Deliverable A3.4

Data Rilascio:16/09/2020

[**Strumento Utilizzato**](#_rz3evkmp7bez) **2**

[Formato ARFF](#_tnh20bk30v3e) 3

[Weka e Java - Vantaggi](#_7r0s3ohnwnlz) 4

[Requirements HW eSW](#_v5szzobd7q2e) 5

[Weka e il Data Mining](#_srvxnuyalcwc) 6

[Forecasting plugin](#_6wu1cra9jkb) 7

[**Operazioni eseguite e risultati**](#_feibwod2fmm6) **11**

[Creazione del file da analizzare](#_7oe09fk3myf5) 11

[Individuazione outliers e valori nulli](#_c8mwm7bg35v1) 13

[Meteo labeling](#_hx5h1uxme6cf) 16

[**Analisi dei dati**](#_owrab8hvekor) **19**

[SIMOreg](#_y3xxqe187rfn) 19

[Random Forest](#_rc7sd56ac63l) 20

[Multilayer Perceptron](#_jgxv7cqitwbn) 23

[Previsione sulla temperatura da valori numerici](#_ylpb3ueavole) 24

[Previsione con inserimento di labeling per le condizioni meteo](#_l60qtlbbk1z9) 25

[RandomForest](#_qnl2kmc0fxbz) 25

[SMO Classify](#_qjqy456gz2oh) 26

[Multilayer perceptron 6nodi](#_v60cuvtv3plh) 27

[Previsione consumi Vs features meteo numeriche](#_ixs4896s9wp2) 28

[RF](#_9nncft66cw9w) 28

[SIMoreg](#_e0wcpxk4dpym) 29

[MPL](#_wprb0tnrwjkl) 30

[Consumo vs labeling meteo](#_4lru7snvugr8) 32

[RF](#_cypwrnw598xs) 32

[SMO](#_htr2tnnrbjmo) 33

[MPL Classify](#_5kr586pmdrwj) 33

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

In quello che segue verranno descritte le attività svolte sul dataset di partenza e il software utilizzato per la valutazione del modello previsionale.

In questa fase si è adottato, come verrà descritto in seguito, uno strumento previsionale che permettesse di poter testare il maggior numero possibile di modelli al fine di individuare quello maggiormente performante.

# Strumento Utilizzato

Weka (Waikato Environment for Knowledge Analysis) è una collezione di algoritmi di machine learning open-source che consento di effettuare preprocessing, classificazione, clustering e regressione su set di dati in maniera supervisionata e non. Nell’esecuzione degli algoritmi disponibili Weka permette il setting di parametri in modo da personalizzare l’esecuzione.

*“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.”*

La scelta fatta è ricaduta su questo strumento per le notevoli opportunità offerte:

**Portabilità:** Weka è scritto ed implementato in Java, un linguaggio interpretato, quindi per poter eseguire il programma Weka è sufficiente che sia installato nel computer l’ambiente di esecuzione Java (JRE, Java Runtime Environment).

**Gestione della memoria:** dal momento che un programma Java viene eseguito all’interno di una macchina virtuale, la gestione della memoria è controllata e gestita da un opportuno modulo, detto Garbage Collection.

**Object-oriented**.

**Documentazione Community:** Weka è dotato di una grande community che lavora al suo sviluppo e all'implementazione di nuovi plugin. Inoltre è molto semplice in rete reperire materiale per lo studio e per il suo utilizzo anche in ambiti complessi.

**API:** il codice di Weka è stato scritto secondo lo standard JSR-000073 o Java Data Mining API Specification Request al fine di garantire una facile ed efficace integrazione tra una generica applicazione e i il suo motore.

**Differenti modalità di utilizzo.** Il software Weka può essere utilizzato con modalità differenti: da linea di comando ,tramite interfaccia grafica o inserendo in progetto Java. Tutte ben documentate.

## Formato ARFF

IL formato dei dataset per Weka è **.ARFF**, Attribute Relationship File Format.

ARFF è un formato per testi che riporta al suo interno la struttura di un database. Un arff file contiene sostanzialmente due sezioni distinte: header e dati.

***L’header*** contiene i campi relativi al nome del dataset e agli attributi in esso contenuti, ad esempio:

@relation YB2\_19\_20

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

@attribute tonnellate numeric

@attribute ampere numeric

@attribute consumo numeric

@attribute Hour numeric

@attribute Date numeric

@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 humidity numeric

@attribute wind\_speed numeric

@attribute clouds\_all numeric

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

Nella parte relativa ai dati si trovano i valori veri e propri delle misure effettuate o delle grandezze osservate. Nel nostro caso:

@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 e Java - Vantaggi

Come espresso in precedenza sono molti i motivi per appoggiarsi, nelle fasi preliminari della costruzione di un modello previsionale ad un software come Weka. Va, inoltre, considerato che Weka può essere utilizzato anche in fasi più avanzate, come quella della messa in produzione dell'algoritmo, infatti è possibile l’integrazione di un package jar Weka nel progetto Java che si sta implementando, come qualsiasi altra libreria.

Dopo avere integrato il jar nel progetto è possibile importare le classi per l’utilizzo di tutte le funzionalità che Weka è in grado di fornire. In questo modo è possibile implementare un’applicazione Data Mining istanziando nel codice oggetti come classificatori, filtri e tutti gli altri strumenti presenti all’interno di Weka ed utilizzare i diversi algoritmi disponibili ad essi associati senza dover implementare il codice di volta in volta.

Di seguito si riporta un esempio di utilizzo di Weka all’interno di un progetto Java, in cui si prova ad utilizzare un classificatore Bayesiano su variabili numeriche, vettoriali e booleane e si ritornano i risultati del testing.

All’inizio della classe si importano le classi weka che si intende utilizzare e poi si richiamano e si istanziano come sempre in Java.

//Esempio di utilizzo del software Weka in un programma Java

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 Prova{

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();

}

}

}

## Requirements HW eSW

La matrice seguente mostra quale versione minima di Java è necessaria per eseguire una specifica versione di Weka. Le ultime versioni ufficiali di Weka richiedono Java 8 o versioni successive. Va notato che in casi utilizzo di una macchina Windows con un display HiDPI, potrebbe essere necessario utilizzare Java 9 o versioni successive per evitare problemi con il ridimensionamento inappropriato delle interfacce utente grafiche di Weka.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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***

Oltre le specifiche relative al Java, Weka non necessita di altri particolari requirement, ovviamente la ram a disposizione deve essere proporzionata con la mole dei dati in analisi, si consigliano comunque macchine con una ram non inferiore ai 32G.

## Weka e il Data Mining

Per il data mining weka mette a disposizioni numerosi algoritmi e tecniche che permettono una grande versatilità dal punto di vista operativo, versatilità che si aggiunge alla sua portabilità dovuta al linguaggio Java.

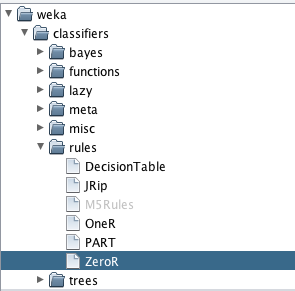
Inoltre, Weka può essere utilizzato:

* mediante una interfaccia grafica (GUI)
* da linea di comando
* come libreria importabile in applicazioni Java.

Weka effettuata una classificazione salva i modelli generati in file di estensione .model permettendone facilmente il riutilizzo in momenti successivi. Di default, senza aggiunta di plugin, nella fase di dataminig offre le seguenti funzionalità:

* Preprocessing dei dati.
* Selezione degli attributi.
* Classificazione.
* Regressione.
* Clustering.
* Regole associative.

Si riporta, in Fig. 1, a titolo esemplificativo una schermata relativa agli algoritmi disponibili in Weka:



***Fig. 1 Esempio modelli Weka***

## Forecasting plugin

Al fine di poter analizzare e prevedere al meglio i dati ottenuti dalla piattaforma Ema Control, si è integrato in Weka un plugin sviluppato appositamente per l’analisi delle serie temporali, un plugin che permette quindi di analizzare una serie di punti dati dipendente dal tempo o meglio in successione.

La previsione delle serie temporali è un processo di utilizzo di un modello per generare previsioni sulla base di eventi passati noti. I dati delle serie temporali hanno un ordinamento temporale naturale, che differisce dalle tipiche applicazioni di data mining/machine learning in cui ogni punto è un esempio indipendente da apprendere e l'ordine dei punti dati all'interno di un set di dati non ha importanza.

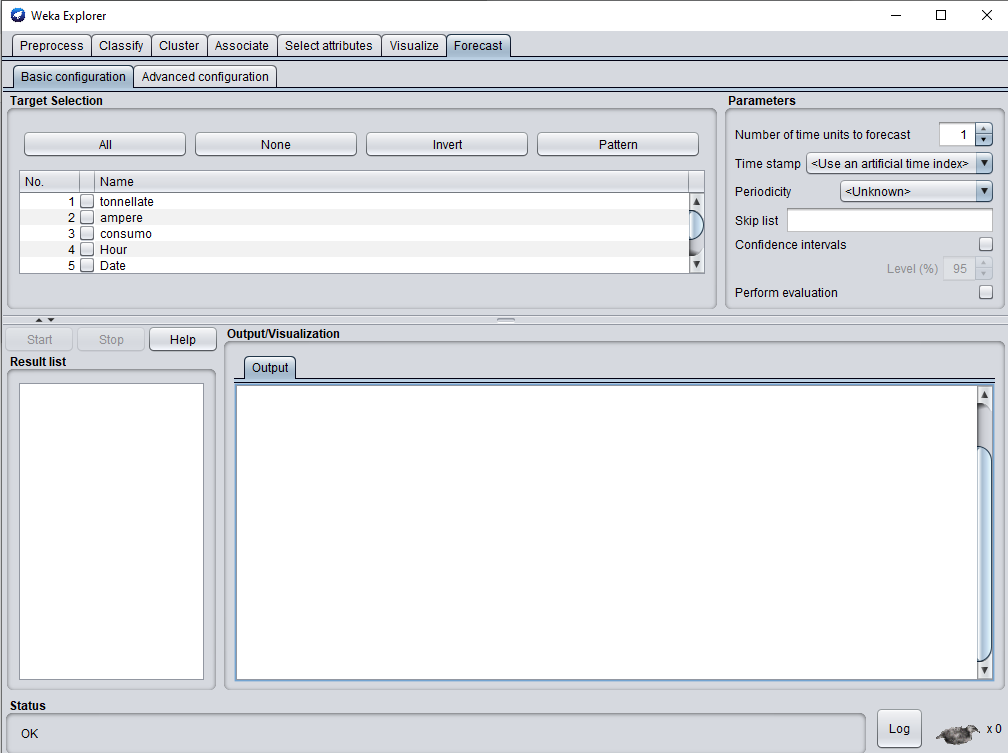
In Weka (> = 3.7.3) è possibile incorporare questo strumento per l’analisi delle serie temporali che consente di sviluppare, valutare e visualizzare modelli di previsione. Questo ambiente assume la forma di una scheda aggiunta nell'interfaccia utente grafica "Explorer" di Weka e può essere installato tramite il gestore di pacchetti.

In particolare, il plugin utilizzato adotta un approccio machine learning/data mining per modellare le serie temporali trasformando i dati in formati che gli algoritmi di apprendimento possano elaborare. L’ambiente di modellazione delle serie temporali "core" è disponibile come software gratuito open source e può essere installato seguendo le indicazioni riportate sul sito dagli sviluppatori[[1]](#footnote-1).

Il plugin permette di utilizzare una versione di base ed una versione avanzata, integrabili fra loro.

**Basic Configuration** (Fig .2), di seguito si descrivono le funzionalità presente in questa configurazione.

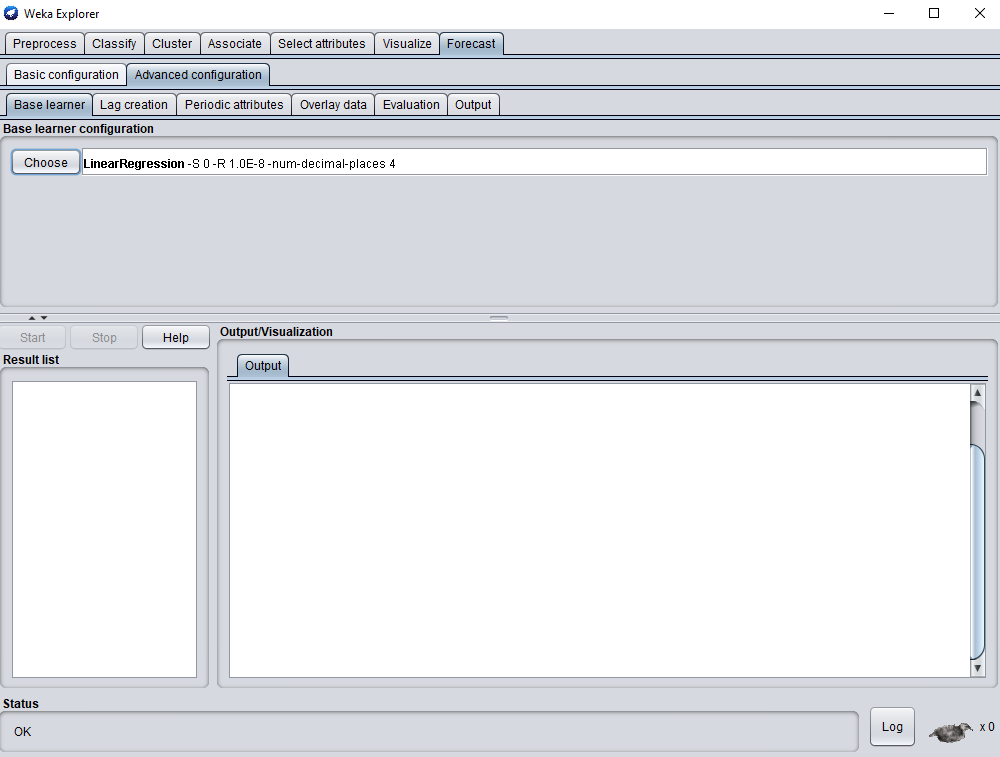
* Il box menù permette di selezionare il target in analisi;
* L’indicatore *Number of units to forecast*, definisce per quanti “passi temporali” verrà prodotta la previsione. L'impostazione predefinita è 1, ovvero il sistema eseguirà una singola previsione di 1 passo avanti.
* L’indicatore *Timestamp*, consente all'utente di selezionare quale, se presente, è il campo nei dati contiene il timestamp. Se nei dati è presente un campo data, il sistema lo seleziona automaticamente. Se non è presente alcuna data nei dati, l'opzione *"<Use an artificial time index>"* viene selezionata automaticamente. Va sottolineato che il timestamp è un campo numerico non data (perché il sistema non è in grado di distinguerlo da un potenziale target). L'utente ha anche la possibilità di selezionare *"<None>"* dalla casella a discesa per indicare al sistema che non deve essere utilizzato alcun timestamp (artificiale o meno).
* Il menù a tendina *Periodicity*, consente all'utente di specificare la periodicità dei dati. Se un campo data è stato selezionato come timestamp, il sistema può utilizzare l'euristica per rilevare automaticamente la periodicità - *"<Detect automatically>"* sarà impostato come predefinito. Se il timestamp non è una data, l'utente può indicare esplicitamente al sistema qual è la periodicità o selezionare *"<Unknow>"* se non è noto.
* La barr*a Skip List,* consente all'utente di specificare periodi di tempo che non devono essere considerati come un incremento del timestamp rispetto al processo di modellazione, previsione e visualizzazione (nel nostro caso ad esempio le ore notturne di chiusura dell'impianto). L'euristica utilizzata per rilevare automaticamente la periodicità non può far fronte a questi "buchi" nei dati, quindi l'utente deve specificare una periodicità da utilizzare e fornire i periodi di tempo che non devono essere considerati come incrementi nel campo di testo Salta elenco. Nel campo si possono inserire stringhe come "weekend", "sat", "tuesday", , date specifiche (con una stringa di formattazione opzionale) come "2011-07-04@yyyy-MM-dd "e numeri interi (che vengono interpretati in modo diverso a seconda della periodicità specificata).
* La casella di controllo e il campo numerico *Confidence intervals* permettono all’utente di inserire i limiti di confidenza sulle previsioni in svolgimento.
  + Il livello di confidenza predefinito è del 95%. Il sistema utilizza le previsioni fatte per i valori target noti nei dati di addestramento per impostare i limiti di confidenza. Quindi, un livello di confidenza del 95% significa che il 95% dei valori target reali rientra nell'intervallo. Si noti che gli intervalli di confidenza sono calcolati per ogni passo avanti in modo indipendente, cioè tutte le previsioni di un passo avanti sui dati di addestramento vengono utilizzate per calcolare l'intervallo di confidenza di un passo avanti, tutte le previsioni di due passi avanti sono utilizzato per calcolare l'intervallo di due passi avanti e così via.
* *Perform evaluation*, indica al sistema di eseguire una valutazione del previsore utilizzando i dati di addestramento. Cioè, una volta che il modello è stato addestrato sui dati, viene applicato per fare una previsione in ogni momento (in ordine) passando attraverso i dati. Queste previsioni vengono raccolte e riepilogate, utilizzando varie metriche, per ogni passo temporale futuro previsto, ovvero tutte le previsioni di un passo avanti vengono raccolte e riassunte, tutte le previsioni di due passi vengono raccolte e riepilogate e così via. Ciò consente all'utente di vedere, in una certa misura, come le previsioni più lontane nel tempo si confrontano con quelle più vicine nel tempo.



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

**Advanced configuration (Fig .3)**, selezionando questo tab l'utente ha il pieno controllo sulla previsione. E’, infatti, possibile scegliere il modello, i parametri, la creazione di variabili lagged, la creazione di variabili derivate da una data e ora, la specifica dei dati "overlay", le opzioni di valutazione e il controllo dell'output creato. Ognuna di queste funzionalità ha un pannello secondario dedicato, descritto in quanto segue e mostrato nella figura successiva.

* **Base learner**, è il tab che fornisce il controllo su quale algoritmo di apprendimento utilizzare per modellare la serie temporale e permette all'utente di configurare parametri specifici per l'algoritmo di apprendimento selezionato.



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

* **Lag creation**, permette all’utente di controllare il lag sulle variabili.
* **Periodic attributes**, consente all'utente di personalizzare gli attributi periodici derivati dal timestamp. Questa funzionalità è disponibile solo se i dati contengono un indicatore di data e ora. Se il timestamp è una data, vengono impostati automaticamente alcuni valori predefiniti
* **Overlay data** , permette all’utente di specificare,se c’è, una sovrapposizione fra i dati.
* **Evaluation**, permette all’utente di selezionare le modalità di valutazione dell’algoritmo utilizzato. Le metriche disponibili sono:

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**, permette di indicare che tipo di output si vuole avere se numerico o grafico.

# Operazioni eseguite e risultati

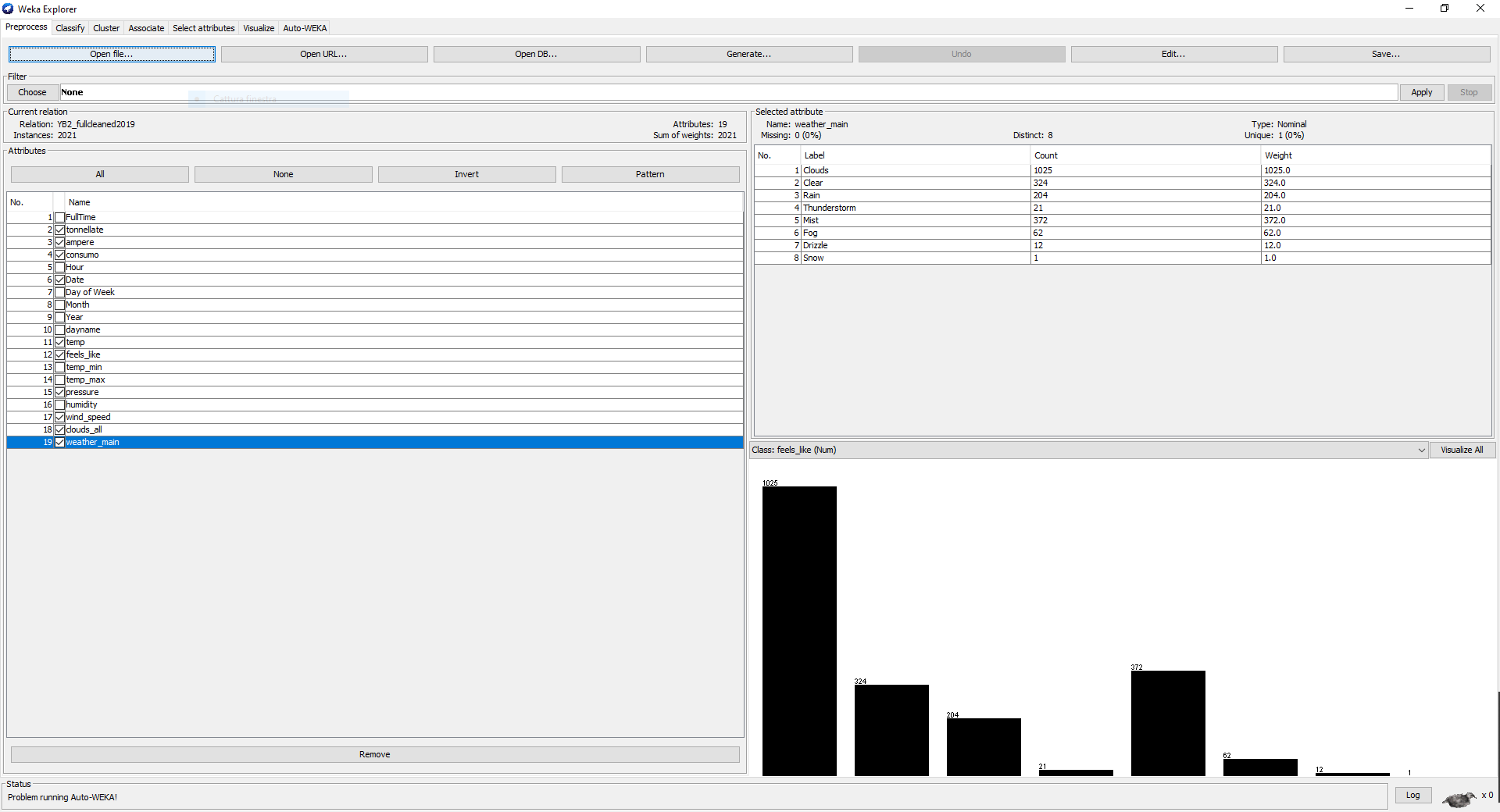
## Creazione del file da analizzare

Per l’utilizzo di weka, come descritto in precedenza, è necessario convertire il dataset da formato .csv ad .arff. Inoltre al fine di riscontrare eventuali periodicità, stagionalità, si sono uniti i dati relativi all’anno 2019 e al 2020 per l’impianto “YB2”. Per l’unione e la trasformazione in .arff dei due dataset è possibile utilizzare due strade:

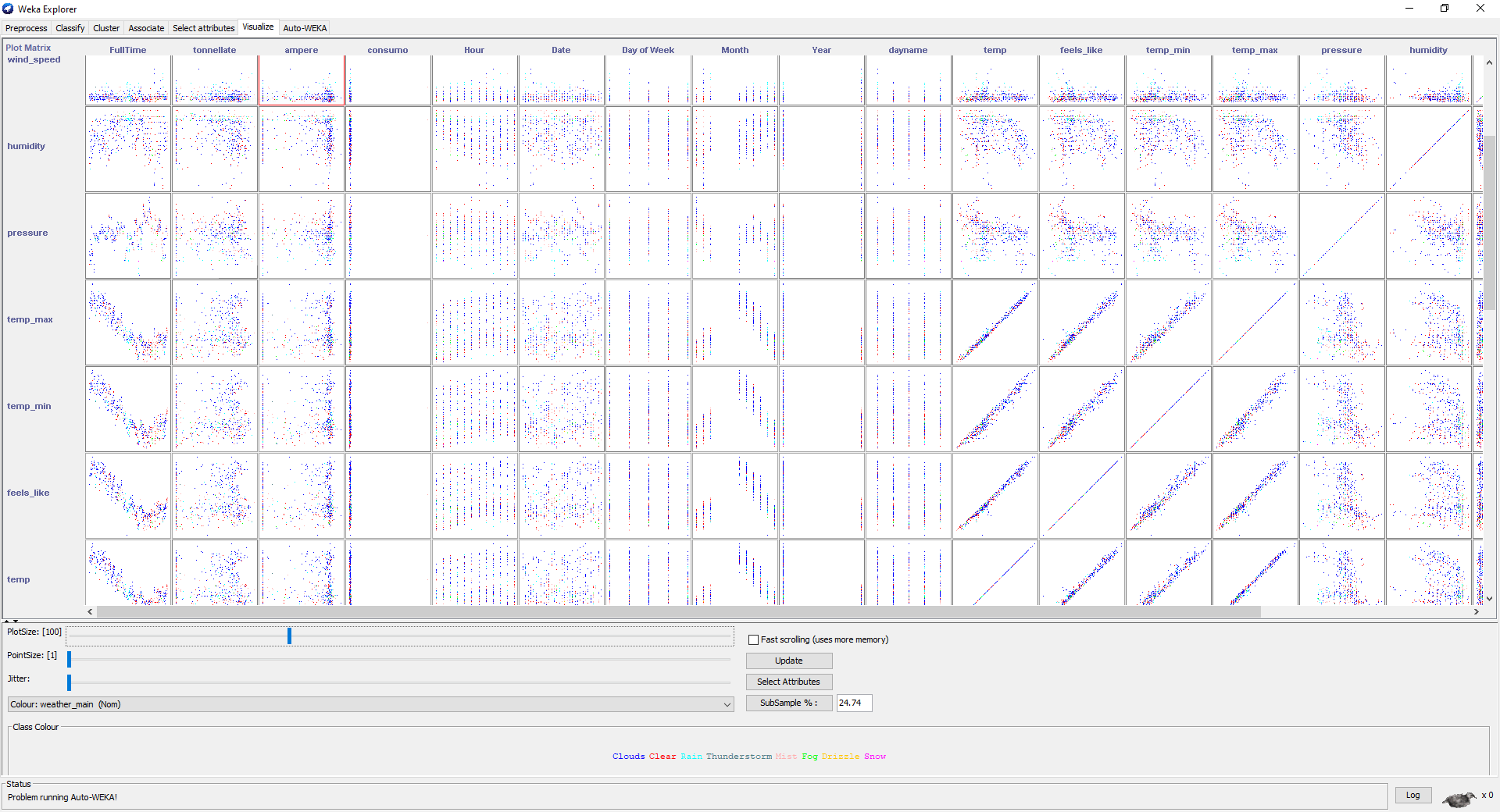
1. unire manualmente i due .csv, e poi importarli i Weka utilizzando il tool **AfterViewer** per la conversione csv → arff (nella conversione prestare particolare attenzione alla sostituzione del “;” con la “,”)
2. in maniera automatica utilizzare uno script in python che permetta la conversione csv → arff e che poi faccia il merge dei due file (in appendice si riporta un codice esemplificativo per le due operazioni)

La prima soluzione è la più semplice ma la seconda comporta la possibilità di convertire a runtime i file e di poterli processare in tempo reale senza dover intervenire manualmente.

Dopo aver caricato il dataset, Weka automaticamente mostra gli attributi disponibili dai quali è possibile selezionare le feature visualizzando statistiche e classi (Fig .4). Inoltre cliccando “visualiza all” (Fig .5) è possibile vedere le distribuzioni relative di ogni parametro rispetto agli altri.

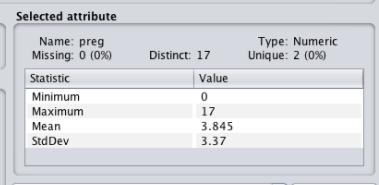
******

***Fig .4 Dataset “YB2” in weka***

******

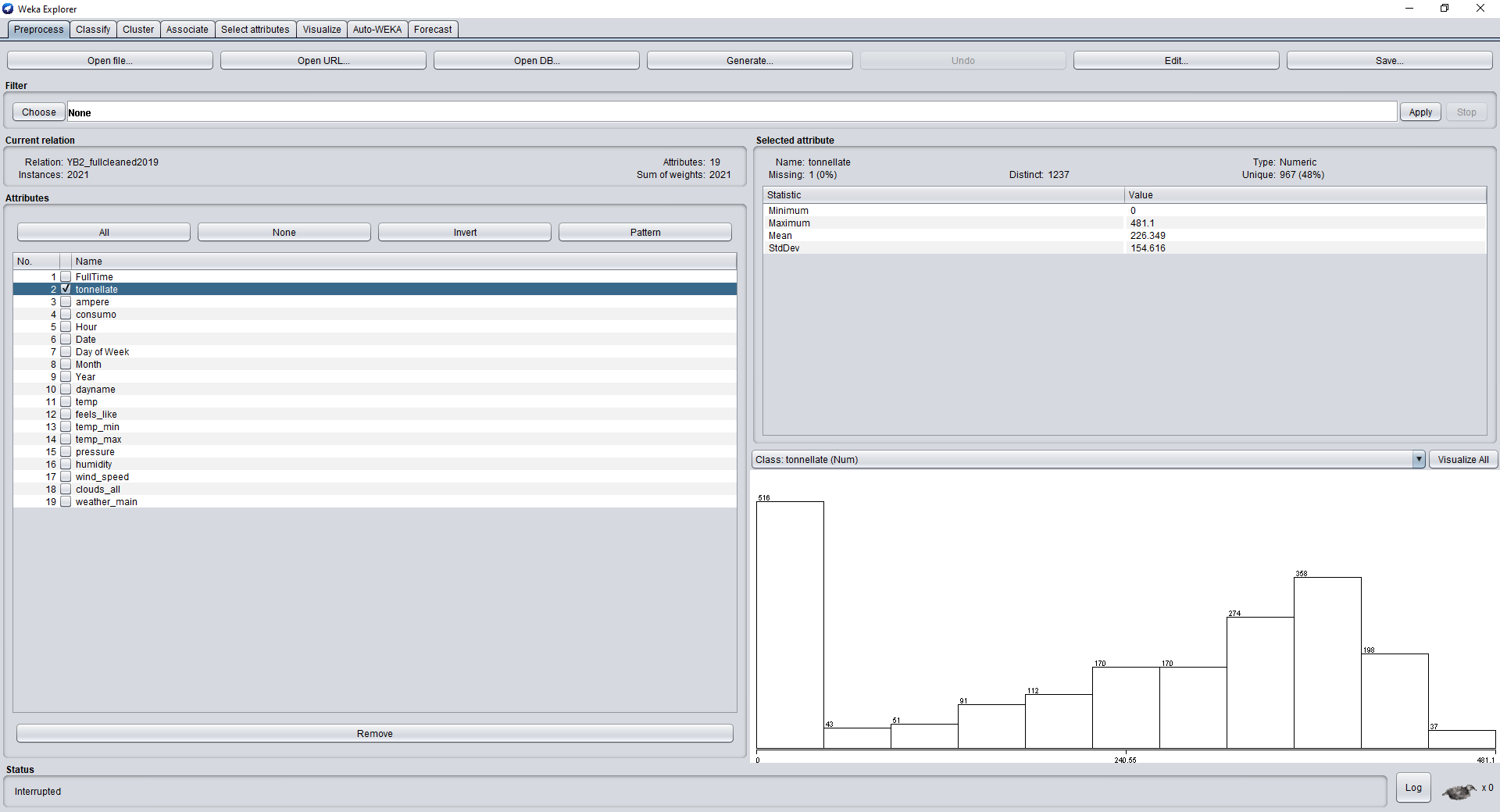
***Fig .5 Visualizzazione del confronto fra tutte le variabili del Dataset “YB2” in weka***

Inoltre selezionando ogni variabile nel box a sinistra è possibile ottenere (Fig .6)una serie di informazioni relative a quella variabile: nome, tipo, %di valori nulli, valore minimo e massimo, etc.



***Fig .6***

## Individuazione outliers e valori nulli



***Fig .7 “YB2 Dataset”***

Dall’analisi dei dati eseguita anche durante la costruzione del dataset risulta evidente una continua oscillazione dei valori, nell'intorno di un valore di riferimento e la presenza di moltissimi valori nulli che prescindono da momenti di chiusura, festività e giornate non lavorative.

Di seguito si riportano alcuni giorni in cui si sono riscontrati andamenti anomali, su queste giornate andrebbero interrogati i gestori dell'impianto al fine di capire se le anomalie siano dovute a malfunzionamenti HW o ad una gestione non esplicitata degli impianti o alle condizioni meteo.

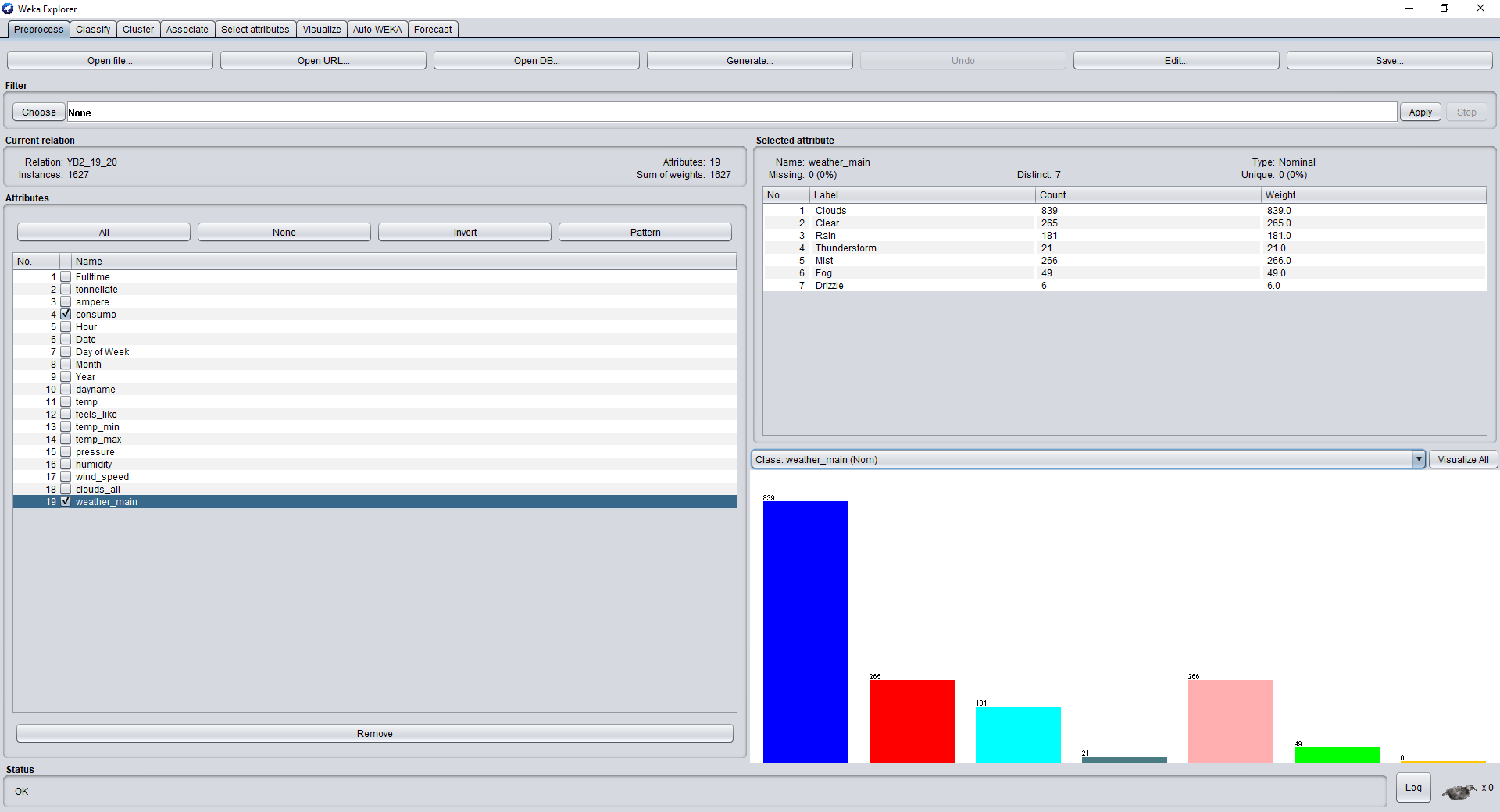
|  |  |  |
| --- | --- | --- |
| **Data/ora** | **Dati** | **Possibili Cause** |
| 16 agosto 2019 dalle 14:00:00 | tutti nulli | * interruzione attività * malfunzionamento HW * meteo passaggio da sereno a nuvoloso |
| dalle 2019-08-20 12:00:00 alle 2019-08-21 17:00:00 | tutti nulli | * interruzione attività * malfunzionamento HW * nuvoloso |
| dalle 2019-08-23 06:00:00 alle 2019-08-26 06:00:0 | tutti nulli | * interruzione attività * malfunzionamento HW * nuvoloso |
| 2019-09-05 dalle 06:00:00 fino alle 11:00:00 | tutti nulli | * interruzione attività * malfunzionamento HW * riunioni * meteo passaggio da sereno a nuvoloso |
| dal 2019-09-13 15:00:00 (venerdì) al 2019-09-16 17:00:00 (lunedì) | tutti nulli | * interruzione attività * malfunzionamento HW * meteo variabile |
| 2019-09-30 (lunedì) | tutti nulli | * interruzione attività * malfunzionamento HW * meteo variabile |
| 2019-10-02 (mercoledì) | tutti nulli | * interruzione attività * malfunzionamento HW * meteo variabile |
| Dal 2019-10-15 14:00:00 al 2019-10-24 16:00:00 | problemi di lettura degli ampere e dei consumi.  Susseguirsi di 0 alternati e un valore assolutamente fuori quota per il consumo (240.3) la situazione si stabilizza 2019-10-25 11:00:00 | * malfunzionamento * meteo variabile |
| dal 2019-10-30 06:00:00 (Mercoledì) 2019-10-30 17:00:00 | tutti nulli | * interruzione attività * malfunzionamento HW * nuvoloso |
| 2019-11-04 06:00:00 (lunedì) per mezza giornata | tutti nulli | * interruzione attività * malfunzionamento HW * riunioni * meteo variabile |
| dal 2019-11-06 16:00 (mercoledì) fino alle 7 del venerdì successivo | tutti nulli | * interruzione attività * malfunzionamento HW * meteo variabile |
| dal 2019-12-05 06:00:00 al 2019-12-05 17:00:00 (giovedì) | tutti nulli | * interruzione attività * malfunzionamento HW * sereno |
| 2019-12-13 06:00:00 tutto il giorno (venerdì) | tutti nulli | * interruzione attività * malfunzionamento HW * pioggia |
| martedì 2019-12-24 06:00:00 tutta la giornata | tutti nulli | * interruzione attività * malfunzionamento HW * sereno/nuvoloso |
| dal 2020-01-15 13:00:00 al 2020-01-16 17:00:00 | alcuni parametri nulli altri no | * malfunzionamento HW * nuvoloso |
| dal 2020-02-13 06:00:00 chiusura fino al 2020-02-17 16:00:00 | tutti nulli | * interruzione attività * malfunzionamento HW * Covid 19 * meteo variabile |
| 2020-02-20 11:00:00 al 2020-02-26 06:00:00 | tutti nulli | * interruzione attività * malfunzionamento HW * Covid 19 * meteo variabile |
| 2020-03-09 06:00:00 al 2020-03-10 17:00:00 | tutti nulli | * interruzione attività * malfunzionamento HW * Covid 19 * meteo variabile |

***Tab .2 “Problemi nei dati”***

Da questo post processing del dataset risultano evidenti alcune considerazioni sui dati:

* sembrerebbe esserci una correlazione fra la chiusura degli impianti e la condizione di nuvolosità, vanno verificate in maniera puntuale le condizioni meteo e vanno clusterizzate (come si mostrerà in seguito) al fine di correlare correttamente con gli andamenti dei consumi;
* probabilmente le condizioni covid si sono palesate precedentemente alla sospensione dichiarata delle attività;
* esistono dei malfunzionamenti di misura, probabilmente dovuti alla sensoristica utilizzata.

## Meteo labeling



***Fig .8 Labeling Meteo***

In Fig .8 è possibile vedere quali siano i colori associati alle condizioni meteo descritte nella variabile “Weather\_main” del dataset. Si riportano di seguito i valori utilizzati per la classificazione. Inoltre si noti che in fig 9 sono riportato l’andamento della misura dei consumo, ampere e tonnellate in relazione alla condizione meteo.

***blu→ cloud***

***red→ sereno***

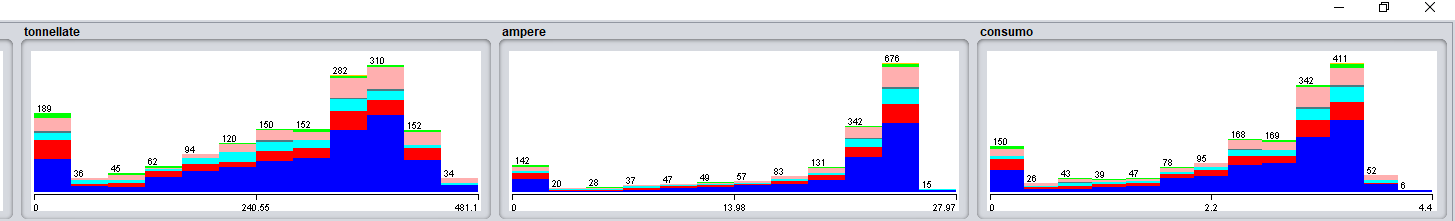
***azzurro→ pioggia***

***verde→ pioggia e vento***

***rosa→ nebbia***

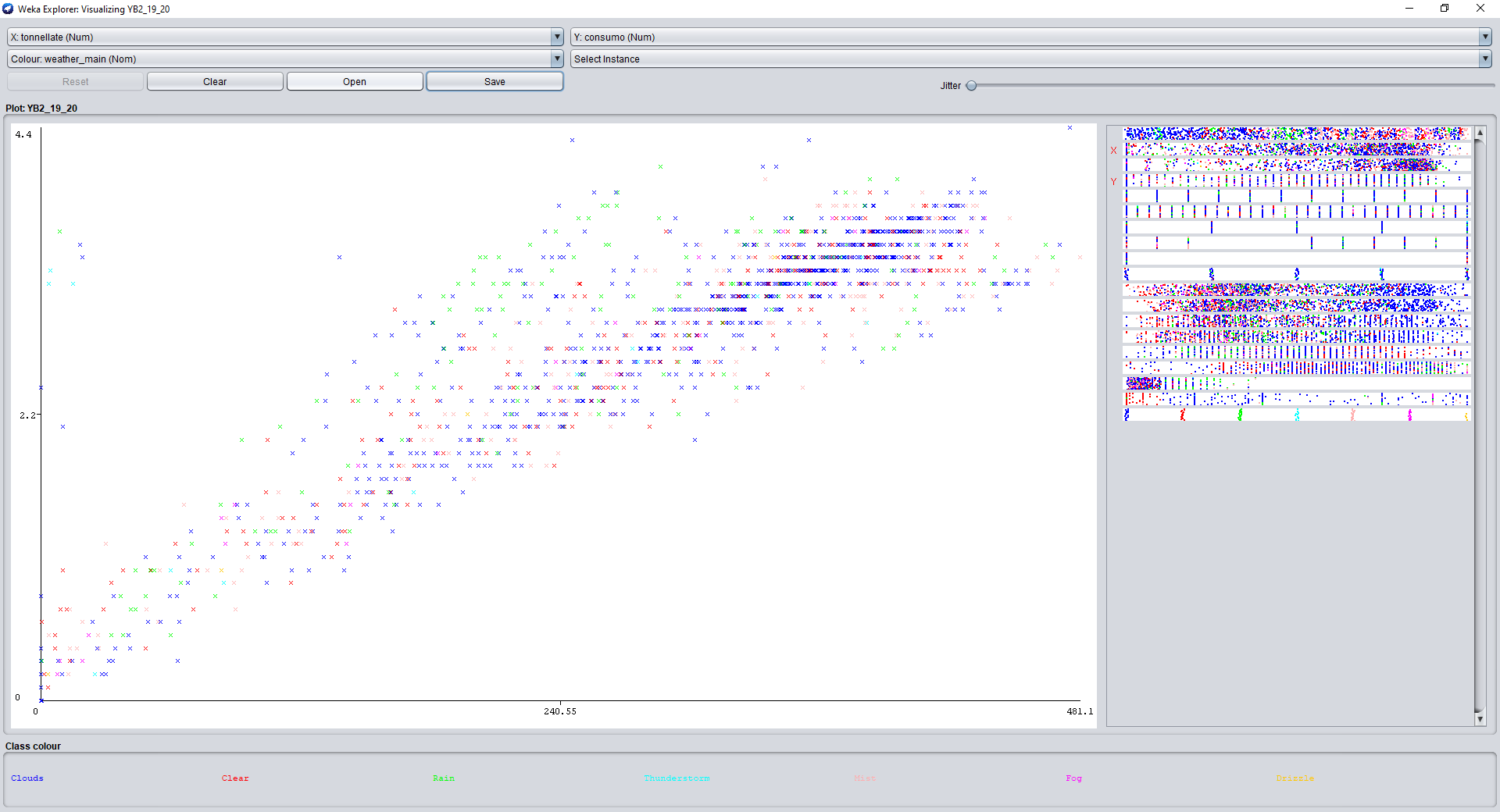
***verde fluo→ nebbia intensa***

***giallo → pioviggine***

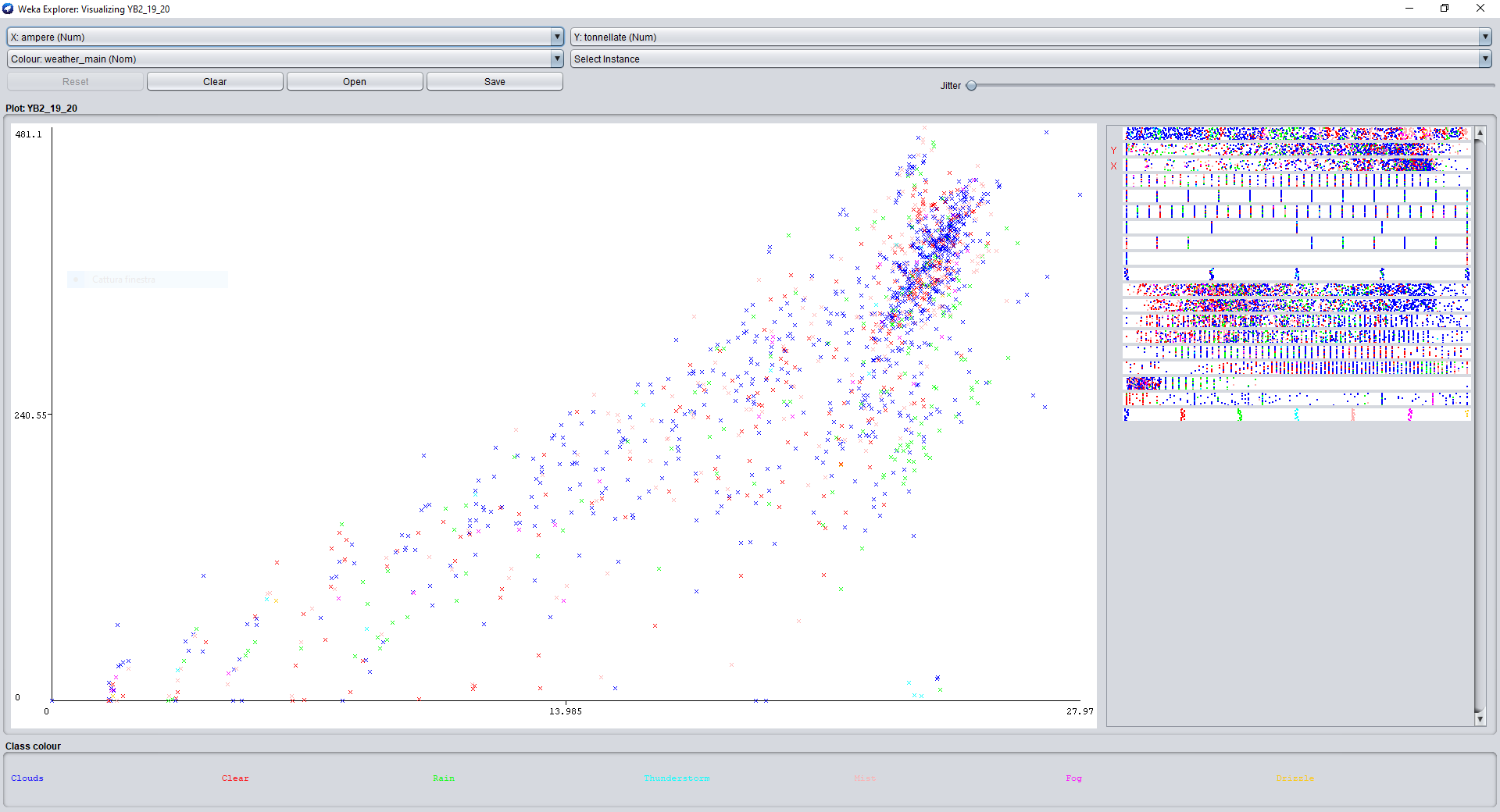


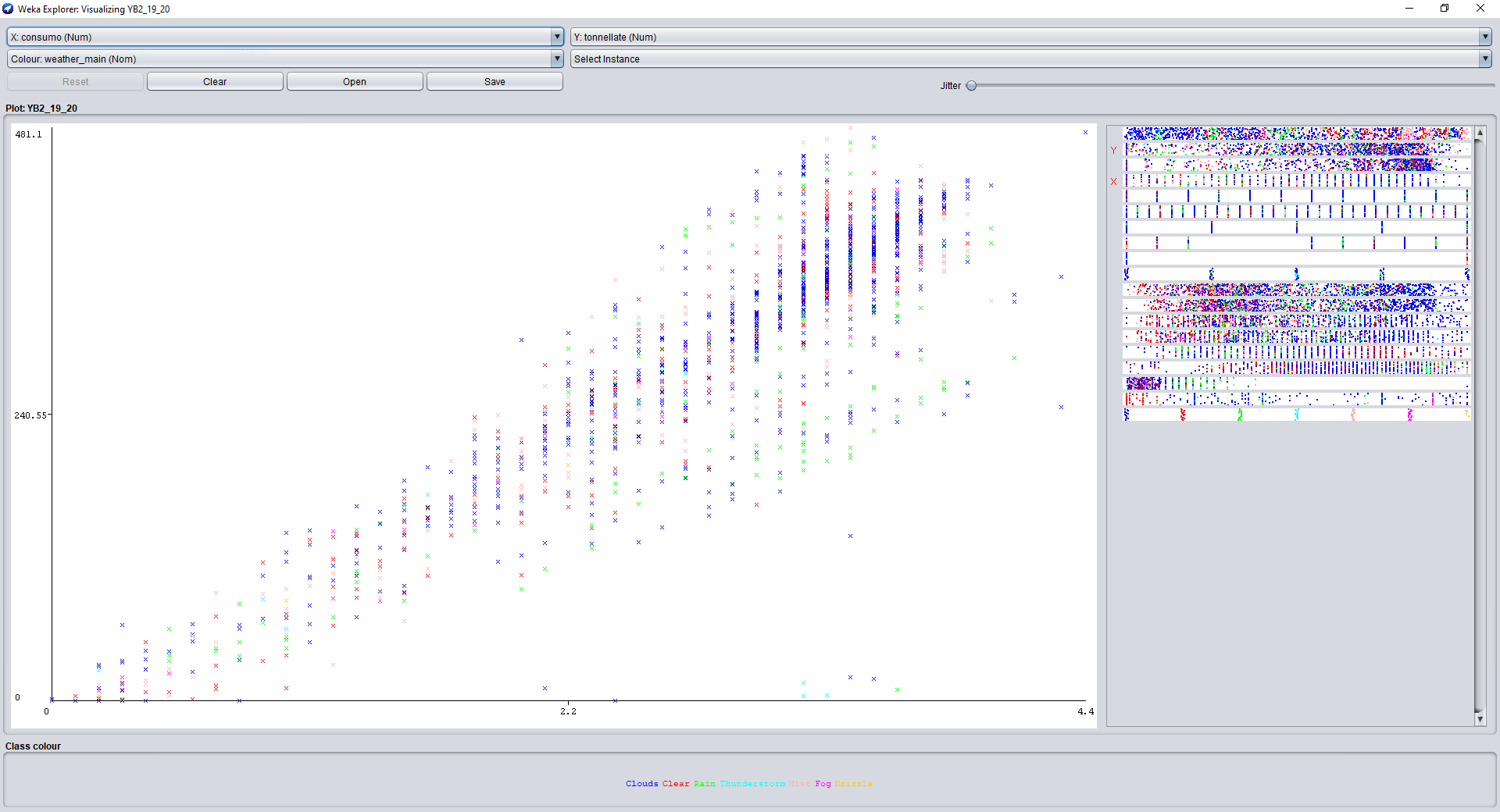
***Fig .9 Consumo Vs meteo, Ampere Vs meteo, Tonnellate Vs meteo***

Come è immaginabile le variabili consumo-ampere-tonnellate sono strettamente correlate, come mostrato in Fig 10,11 e 12.



***Fig .10 Ampere vs Consumo***

***Fig .11 Tonnellate vs Ampere***



***Fig .12 Tonnellate vs Consumo***

La variabile “Consumo” dal dataset studiato è affetta da un minor numero di outliers e contiene meno valori nulli, per questo in quello che segue verrà utilizzato come target.

Valori nulli da dataset iniziale:

**Consumo→ 3968**

**ampere→ 4207**

**tonnellate→ 4119**

# Analisi dei dati

In quello che segue si presentano i modelli utilizzati e i risultati ottenuti. I dati sono stati analizzati considerando la correlazione fra le variabili meteo e i valori di consumo prodotti dall’impianto.

Si riportano i risultati delle previsioni ottenute su orari giornalieri dalle *06:00:00 alle 17:00:00, ottenuti con uno skip per le ore notturne e con lag unitario*.

La scelta dei modelli è stata effettuata basandosi sui risultati ottenuti dall’utilizzo di un altro plugin weka, “Auto-Weka”, che permette fra le altre cose il raffronto fra modelli diversi in funzione delle feature previste.

Sono state eseguite le seguenti previsioni:

* una previsione sulla temperatura basata sull’utilizzo esclusivo di dati numerici meteorologici.
* una previsione sulla temperatura basata sull’utilizzo di dati numerici meteorologici e di label sulla condizione meteo. Questa previsione è stata attuata per verificare l’efficacia previsionale nell’inserimento di una label.
* una previsione sui consumi basata sull’utilizzo di parametri numerici meteo
* una previsione sui consumi basata sull’utilizzo di label meteo

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

SMOreg (Sequential Minimal Optimization) è un algoritmo di machine learning per SVM (Macchine a Vettori di Supporto) che implementa approssimatori e può essere utilizzato anche per realizzare previsioni su serie storiche. Weka utilizza come kernel, il PolyKernel o Kernel Polinomiale.

Le reti neurali di tipo LSTM (Long Short Term Memory) sono considerate le tipologie di reti neurali più adatte per la previsione da serie storiche ma per utilizzarle è necessario avere a disposizione una lunga serie temporale e disporre di risorse HW maggiormente complesse. Avendo un numero limitato di dati e volendo si è scelto di utilizzare un classico algoritmo supervisionato che potrebbe arrivare a livelli di accuratezza accettabili e con un costo computazionale decisamente più basso rispetto a quello di una LSTM.

Va ricordato che:

* SMOreg torna risultati anche negativi e non permette il vincolo di positività (vincolo presente nel nostro caso)
* Laddove sia possibile utilizzare una variabile labelizzata puntuale e meglio utilizzare SMO, che ben si adatta a variabile testuali e permette di inserire i vincoli di positività.

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

Normalmente questo tipo di algoritmo non viene utilizzato per l'analisi di serie temporali, ma adattando alcuni parametri è possibile ricavare informazioni preziose da questo tipo di modelli (molti esempi di utilizzo sono riportati in letteratura) soprattutto quando sia abbia a che fare con dataset con notevole variabilità e si vogliano fare previsioni a breve termine (come nel caso di Ema Control). In generale Random Forest è un miglioramento del Bagging. Utilizza un algoritmo di apprendimento ad albero che seleziona, ad ogni divisione del candidato nel processo di apprendimento, un sottoinsieme casuale delle caratteristiche, per l’analisi delle serie storiche è fondamentale la scelta dei parametri e la definizione del *ntree* e del numero di variabili campionate in ciascuna suddivisione candidata.

Con weka è possibile configurare alcuni parametri del modello, come mostrato in Fig .13:

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

In aggiunta i parametri per il bagging, come mostra la figura precedente è presente un parametro chiave che è il numero di attributi da considerare per lo split point. In Weka questo parametro è controllato da numFeatures attribute, che di default è settato 0.

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

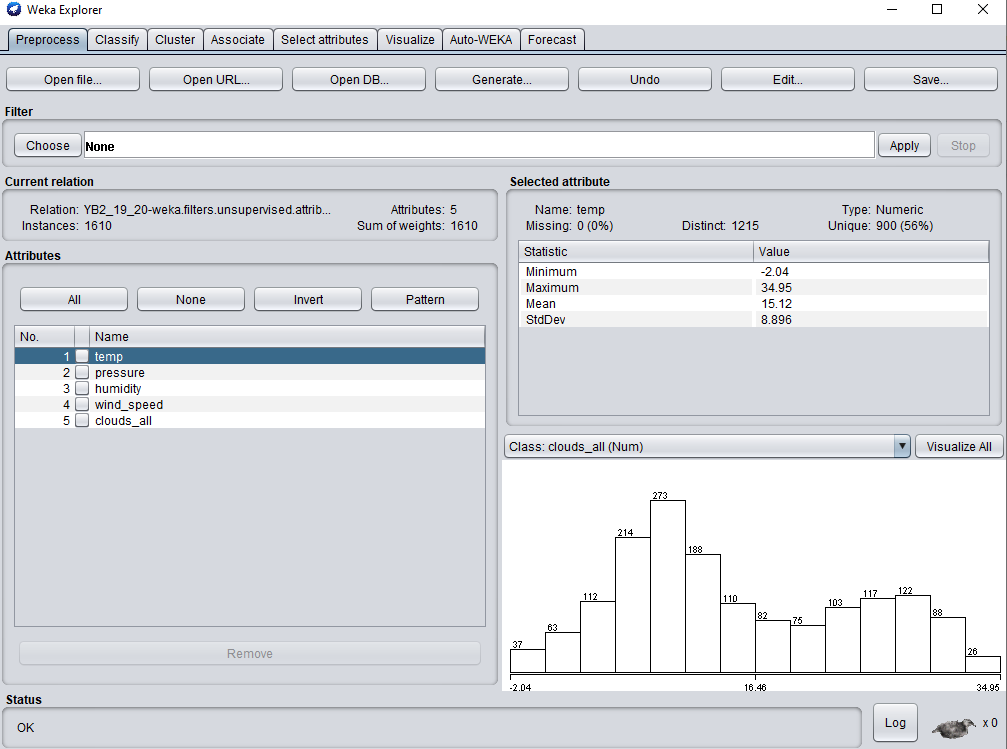
Anche in questo caso è possibile settare i parametri relativi al modello in maniera abbastanza puntuale, come mostrato In Fig .14. In particolare Weka aiuta notevolmente nella creazione di vettori “caratteristica”, basati su lag e nella definizione a priori del dataset di training e testing.

## 

***Fig .14 Configurazione MLP***

## Previsione sulla temperatura da valori numerici

Le feature utilizzate come primo approccio, sono esclusivamente quelle meteo non labellizzate (solo numeric), come mostrato in Fig.15. Di seguito si riportano i risultati ottenuti, da quanto visto nel caso in cui non si utilizzino sistemi di labeling l’algoritmo migliore risulta essere il Random Forest pur presentando valori notevoli per gli errori.



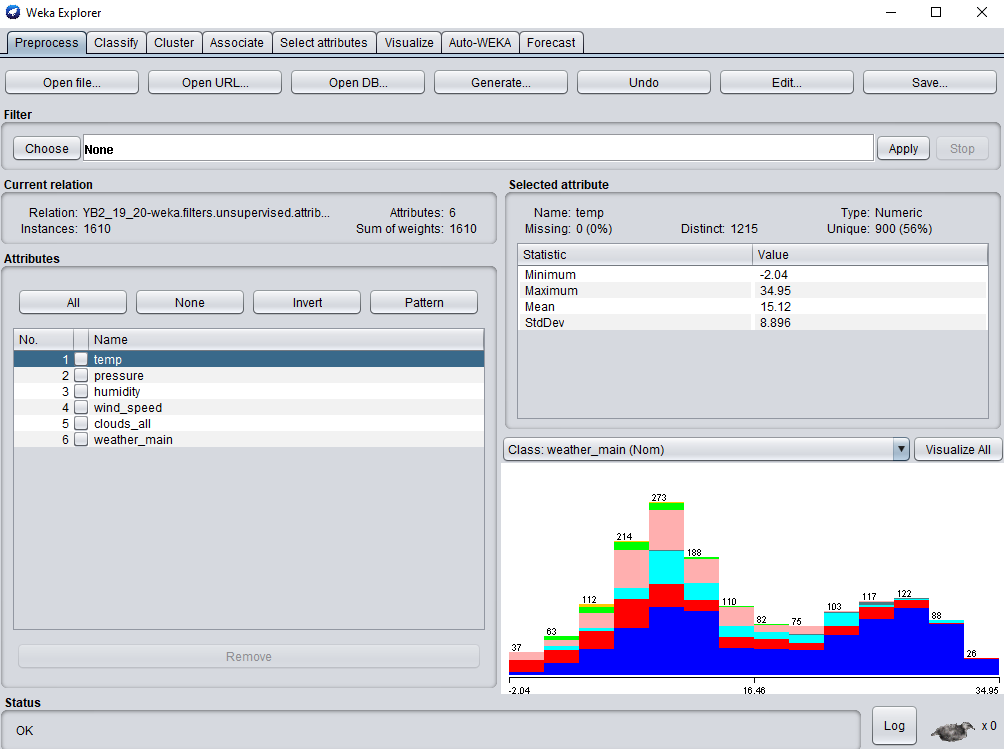
***Fig .15***

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

## 

## 

## Previsione con inserimento di labeling per le condizioni meteo



***Fig .16***

Di seguito si riportano i risultati ottenuti utilizzando i modelli precedentemente esposti. Il risultato più rilevante è quello relativo all’utilizzo dell’algoritmo SMO, da cui ci si sarebbe aspettati una classificazione migliore del Random Forest, cosa che invece come si riporta in seguito non è stata riscontrata. Una motivazione a questo andamento potrebbe essere dettata dal tipo di classificazione meteo inserita. Nello svolgimento del progetto andrà verificata la nomenclatura utilizzata e le condizioni in cui viene inserita. In particolare si è notato che diminuendo e definendo in maniera maggiormente univoca le variabili meteo introdotte si ottengono risultati migliori.

#### RandomForest

Nello specifico l’albero decisionale adottato le prestazioni sono state calcolate utilizzando la valutazione metriche che includono parametri come precisione, accuratezza, frequenza FP rate, TP rate, F-measure e area ROC.

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 ? 0,792 ? ? 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 | 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 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 | a = Clouds

1 91 0 0 1 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 | e = Mist

9 0 0 0 11 0 0 | f = Fog

1 0 0 0 3 0 0 | g = Drizzle

#### Multilayer perceptron 6 nodi

=== 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 0,091 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 | a = Clouds

0 89 0 0 4 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 0 3 0 | g = Drizzle

## 

***Fig.17 Andamento della Temperatura ultime 6h previsione***

## Previsione consumi Vs features meteo numeriche

Come in precedenza si riportano i risultati dei modelli ottenuti utilizzando le variabili meteo numeriche e come target il consumo.

#### 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

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

consumo

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 ===

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

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

consumo

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 1.1332

#### 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

consumo

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 ===

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

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

consumo

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 2.798

Total number of instances: 1608

## Consumo vs labeling meteo

La tabella seguente mostra la sintesi dei risultati ottenuti utilizzando i 3 algoritmi con 2 tipi

di metriche di valutazione e con cinque (6) step-head, il dettaglio delle operazioni è riportato in seguito

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Algoritmo Usato | 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 ===

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

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

consumo

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

### 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 ===

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

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

consumo

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 1.1332

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 ===

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

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

consumo

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 2.798

Total number of instances: 1608

L’analisi effettuata finora mostra come l’algoritmo che offre migliori performance è il random forest applicato a consumi e dati meteo labellizzati (lo stesso risultato si ottiene utilizzando i dati meteo numerici e il consumo), al contempo l’analisi precedente fatta su dati meteo numerici e labeling mostra come la labelizzazione non sia sempre ben correlata con il dato meteo. E’ quindi necessario determinare un metodo di labeling migliore.

Il risultato ottenuto era prevedibile, questo tipo di algoritmo infatti risulta molto efficiente nelle previsioni a breve termine e con alta variabilità come nel caso della piattaforma emacontrol (previsioni orarie).

Il random Forest è un metodo di apprendimento di ensemble che genera molti alberi di regressione (CART) e aggrega i loro risultati. Il modello si applica bene su cicli stagionali delle serie temporali e semplifica il problema di previsione soprattutto quando una serie temporale mostra non stazionarietà, eteroschedasticità, tendenza e cicli stagionali multipli. I principali vantaggi del modello sono la sua capacità di generalizzazione, convalida incrociata incorporata e bassa sensibilità ai valori dei parametri.

Sarebbe opportuno avendo a disposizioni più dati, almeno due anni, testare algoritmi stagionali quali Arima o comunque tentare di testare gli algoritmi sopracitati dividendo il dataset in stagioni.

Inoltre, con la presenza di più dati si potrebbe ragionare su una etichettatura giornaliera per la condizione meteo e per i dati di produzione dell’impianto e riprovare la previsione con modelli tipo SMO che ben si adatta al risultato che si vorrebbe ottenere, ma che in mancanza di albel risulta non utilizzabile.

# Appendice

#CSV > arff

import csv

from time import sleep

class convert(object):

content = []

name = ''

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

self.csvInput()

self.arffOutput()

print '\nFinished.'

#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

del class\_items[0]

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

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

#write data

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

del self.content[0]

for row in self.content:

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

#close file

new\_file.close()

sleep(2)

#####

run = convert()

#Unione di due 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"

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

self.files.append({

'data': data,

'meta': meta

})

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

self.files.append({

'data': data,

'meta': meta

})

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 attribute 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 data

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)