5645 0.6017 0.6431 0.6742 0.7049 0.7281 0.748 0 .7682

Root mean squared error 0.3263 0.6433 0.7273 0.7626 0.7985 0.8392 0.8786 0.9094 0.9438 0.9648 0.9855 1.0053

Total number of instances: 1608

#### SMoreg

=== Run information ===

Test mode: split 66.0% train, remainder test

== Run information ==

Test mode: split 66.0% train, remainder test

=== Classifier model (full training set) ====

SMOreg

weights (not support vectors):

+ 0.0048 \*(normalized) temp

- 0.0742 \*(normalized) pressure

- 0.1276 \*(normalized) humidity

+ 0.0279 \* (normalized) wind\_speed

+ 0.0278 \* (normalized) clouds\_all

+ 0.801

Number of kernel evaluations: 7327026 (59.378% cached)

Time taken to build model: 0.81 seconds

=== Evaluation on test split ====

Time taken to test model on test split: 0 seconds

=== Summary ===

Correlation coefficient 0.1431

Mean absolute error 0.8365

Root mean squared error 1.2105

Relative absolute error 94.5327

Root relative squared error 107.4694 %

Total Number of Instances 547

=== Evaluation on training data ===

Targe t 1-step-ahead 2-steps-ahead 3-steps-ahead 4-steps-ahead 5-steps-ahead 6-steps-ahead 7-steps-ahead 8-steps-ahead 9-steps-ahead 10-steps-ahead 11-steps-ahead 12-steps-ahead

=============================================================================================================================================================================================================

consumption

N 1595 1594 1593 1592 1591 1590 1589 1588 1587 1586 1585 1584

Mean absolute error 0.671 0.7738 0.7883 0.791 0.7912 0.7 912 0.7917 0.7924 0.7942 0.794 0.799 0.7987

Root mean squared error 0.9431 1.103 1.1388 1.1468 1.1488 1.1489 1.147 1.1455 1.1424 1.1372 1.132 1.1332

#### MPL

Scheme: weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Relation: YB2\_19\_20-weka.filters.unsupervised.attribute.Remove-R1-3,5-8,10,12-14,19

Instances: 1608

Attributes: 6

consumption

temp

pressure

humidity

wind\_speed

clouds\_all

Test mode: split 66.0% train, remainder test

=== Classifier model (full training set) ====

Linear Node 0

Inputs Weights

Threshold -0.11474088523856105

Node 1 0.5285380565780566

Node 2 1.8537930202604485

Node 3 1.4968279358298788

Sigmoid Node 1

Inputs Weights

Threshold -6.082754789449108

Attrib temp 7.388051407978836

Attrib pressure -2.919651466029679

Attrib humidity 3.269167837548918

Attrib wind\_speed 2.8125476950751676

Attrib clouds\_all -0.5414900100290666

Sigmoid Node 2

Inputs Weights

Threshold -5.600041601402829

Attrib temp -0.4240051675261382

Attrib pressure -1.5458015520485084

Attrib humidity -0.7804181390728443

Attrib wind\_speed 0.6007773855731061

Attrib clouds\_all -0.15707437755358708

Sigmoid Node 3

Inputs Weights

Threshold -4.234227861415871

Attrib temp -0.17215990932724853

Attrib pressure -0.3806446386867322

Attrib humidity -2.094397066136712

Attrib wind\_speed -2.206165921811553

Attrib clouds\_all 0.3704636101704999

Class

Input

Node 0

Time taken to build model: 0.38 seconds

=== Evaluation on test split ====

Time taken to test model on test split: 0 seconds

=== Summary ===

Correlation coefficient 0.1856

Mean absolute error 0.8303

Root mean squared error 1.2071

Relative absolute error 93,823

Root relative squared error 107.166 %

Total Number of Instances 547

=== Evaluation on training data ===

Targe t 1-step-ahead 2-steps-ahead 3-steps-ahead 4-steps-ahead 5-steps-ahead 6-steps-ahead 7-steps-ahead 8-steps-ahead 9-steps-ahead 10-steps-ahead 11-steps-ahead 12-steps-ahead

=============================================================================================================================================================================================================

consumption

N 1595 1594 1593 1592 1591 1590 1589 1588 1587 1586 1585 1584

Mean absolute error 0.5735 0.7665 0.8458 0.8 832 0.9074 0.9461 1.0149 1.1481 1.2754 1.3131 1.3686 1.3601

Root mean squared error 0.8289 1.1323 1.2525 1.3175 1.3545 1.5073 1.8001 2.3478 2.7834 2.7749 2.8526 2.798

Total number of instances: 1608

## Consumption vs weather labeling

The following table shows the summary of the results obtained using the 3 algorithms with 2 types

of valuation metrics and with five (6) step-heads, the details of the transactions are shown below

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Used Algorithm | Training Data Evaluation | | | | | | |
| Metrics | 1-S-A | 2-S-A | 3-S-A | 4-S-A | 5-S-A | 6-S-A |
| Random Forest | MAE | 0,24 | 0,41 | 0,47 | 0,53 | 0,56 | 0,61 |
| RMSE | 0,33 | 0,64 | 0,72 | 0,77 | 0,79 | 0,83 |
| SMO | MAE | 0,67 | 0,77 | 0,79 | 0,79 | 0,79 | 0,79 |
| RMSE | 0,94 | 1,03 | 1,09 | 1,14 | 1,15 | 1,15 |
| MPL | MAE | 0,57 | 0,76 | 0,84 | 0,88 | 0,9 | 0,94 |
| RMSE | 0,82 | 1,14 | 1,25 | 1,31 | 1,35 | 1,51 |

#### 

#### RF

=== Summary ===

Correlation coefficient 0.672

Mean absolute error 0.3762

Root mean squared error 0.5227

Relative absolute error 20,0179

Root relative squared error 20.6777 %

=== Evaluation on training data ===

Targe t 1-step-ahead 2-steps-ahead 3-steps-ahead 4-steps-ahead 5-steps-ahead 6-steps-ahead 7-steps-ahead 8-steps-ahead 9-steps-ahead 10-steps-ahead 11-steps-ahead 12-steps-ahead

=============================================================================================================================================================================================================

consumption

N 1595 1594 1593 1592 1591 1590 1589 1588 1587 1586 1585 1584

Mean absolute error 0.2432 0.4075 0.4769 0.5226 0.5645 0.6017 0.6431 0.6742 0.7049 0.7281 0.748 0.7682

Root mean squared error 0.3263 0.6433 0.7273 0.7626 0.7985 0.8392 0.8786 0.9094 0.9438 0.9648 0.9855 1.0 053

Total number of instances: 1608

### SMO

Time taken to test model on test split: 0.03 seconds

=== Summary ===

Correlation coefficient 0.6973

Mean absolute error 0.8762

Root mean squared error 1. 1227

Relative absolute error 20,0179

Root relative squared error 20.6777 %

=== Evaluation on training data ===

Targe t 1-step-ahead 2-steps-ahead 3-steps-ahead 4-steps-ahead 5-steps-ahead 6-steps-ahead 7-steps-ahead 8-steps-ahead 9-steps-ahead 10-steps-ahead 11-steps-ahead 12-steps-ahead

=============================================================================================================================================================================================================

consumption

N 1595 1594 1593 1592 1591 1590 1589 1588 1587 1586 1585 1584

Mean absolute error 0.671 0.7738 0.7883 0.791 0.7912 0.7912 0.7917 0.7924 0.7942 0.794 0.799 0.7987

Root mean squared error 0.9431 1.103 1.1388 1.1468 1.1488 1.1489 1.147 1.1455 1.1424 1.1372 1.132 1.1 332

Total number of instances: 1608

### MPL Classify

=== Cross-validation ===

=== Summary ===

Correlation coefficient 0.0649

Mean absolute error 1.0495

Root mean squared error 1.2418

Relative absolute error 117.9018 %

Root relative squared error 111.551 %

Total Number of Instances 1607

Ignored Class Unknown Instances 1

=== Evaluation on training data ===

Targe t 1-step-ahead 2-steps-ahead 3-steps-ahead 4-steps-ahead 5-steps-ahead 6-steps-ahead 7-steps-ahead 8-steps-ahead 9-steps-ahead 10-steps-ahead 11-steps-ahead 12-steps-ahead

=============================================================================================================================================================================================================

consumption

N 1595 1594 1593 1592 1591 1590 1589 1588 1587 1586 1585 1584

Mean absolute error 0.5735 0.7665 0.8458 0.8832 0.9074 0.9461 1.0149 1.1481 1.2754 1.3131 1.3686 1.3601

Root mean squared error 0.8289 1.1323 1.2525 1.3175 1.3545 1.5073 1.8001 2.3478 2.7834 2.7749 2.8526 2.7 98

Total number of instances: 1608

The analysis performed so far shows that the algorithm that offers the best performance is the random forest applied to consumption and labelled weather data (the same result is obtained using numerical weather data and consumption), at the same time the previous analysis made on numerical weather data and labeling shows that the labeling is not always well correlated with the weather data. It is therefore necessary to determine a better method of labeling.

The result obtained was predictable, this type of algorithm is in fact very efficient in short term forecasts and with high variability as in the case of the emacontrol platform (hourly forecasts).

The random Forest is an ensemble learning method that generates many regression trees (CART) and aggregates their results. The model applies well on seasonal time series cycles and simplifies the problem of forecasting especially when a time series shows non-steadiness, heteroschedasticity, tendency and multiple seasonal cycles. The main advantages of the model are its ability to generalize, built-in cross-validation and low sensitivity to parameter values.

It would be appropriate to have more data, at least two years, to test seasonal algorithms such as Arima or otherwise try to test the above algorithms by dividing the dataset into seasons.

In addition, with the presence of more data you could think about a daily labeling for the weather condition and for the production data of the plant and try again the forecast with models like SMO that is well suited to the result you would like to obtain, but in the absence of albel is not usable.

# Appendix

#CSV > arff

import csv

from time import sleep

class convert(object):

content = []

name = ''

def \_\_init\_\_(self):

self.csvInput()

self.arffOutput()

print 'unFinished.

#import CSV

def csvInput(self):

user = raw\_input('Enter the CSV file name: ')

#remove .csv

if user.endswith('.csv') == True:

self.name = user.replace('.csv', '')

print 'Opening CSV file.

try:

with open(user, 'rb') as csvfile:

lines = csv.reader(csvfile, delimiter = ',')

for row in lines:

self.content.append(row)

csvfile.close()

sleep(2) #sleeps added for dramatic effect!

#just in case user tries to open a file that doesn't exist

except IOError:

sleep(2)

print 'File not found.\n'.

self.csvInput()

#export ARFF

def arffOutput(self):

print 'Converting to ARFF file.\n'.

title = str(self.name) + '. arff'

new\_file = open(title, 'w')

##

#following portions formats and writes to the new ARFF file

##

#write relation

new\_file.write('@relation ' + str(self.name)+ '\n\n')

#get attribute type input

for i in range(len(self.content[0])-1):

attribute\_type = raw\_input('Is the type of ' + str(self.content[0][i]) + ' numeric or nominal? ')

new\_file.write('@attribute ' + str(self.content[0][i]) + ' ' + str(attribute\_type) + '\n')

#create list for class attribute

last = len(self.content[0])

class\_items = []

for i in range(len(self.content)):

name = self.content[i][last-1]

if name not in class\_items:

class\_items.append(self.content[i][last-1])

else:

pass

class\_items[0]

string = '{' + ','.join(sorted(class\_items)) + '}'.

new\_file.write('@attribute ' + str(self.content[0][last-1]) + ' ' + str(string) + '\n')

#write date

new\_file.write('\n@data\n')

self.content[0]

for row in self.content:

new\_file.write(','.join(row) + '\n')

#close file

new\_file.close()

sleep(2)

#####

run = convert()

#Union of two arff

from scipy.io import arff

import sys

class MergeArff:

def \_\_init\_\_(self, first\_arff, second\_arff, output):

self.files = []

self.attributes = {}

self.data = []

self.output = output

print "Reading arff files

date, meta = arff.loadarff(open(first\_arff))

self.files.append({

date': date,

goal': goal

})

date, meta = arff.loadarff(open(second\_arff))

self.files.append({

date': date,

goal': goal

})

self.calculate\_nominal\_fields()

self.merge\_data\_fields()

self.save\_as\_arff()

def calculate\_nominal\_fields(self):

#Detect nominal fields

for attribute in self.files[0]['meta'].\_attributes:

attribute\_type = self.files[0]['meta].\_attributes[attribute][0]

if attribute\_type == 'nominal':

self.attributes[attribute] = list(self.files[0]['meta].\_attributes[attribute][1])

#Merge nominal fields

for attribute in self.attributes:

merge\_fields = list(self.files[1]['meta'].\_attributes[attribute][1])

self.attributes[attribute] = list(set(self.attributes[attribute] + merge\_fields))

def merge\_data\_fields(self):

self.data = []

for row in self.files[0]['data']. tolist():

row = list(row)

for (i,value) in enumerate(row):

row[i] = str(value)

self. data.append(",".join(row))

for row in self.files[1]['data']. tolist():

row = list(row)

for (i,value) in enumerate(row):

row[i] = str(value)

self. data.append(",".join(row))

def save\_as\_arff(self):

print "Writing new arff file

new\_file = open(self.output, 'w')

#write relation

new\_file.write("@relation %s \n\n" % self.output)

#Write attributes

attributes = self.files[0]['meta'].\_attributes.keys()

attributes.sort()

for attributes in attributes:

attribute\_type = self.files[0]['meta].\_attributes[attribute][0]

if attribute\_type == 'nominal':

options = self.attributes[attribute]

new\_file.write("@attribute %s {%s}\n" % (attribute, ",".join(options)))

else:

new\_file.write("@attribute %s %s\n" % (attribute, attribute\_type))

#Write date

new\_file.write('\n@data\n')

new\_file.write('\n'.join(self.data))

new\_file.close()

def main():

MergeArff(sys.argv[1], sys.argv[2], sys.argv[3])

if \_\_name\_\_ == '\_\_main\_\_':

main()

# 

1. <https://wiki.pentaho.com/display/DATAMINING/Time+Series+Analysis+and+Forecasting+with+Weka> [↑](#footnote-ref-1)
2. <https://weka.sourceforge.io/doc.dev/weka/classifiers/functions/SMOreg.html> [↑](#footnote-ref-2)
3. <https://weka.sourceforge.io/doc.dev/weka/classifiers/trees/RandomForest.html> [↑](#footnote-ref-3)
4. <https://weka.sourceforge.io/doc.dev/weka/classifiers/functions/MultilayerPerceptron.html> [↑](#footnote-ref-4)