(FP7 614100)

D5.3 Data Mining and Machine Learning Tools

DATE – Version 1.0

Published by the IMPReSS Consortium

Dissemination Level: Public

**Project co-funded by the European Commission within the 7th Framework Programme and the Conselho Nacional de Desenvolvimento Científico e Tecnológico
Objective ICT-2013.10.2 EU-Brazil research and development Cooperation**

Document control page

Document file:	D5.3 Data Mining and Machine Learning Tools_v1.0.docx
Document version:	1.0
Document owner:	Eduardo Souto (UFAM)
Work package:	WP5 - Data Storage, Analysis & Decision Support
Task:	T5.3 Tool Support for Data Learning
Deliverable type:	P
Document status:	☒ approved by the document owner for internal review ☒ approved for submission to the EC

Document history:

Version	Author(s)	Date	Summary of changes made
0.1	Thiago Rocha	2015-02-02	First Draft
0.2	Wesllen Souza	2015-02-10	Inserting database description to show how the web application will be used
0.3	Thiago Rocha	2015-02-10	Inserting the description over the machine learning algorithms provided by IMPRESS analytics module
0.4	Wesllen Souza	2015-02-16	Ch. 5 (Web application tool) added and refined.
0.4	Eulanda Santos	2015-02-16	Revision after initial feedback from Eduardo (UFAM).
0.5	Eduardo Souto	2015-02-17	Ready for internal review
1.0	Eduardo Souto	2015-02-27	Internal review comments incorporated. Final version ready for submission.

Internal review history:

Reviewed by	Date	Summary of comments
Carlos Kamienski	2015-02- 23	Accepted with minor corrections and comments

Index:

1. Executive summary

This deliverable describes a web application used to provide machine learning techniques to help developers to identify environmental tendencies. The purpose of this web application is to help developers that do not have any knowledge in machine learning algorithms. With the use of the web application the developers will be able to compare different machine learning algorithms that are available at the IMPReSS development platform and choose the best one to fit their needs.

The rest of the deliverable is divided as follows. First, the machine learning algorithms used at the web application are described. Then, the databases that were chosen to be used will be explained. After the format that the database data needs to have is explained and finally, a short tutorial is presented in order to show how the application can be used. This is accomplished by providing examples which illustrate tasks employing algorithms and databases available in the application.

2. Introduction

2.1 Purpose, context and scope of this deliverable

The IMPRESS development platform consists of a set of technologies that help to build general-purpose applications accessing a plethora of information sources, such as information from the physical world, analyzing and fusing relevant data, and performing monitoring and control operations on complex systems. This is achieved through the definition of a number of tools and pre-defined modules that can be managed and combined in order to define a specific logic flow. One of these modules is about machine learning techniques.

The purpose of this deliverable is to present a web application developed to provide machine learning algorithms. In order to help developers to better understand the available algorithms, as well as to allow developers to choose an algorithm that best fits the monitored scenarios, this deliverable also describes the algorithms provided and presents practical examples.

2.2 Background

IMPReSS is an EU-Brazil cooperation project aiming at providing a Systems Development Platform (SDP), which enables rapid and cost effective development of mixed criticality complex systems involving Internet of Things and Services (IoTS) and at the same time facilitates the interplay with users and external systems. The IMPReSS development platform will be usable for any system intended to embrace a smarter society. The demonstration and evaluation of the IMPReSS platform will focus on energy efficiency systems addressing the reduction of energy usage and CO2 footprint in public buildings, enhancing the intelligence of monitoring and control systems as well as stimulating user energy awareness.

The IMPReSS project aims at solving the complexity of system development platform (SDP) by providing a holistic approach that includes an Integrated Development Environment (IDE), middleware components, and a deployment tool.

3. Web Application Algorithms

Machine Learning algorithms are used to solve tasks for which the design of software using traditional programming techniques is difficult. Machine failures prediction, filter for electronic mail messages and user behaviour identification are examples of these tasks. Several different machine learning algorithms have been proposed in the literature. These algorithms may be divided into three categories: regression, classification and clustering algorithms. This categorization takes into account whether or not one of the following aspects is considered: use of labelled training examples, and real or discrete outputs.

In clustering, samples in the training set are not labelled or classified. The objective is to form clusters or natural groupings of the input samples. Cluster analysis can be used to provide insight in the distribution of data, as a pre-processing level for other algorithms, etc. According to the literature, different clustering algorithms lead to different results.

On the other hand, labelled training samples are available in classification and regression problems. The objective of these algorithms is to find the best functional relationship between input and output, called target or decision function. In regression problems, the outputs are continuous values while the outputs are discrete values in classification problems. Again, several regression and classification algorithms are available in the literature. These methods may achieve different performances when evaluated in different problems.

Based on this context, the algorithms provided in the web application are divided into regression, clustering and classification algorithms. It is important to mention that the algorithms were developed using scikit-learn [1] - an open source and commercially usable library based on Python. Another important characteristic is the fact that scikit-learn is broadly used by companies like Evernote, Spotify and DataRobot. Moreover, this library has a large open source community support and documentation. In the next section, each algorithm used at the application will be explained.

3.1 Regression Algorithms

Regression algorithms are tools employed to predict a variable which takes continuous values as the output of a problem. Profit, sales, mortgage rates, house values, square footage, temperature or distance are examples of outputs which can be predicted using regression techniques [2]. For instance, a regression model can be used to predict the energy cost of a house taking into account number of rooms, number of equipment and other features. Among the various regression methods proposed in the literature, three algorithms are available in the web application, as described below.

3.1.1 Linear Regression

Linear regression is the most popular and well-studied form of regression. This algorithm focuses on finding the best-fitting straight line through points. The best-fitting line is called a regression line [3]. Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. This is due to the fact that models linearly related to their unknown parameters are easier to fit than models non-linearly related to their parameters. Moreover, the statistical properties of the resulting estimators are easier to determine [4]. Linear regression has many practical uses. Most applications fall into one of the following two broad categories:

- When the goal is prediction, forecasting, or reduction, linear regression can be used to generate a predictive model based on an observed data set composed of X values and its y labels. After designing such a model, if an unknown value X (with no label y) has to be predicted, the generated model can be used to assign a y value for X.
- Given a variable y and a number of variables X_1, ..., X_p that may be related to y, linear regression analysis can be applied to quantify the strength of the relationship between y and the X_j, to assess which X_j may have no relationship with y at all, and to identify which subsets of the X_j contain redundant information about y.

The linear regression model fits a linear function to a set of data points [5]. The form of the function is:

$$Y = \beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + \ldots + \beta_n * X_n$$

where Y is the target variable and X_1, X_2, ... X_n are the predictor variables and β_1, β_2, … β_n are the coefficients that multiply the predictor variables. β_0 is constant. For example, given a CEO of a shoes company franchise who is considering different cities for opening a new store, in addition, the chain already has stores in various cities and the CEO has data for profits and populations from the cities. In this scenario, the CEO can use this data to help at the selection of which city to expand next, by using linear regression to predict profits for the new store.

3.1.2 Logistic Regression

Logistic Regression is a regression algorithm that predicts the probability of occurrence of an event by fitting data to a logit function (logistic function). Similar to many forms of regression analysis, Logistic Regression relies on using several predictor variables, which may be either numerical or categorical. For instance, the probability that a person may have a heart attack within a specified time period might be predicted based on features such as the person's age, sex and body mass index. This regression algorithm is frequently used in different scenarios like prediction of customer's propensity to purchase a product or cease a subscription in marketing applications and many others [6].

Given *p (x)* the probability of occurrence of event *x,* formally, the logistic regression model is:

$$\log = \frac{p(x)}{1 - p(x)}$$

On the one hand, it is widely accepted that Logistic Regression has several advantages over linear regression [6]. For instance, Logistic Regression is more robust than linear regression since it does not assume linear relationship. As a consequence, it may handle nonlinear effects. On the other, it requires much more data to achieve stable and meaningful results.

3.1.3 SVM Regression

Support Vector Machines (SVM) is a supervised learning algorithm employed to analyse data and to recognize patterns, which is used in classification and regression problems. SVM is based on the notion of a hyperplane that separates two data classes by maximizing the margin between them. Therefore, SVM focuses on creating the largest possible distance between the separating hyperplane and the instances on either side of it. The literature has shown that SVM may reduce both the training and the expected generalisation error.

SVR, SVM version for regression, maintains all the main features that characterise the maximal margin algorithm: a non-linear function is learned by a linear learning machine in a kernel-induced feature space, while the capacity of the system is controlled by a parameter that does not depend on the dimensionality of the space. In SVR, the basic idea is to map the data x into a high-dimensional feature space F via a nonlinear mapping T and to employ linear regression in space F [7].

In SVR, the objective is to estimate the functional dependence of a dependent variable *y* on a set of independent variables *x*. It is assumed, as in other regression problems, that the relationship between the independent and dependent variables is given by a deterministic function *f* plus the addition of some noise, as shown below:

$$\mathrm{y} = \mathrm{f(x)} + noise$$

Thus, the task involves finding a functional form for *f* that will lead to a correctly prediction of new cases. The SVR training process is accomplished through the sequential optimization of an error function. Depending on the definition of this error function, different types of SVM models can be recognized.

In the web application, both SVR and SVM for classification, known as SVC, are provided.

3.2 Clustering Algorithms

Clustering techniques try to identify natural clusters of input data according to a similarity measure, for instance Euclidian distance. The members of the same cluster are more similar to each other than they are in comparison with members of other clusters. The goal of clustering analysis is to find high-quality clusters such that the inter-cluster similarity is low and the intra-cluster similarity is high [2]. Among several clustering algorithms proposed in the literature, two algorithms are available in the web service. These algorithms are described in this section.

3.2.1 K-Means

K-Means is one of the simplest unsupervised learning algorithms applied to solve clustering problems. The procedure follows a simple and easy way to group a given data set into *k* clusters, where *k* is fixed a priori. First, k centroids are chosen, one for each cluster. These centroids should be placed in a cunning way because different location causes different results. So, the best choice is to place them as much as possible far away from each other [8].

In the second step, each sample belonging to a given data set is associated to the nearest centroid. The nearest centroid of each sample is pointed out by means of similarity measures such as Euclidian distance. When no sample is pending, each group is then defined. After, k new centroids are re-calculated as barycentre's of the clusters generated in the previous step. After the identification of these k new centroids, a new comparison is done between the same data set samples and the nearest new centroid. Thus, a loop is generated. The result of this loop may lead the k centroids change their location step by step until no more changes are detected. Then, the algorithm attains its convergence.

The web application described in this deliverable also provides an alternativeversion of K-Means, called Mini Batch K-Means. This version uses mini-batches to reduce the computation time, while still attempting to optimise the same objective function. Mini-batches are subsets of the input data, randomly sampled in each training iteration. These mini-batches drastically reduce the amount of computation required to converge to a local solution. In contrast to other algorithms that reduce the convergence time of k-means, mini-batch k-means produces results that are generally only slightly worse than the standard algorithm [9].

3.2.2 MeanShift

Another algorithm available in the web application is MeanShift. The MeanShift algorithm is a nonparametric clustering technique which does not require prior knowledge of the number of clusters, and does not constrain the shape of the clusters. This algorithm considers feature space as an empirical probability density function.

Given a set of samples, MeanShift considers them as sampled from the underlying probability density function. If clusters are presented in the feature space, then they will correspond to the mode of the probability density function. For each sample, MeanShift associates it with the nearby peak of a dataset probability density function. Thus, for a set of samples, MeanShift defines a window around it and computes the mean of each set. Then, it shifts the centre of the window to the mean and repeats the algorithm until it converges. After each iteration, it is considered that the window shifts to a denser region of the dataset.

At the high level, we can specify MeanShift as follows:

- Fix a window around each sample.
- Compute the mean of data within the window.
- Shift the window to the mean and repeat till convergence.

3.3 Classification Algorithms

Handwritten digits recognition, business modeling, and credit analysis are examples of classification problems. As a type of supervised learning, a classification algorithm builds a model that is used to assign labels to the unknown examples. The data used to generate the models may include energy consumption information such as cost and time to allow the prediction of classes of behaviors. The input or training data for a supervised learning algorithm requires the presence of attributes to represent each training sample, as well the classes assigned to the training samples [11].
Three classification methods are provided in the web service: Decision Tree and Naïve Bayes, which are single-based classifier algorithms; and random forest – an ensemble-based classification method.

3.3.2 Decision Tree

Decision tree builds classification models in the form of a tree structure. It breaks down a dataset into smaller subsets while at the same time an associated decision tree is incrementally developed. Therefore, Decision tree uses a flowchart structure to evolve a set of data inside of some pre-defined classes, providing a description, categorization and generalization of a set of data. The final result is a tree with decision nodes and leaf nodes: decision nodes have two or more branches, while leaf nodes represent a classification or decision. The topmost decision node in a tree is called root node, which corresponds to the most relevant feature among all features used to describe each sample of the training dataset. Decision trees can handle both categorical and numerical features [14].
Decision Tree-based algorithms have the following main features [15].

- Are simple to understand and interpret. People are able to understand decision tree models after a brief explanation.
- Provide important information even with little hard data. Important insights can be generated based on experts describing a situation and their preferences for outcomes.
- Allow the addition of new possible scenarios.
- Help determine worst, best and expected values for different scenarios.
- Can be combined with other decision techniques.

3.3.3 Naive Bayes

Naive Bayes classifier is a probabilistic method based on applying Bayes theorem with naive independence assumptions. The independence assumption leads a naive Bayes classifier to assume that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. For instance, a user may be considered to be a highly energy consumer if he/she has air conditioner always working, several guests, and about 10 rooms in his/her home. Even if these features depend on each other or upon the existence of the other features, a naive Bayes classifier considers all of these properties to independently contribute to the probability that this person is highly energy consumer [16].

Naive Bayes classifiers can be trained in a supervised learning setting. In many practical applications, parameter estimation for Naive Bayes models uses the method of maximum likelihood. In addition, an advantage of the Naive Bayes classifier is that it requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix. The web application has the Gaussian Naive Bayes type developed, this kind of naïve Bayes classifier may deal with continuous data.

3.3.1 Random Forest

Classifier ensembles attempt to overcome the complex task of designing a robust, well suited individual classifier by combining the decisions of relatively simpler classifiers. It has been shown that significant performance improvements can be obtained by creating classifier ensembles and combining their classifier members' outputs instead of using single classifiers. The construction of classifier ensembles may be performed by adopting different strategies. One possibility is varying the data, for instance using different data sources, different pre-processing methods, different sampling methods, distortion, etc. It is important to mention that the generation of an ensemble of classifiers involves the design of the classifiers members and the choice of the fusion function to combine their decisions.

Even though several ensemble methods are reported in the literature, only one of the most popular ensemble construction methods is provided in the web service: Random Forest. This classifier ensemble consists of a collection or simple tree predictors, each capable of producing a response when presented with a set of predictor values. For classification problems, this response takes the form of a class membership which associates or classifies a set of independent predictor values with one of the categories present in the dependent variable [12].

Random Forest is obtained by manipulating the original set of features available for training. The objective is to provide a partial view of the training dataset to each ensemble member, leading them to be different from each other. In addition, this ensemble method is also based on varying the training samples in order to generate different datasets for training the ensemble members. In this way, the classifier members will be accurate and different from each other to ensure performance improvement.

In terms of combination function, given a set of simple trees and a set of random predictor variables, the Random Forest method defines a margin function that measures the extent to which the average number of votes for the correct class exceeds the average vote for any other class present in the dependent variable. This measure provides not only a convenient way of making predictions, but also a way of associating a confidence measure with those predictions. Random Forest has some features [13]:

- It is unexcelled in accuracy among current algorithms.
- It runs efficiently on large data bases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
- It has methods for balancing error in class population unbalanced data sets.
- Prototypes are computed that give information about the relation between the variables and the classification.
- It offers an experimental method for detecting variable interactions.

Besides learning algorithms, the web service also provides some databases in order to help developers to better understand how the learning algorithms work. The databases are described in the next section.

4. Database

Two databases were chosen to show how the web application will be used. These databases were downloaded from [17], a very well-known site with machine learning databases. The goals of each database will be explained below. Then, a how-to is presented to show how the data from these databases have to be formatted to be used in the application.

4.1 Energy Efficiency Data Set

This dataset was created by Angeliki Xifara and processed by Athanasios Tsanas. The goal is to evaluate the heating cooling load requirements of buildings (that is, energy efficiency) as a function of building parameters.

The energy analysis was performed using 12 different building shapes. They differ with respect to the glazing area, the glazing area distribution, and the orientation, amongst other parameters. Various settings were simulated as functions of the afore-mentioned characteristics to obtain 768 building shapes.

As mentioned above, the dataset comprises 768 samples and 8 features (denoted by X1...X8), aiming to predict two real valued responses (denoted by y1 and y2). Therefore, it is a dataset for regression algorithms, which can be converted to classification problems.

Originally, the objective is to use the eight features to predict each of the two responses. The features are listed below:

- X1 Relative Compactness
- X2 Surface Area
- X3 Wall Area
- X4 Roof Area
- X5 Overall Height
- X6 Orientation
- X7 Glazing Area
- X8 Glazing Area Distribution
- y1 Heating Load
- y2 Cooling Load

4.2 Individual Household Electric Power Consumption Data Set

This dataset was created by Georges Hebrail, to measure the electric power consumption in one household with a one-minute sampling rate over a period of almost 4 years. Again, it is a regression problem.

Different electrical quantities and some sub-metering values are available. The dataset contains 2.075.259 measurements gathered between December 2006 and November 2010 (47 months). For the web service, we got information only from 2007. The dataset contains nine attributes, as listed below:

- date: Date in format dd/mm/yyyy
- time: time in format hh:mm:ss
- global_active_power: household global minute-averaged active power (in kilowatt)
- global_reactive_power: household global minute-averaged reactive power (in kilowatt)
- voltage: minute-averaged voltage (in volt)

- global_intensity: household global minute-averaged current intensity (in ampere)
- sub_metering_1: energy sub-metering No. 1 (in watt-hour of active energy). It corresponds to the kitchen, containing mainly a dishwasher, an oven and a microwave (hot plates are not electric but gas powered).
- sub_metering_2: energy sub-metering No. 2 (in watt-hour of active energy). It corresponds to the laundry room, containing a washing-machine, a tumble-drier, a refrigerator and a light.
- sub_metering_3: energy sub-metering No. 3 (in watt-hour of active energy). It corresponds to an electric water-heater and an air-conditioner.

4.3 Database Format

To be used in the web application, the databases have to follow a JSON pattern. Classification and regression algorithms follow the same pattern. On the other hand, clustering and the metrics of each algorithm have different forms. Frame 1 shows the JSON format for classification and regression input.

$$
\begin{array}{ll}
\{ & \\
\quad \text{"data":} & [[d_{11}, d_{12}, d_{13}, \dots, d_{1m}],\\
 & \;[d_{21}, d_{22}, d_{23}, \dots, d_{2m}],\\
 & \;[\;\vdots\;,\;\vdots,\;\vdots\;,\;\vdots\;,\;\vdots\;],\\
 & \;[d_{n1}, d_{n2}, d_{n3}, \dots, d_{nm}]],\\
\quad \text{"target":} & [t_1, t_2, t_3, \dots, t_n],\\
\quad \text{"predict":} & [[p_{11}, p_{12}, p_{13}, \dots, p_{1m}],\\
 & \;[p_{21}, p_{22}, p_{23}, \dots, p_{2m}],\\
 & \;[\;\vdots\;,\;\vdots,\;\vdots\;,\;\vdots\;,\;\vdots\;],\\
 & \;[p_{n1}, p_{n2}, p_{n3}, \dots, p_{nm}]]\\
\} &
\end{array}
$$

Frame 1 JSON Input format for classification and regression algorithms.

As can be observed in Frame 1, JSON format contains three keys explained below:

- **Data:** This key represents the data that will be learned to generate the model. The data should be a matrix (d_{nm}), where *n* indicates the number of samples and *m* indicates the number of features. The features should be integers or float numbers.
- **Target:** This key represents the target attributes corresponding to each line of the data matrix. The target attributes are used by the algorithm to map input to output in order to predict new data contained in the key predict. The array should contain integers or float numbers. The array length should be exactly the matrix line numbers, i.e. *n*.
- **Predict:** This key has the unknown samples that will be used to make predictions. These data can be represented in an array or a matrix that contains either integers or float numbers.

Frame 2 shows the results returned from applying the classification algorithms, with two keys:

- **Predicted:** This key returns an array of data with integer or floats values and shows the prediction target assigned to each sample. The samples are collected from the predict key in Frame 1.

- **Score:** This key represents the accuracy on the given test data and returns a float value.

{
"Predicted": $[p_{11}, p_{12}, p_{13}, \dots, p_{1m}]$,
"Score": $[s_1, s_2, s_3, \dots, s_n]$
}

Frame 2. Result from classification algorithms.

Frame 3 shows the result returned from regression algorithms. The result has three keys:

- **Decision Function:** This key returns an array of data with integer or float values and represents the decision function from the model used to predict confidence scores for samples.
- **Predicted:** This key returns an array of data with integer or float values and shows the prediction of each sample. The samples are collected from the predict key in Frame 1 and the prediction is based on the data contained in the key decision function.
- **Score:** This key represents the accuracy on the given test data and returns a float value.

{
"Decision Function": $[d_{11}, d_{12}, d_{13}, \dots, d_{1m}]$,
"Predicted": $[p_{11}, p_{12}, p_{13}, \dots, p_{1m}]$,
"Score": $[s_1, s_2, s_3, \dots, s_n]$
}

Frame 3. Result from regression algorithms.

{
"data": $[[d_{11}, d_{12}, d_{13}, \dots, d_{1m}],$
$[d_{21}, d_{22}, d_{23}, \dots, d_{2m}],$
$[\ \vdots\ ,\ \vdots,\ \ \vdots\ ,\ \vdots\ ,\ \vdots\ \ ],$
$[d_{n1}, d_{n2}, d_{n3}, \dots, d_{nm}]],$
"predict": $[[d_{11}, d_{12}, d_{13}, \dots, d_{1m}],$
$[d_{21}, d_{22}, d_{23}, \dots, d_{2m}],$
$[\ \vdots\ ,\ \vdots,\ \ \vdots\ ,\ \vdots\ ,\ \vdots\ \ ],$
$[d_{n1}, d_{n2}, d_{n3}, \dots, d_{nm}]]$
}

Frame 4. JSON Input format for clustering algorithms.

Frame 4 shows the json format that covers the clustering algorithms (Kmeans, MeanShift, MiniBatchKMeans). The format contains two keys:

- **Data:** This key represents the data that will be trained to generate the model. The data should be a matrix (d_{nm}), again *n* indicates the number of samples and *m* indicates the number of features. The features should be integers or float numbers.
- **Predict:** This key has the samples that will be used to make predictions (test set) to discover the clusters. These data can be represented in an array or a matrix that contains either integers or float numbers.

Frame 5 shows the result returned from clustering algorithms. The result has two keys:

- **Predicted:** This key returns an array of data with integer or float values and indicates which cluster the data from the predict key from Frame 4 belongs to.
- **Fit Predict:** This key returns an array of data with integer or float values and indicates which cluster the data from the data key from Frame 4 belongs to.

{
"Predicted": $[p_{11}, p_{12}, p_{13}, \dots, p_{1m}]$,
"Fit Predict": $[s_1, s_2, s_3, \dots, s_n]$
}

Frame 5. Result returned from clustering algorithms.

{
"y_predict": $[p_1, p_2, p_3, \dots, p_n]$,
"y_true": $[t_1, t_2, t_3, \dots, t_n]$,
"y_binary": $[b_1, b_2, b_3, \dots, b_n]$,
"y_scores": $[s_1, s_2, s_3, \dots, s_n]$
}

Frame 6. JSON Input format for metrics algorithms.

Frame 6 shows the format that covers the metrics algorithms. The format has four keys:

- **Y_predict:** This key returns an array that contains either integer or floats numbers and represents the predicted labels that were returned by a classifier.
- **Y_true:** This key represents the ground truth (correct) labels. These data can be represented as an array that contains either integers or float numbers.
- **t_binary:** This key represents the true binary labels in binary label indicators. These data can be represented as an array that contains either integer or float numbers.
- **Y_scores:** This key represents the target scores that can either be probability estimates of the positive class, confidence values, or binary decisions. These data can be represented as an array that contains either integers or float numbers.

Metrics algorithms are divided into three groups. Each group is responsible for measuring the results of the classification, cluster and regression algorithms. Also, the algorithms do not use all the keys described in Frame 6. Table 1 shows the algorithms with their respective descriptions and keys used.

Table 1. Metrics algorithms.

Algorithms	**Description**	**Keys**
	Metrics for classification algorithms	
Accuracy Score	In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must *exactly* match the corresponding set of labels in y_true.	y_true, y_predict
Average Precision Score	Compute average precision from prediction scores. This score corresponds to the area under the precision-recall curve. This implementation is restricted to the binary classification task or multilabel classification task.	y_binary, y_scores
Precision Score	Compute the precision. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative. The best value is 1 and the worst value is 0.	y_true, y_predict
Recall Score	Compute the recall. The recall is intuitively the ability of the classifier to find all the positive samples. The best value is 1 and the worst value is 0.	y_true, y_predict
Roc Auc Score	Compute Area Under the Curve from prediction scores. This implementation is restricted to the binary classification task or multilabel classification task in label indicator format.	y_binary, y_scores
	Metrics for cluster algorithms	
Adjusted Rand Score	Rand index adjusted for chance. The Rand Index computes a similarity measure between two clusters by considering all pairs of samples and counting pairs that are assigned in the same or in different clusters in the predicted and true clusters.	y_true, y_scores
	Metrics for regression algorithms	
Mean Squared Error	Mean squared error regression loss.	y_true, y_predict
R2 Score	R^2 (coefficient of determination) regression score function. Best possible score is 1.0, lower values are worse.	y_true, y_predict
Explained Variance Score	Explained variance regression score function. Best possible score is 1.0, lower values are worse.	y_true, y_predict

The statistics algorithms also have a format that is depicted in frame 7 and has four keys:

- Value: This key represents a value, which must be integer or float.
- Vector, vector 2: These keys represent an array of data with integer or float values.
- Matrix: This key represents the matrix that contains either integers or float numbers.

{

"value": value,

"vector ": $[v_1, v_2, v_3, \dots, v_n]$,

"vector2": $[v_1, v_2, v_3, \dots, v_n]$,

"matrix": $\begin{array}{l}[[d_{11}, d_{12}, d_{13}, \dots, d_{1m}],\\ \;[d_{21}, d_{22}, d_{23}, \dots, d_{2m}],\\ \;[\;\vdots\;,\;\vdots,\;\;\vdots\;,\vdots\;,\;\vdots\;\;],\\ \;[d_{n1}, d_{n2}, d_{n3}, \dots, d_{nm}]]\end{array}$

}

Frame 7. Json Input format for statistic algorithms.

Statistic algorithms are divided into three groups (orders, average and variances, correlating). The algorithms do not use all the keys described in Frame 7. Table 2 shows the algorithms with their respective descriptions and keys.

Table 2. Statistic algorithms.

Algorithms	**Description**	**Keys**
	Orders	
Minimum	Return the minimum of an array or minimum along an axis.	vector
Maximum	Return the maximum of an array or maximum along an axis.	vector
Nan Minimum	Return minimum of an array or minimum along an axis, ignoring any NaNs.	vector
Nan Maximum	Return the maximum of an array or maximum along an axis, ignoring any NaNs.	vector
Percentile	Compute the qth percentile of the data along the specified axis.	Matrix, value
	Correlating	
Correlation Coefficients	Return correlation coefficients.	vector
Correlate	Cross-correlation of two 1-dimensional sequences.	Vector, Vector2
Covariance Matrix	Estimate a covariance matrix, given data.	Matrix
	Average and Variances	
Average	Compute the weighted average along the specified axis.	vector
Mean	Compute the arithmetic mean along the specified axis.	vector
Median	Compute the median along the specified axis.	vector
Standard Deviation	Compute the standard deviation along the specified axis.	vector
Variance	Compute the variance along the specified axis.	vector

5. Web Application

A web application[1] was developed to provide regression, classification and clustering algorithms, in order to compare algorithms and to help users in understanding and choosing which algorithm fits in specific scenarios. Figure 1 show the initial interface.

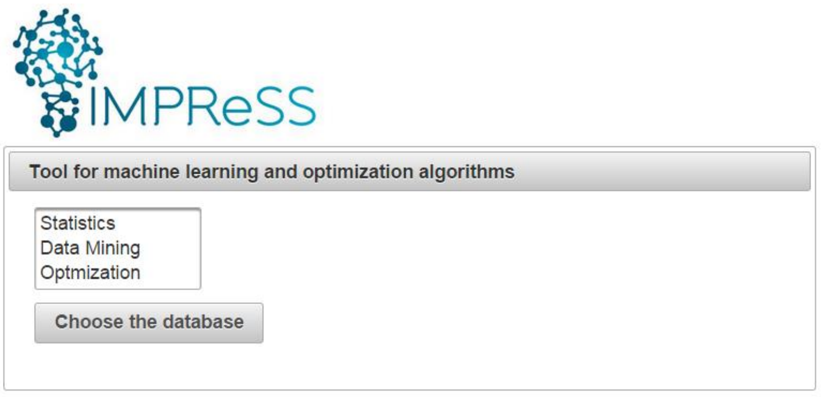


Figure 1: Web Application initial screen.

As emphasized by Figure 1, users are able to choose between statistics, data mining, and optimization algorithms. This deliverable only focuses on the data mining algorithms. After choosing "Data Mining", the user may select the category and the algorithm to be used. For instance, category Cluster and algorithm k-Means, as shown in Figure 2.

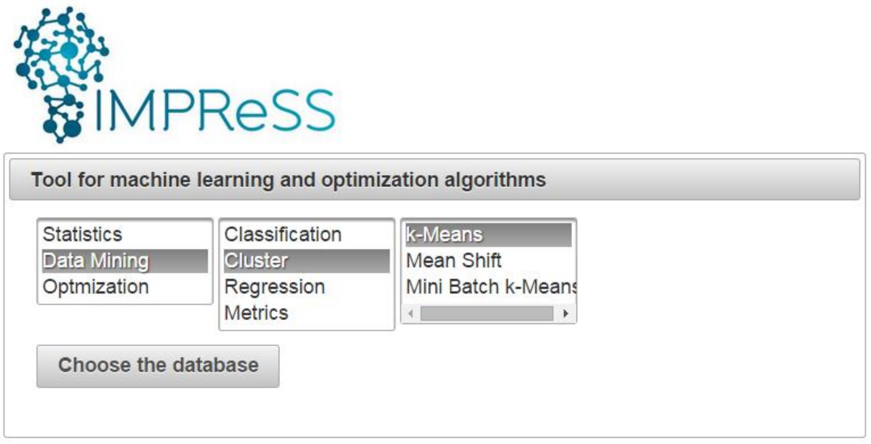


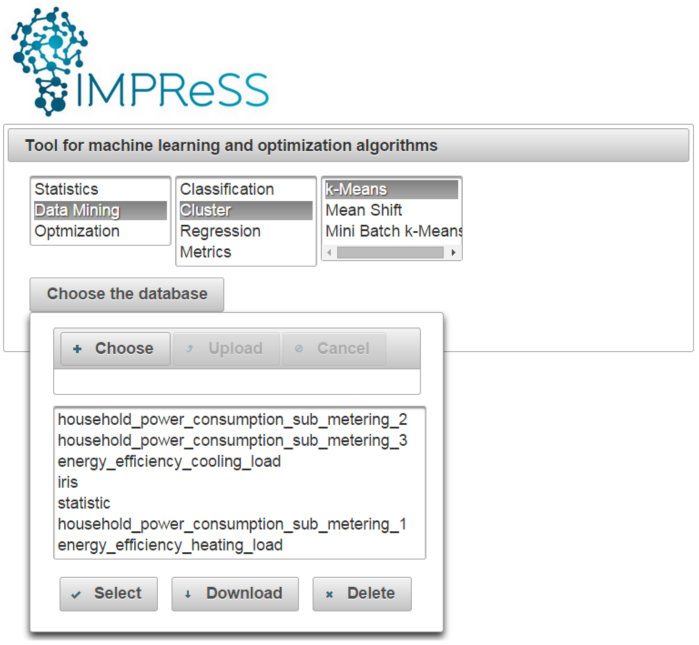
Figure 2: Web Application configuration.

After the user has chosen a category and an algorithm, a database has to be selected in order to allow the process to continue. As discussed in Section 4, the database must have a specific format. Figure 3 shows a snapshot of the screen. When clicking on the "choose the database" button from Figure 3, users may execute the following tasks:

1. **Choose:** Option that allows users to choose a database from their computer to upload it to the server.
2. **Upload:** This option starts the upload process of the database.
3. **Cancel:** Option that is used to cancel an upload.
4. **Select:** This option is used if the database is already at the server and the user just wants to select it again.
5. **Download:** Option used to download a database from the server to the computer.

[1] http://impressufam.cloudapp.net:8080/AlgorithmsTools

6. **Delete:** Option used to delete a database from the server.



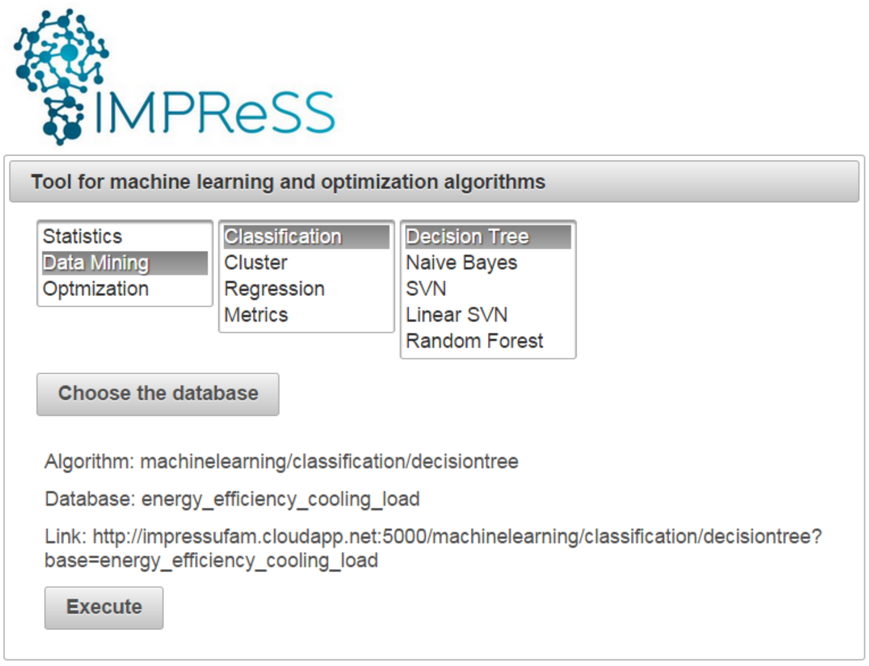
household_power_consumption_sub_metering_2
household_power_consumption_sub_metering_3
energy_efficiency_cooling_load
iris
statistic
household_power_consumption_sub_metering_1
energy_efficiency_heating_load

Figure 3: Web Application database selection.

After choosing these parameters, the users have to click on the "execute" button to start the execution of the algorithms and check the results. Next, an example of usage with real databases will be depicted.

5.1 Classification Example

To run the classification example, the decision tree algorithm was chosen along with the energy_efficient_cooling_load database. After selecting these parameters, the web application should looks like Figure 4.



Algorithm: machinelearning/classification/decisiontree

Database: energy_efficiency_cooling_load

Link: http://impressufam.cloudapp.net:5000/machinelearning/classification/decisiontree?base=energy_efficiency_cooling_load

Figure 4: Configuring a classification algorithm in the web application.

The goal of this example is to use the energy_efficient_cooling_load to discover the class of a sample. As explained in section 4, this sample is added in the "predict" key. In this case, the "[0.62,808.50,367.50,220.50,3.50,5,0.40,5]" sample was added, where each number into the array represents respectively the attributes of the sample. By clicking on the application execute button, it will be shown the following result, as depicted at Figure 5.

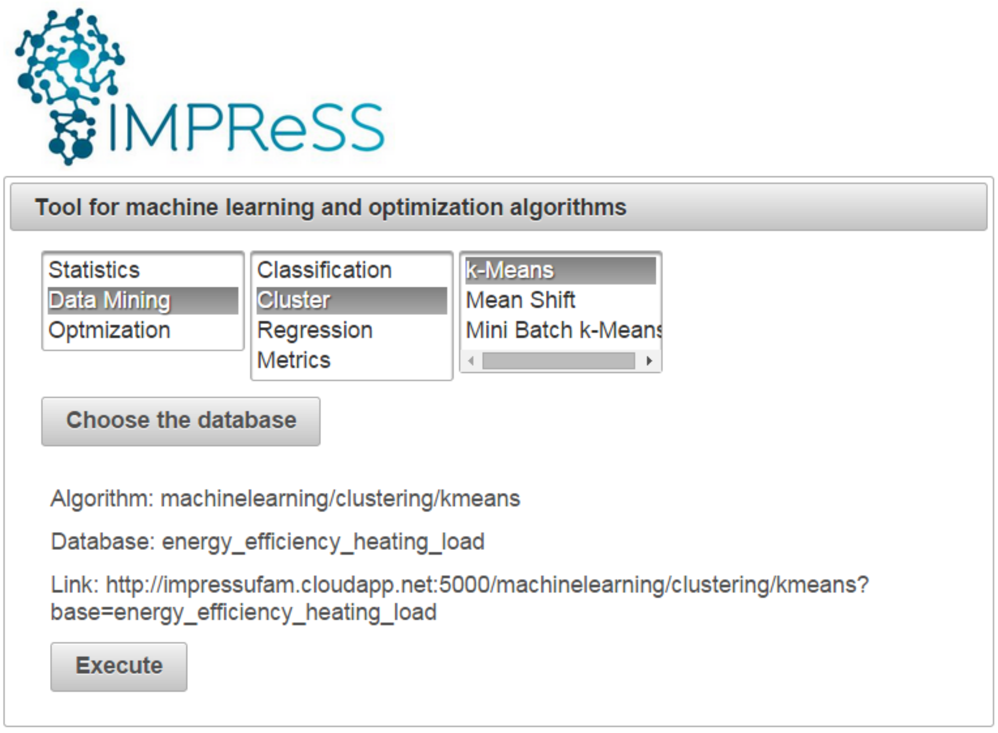
```
{"Predict": [16], "Score": [1.0]}
```

Figure 5: Web Application classification result.

This result shows that the sample "[0.62,808.50,367.50,220.50,3.50,5,0.40,5]" belongs to the class number 16 and that the algorithm score was 1.0.

5.2 Clustering Example

To run a clustering example, K-Means algorithm was chosen along with the energy_efficient_heating_load database. After selecting these parameters, the web application should looks like Figure 6.



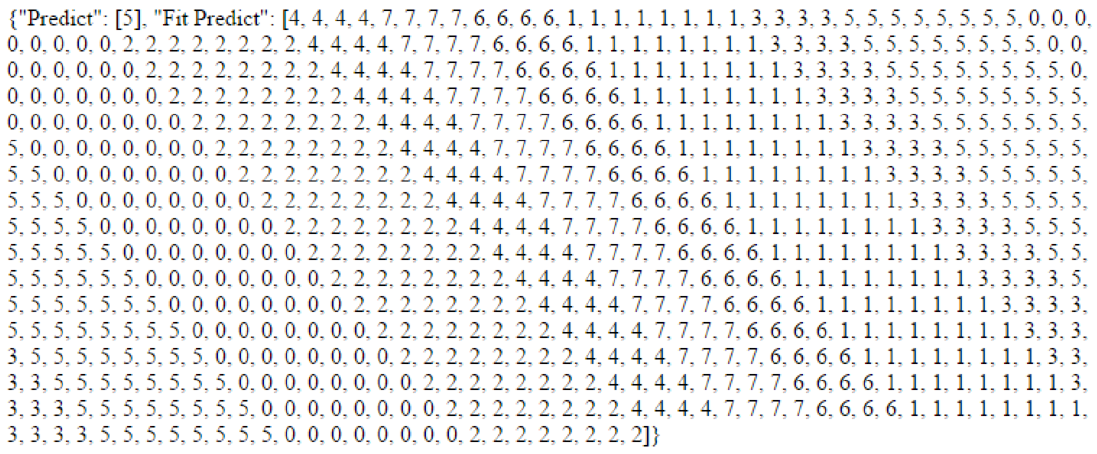
Algorithm: machinelearning/clustering/kmeans

Database: energy_efficiency_heating_load

Link: http://impressufam.cloudapp.net:5000/machinelearning/clustering/kmeans?base=energy_efficiency_heating_load

Figure 6: Configuring a clustering algorithm in the web application.

The goal of this example is to use the energy_efficient_heating_load to discover the group of a sample. Again, as explained in section 4, this sample is added in the "predict" key, in this case the "[0.62,808.50,367.50,220.50,3.50,5,0.40,5]" sample was added, where each number into the array represents respectively the attributes of the sample. By selecting on the application execute button, the following result depicted in Figure 7 will be obtained. The "predict" key shows that the sample belongs to group 5 and the "fit_predict" key shows the groups from the other data that are configured in the "data" key.

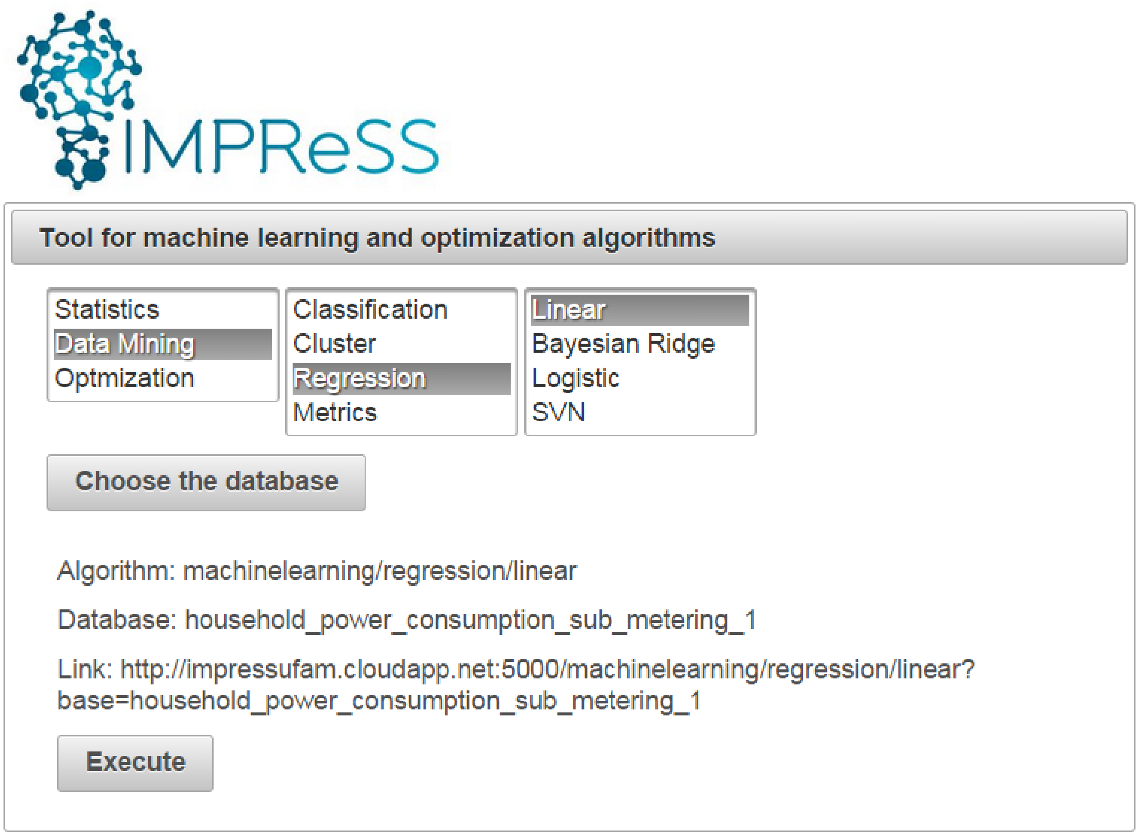


```
{"Predict": [5], "Fit Predict": [4, 4, 4, 4, 7, 7, 7, 7, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 5, 0, 0, 0,
0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 7, 7, 7, 7, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 5, 0, 0,
0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 7, 7, 7, 7, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 5, 0,
0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 7, 7, 7, 7, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 5,
0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 7, 7, 7, 7, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5,
5, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 7, 7, 7, 7, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 5, 5, 5, 5, 5, 5,
5, 5, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 7, 7, 7, 7, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 5, 5, 5, 5, 5,
5, 5, 5, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 7, 7, 7, 7, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 5, 5, 5, 5,
5, 5, 5, 5, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 7, 7, 7, 7, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 5, 5, 5,
5, 5, 5, 5, 5, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 7, 7, 7, 7, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 5, 5,
5, 5, 5, 5, 5, 5, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 7, 7, 7, 7, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 5,
5, 5, 5, 5, 5, 5, 5, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 7, 7, 7, 7, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3,
5, 5, 5, 5, 5, 5, 5, 5, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 7, 7, 7, 7, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3,
3, 5, 5, 5, 5, 5, 5, 5, 5, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 7, 7, 7, 7, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3,
3, 3, 5, 5, 5, 5, 5, 5, 5, 5, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 7, 7, 7, 7, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 3,
3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 5, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 7, 7, 7, 7, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1,
3, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 5, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2]}
```

Figure 7: Web Application clustering result.

5.3 Regression Example

In order to run a regression example, Linear Regression algorithm was chosen to learn household_power_consumption_sub_metering_1 database. After selecting these parameters, the web application should looks like Figure 8.



Algorithm: machinelearning/regression/linear

Database: household_power_consumption_sub_metering_1

Link: http://impressufam.cloudapp.net:5000/machinelearning/regression/linear?base=household_power_consumption_sub_metering_1

Figure 8: Configuring a regression algorithm in the web application.

The goal of this example is to use the household_power_consumption_sub_metering_1 to train data from January and to predict the values that should appear in February. The results are shown in the "predict" key, but they are too large to show in a figure.

The examples provided in this section show how the learning algorithms provided in the web service may be used, as well as how the input databases must be formatted.

6. References

[1] Scikit-learn:machine learning in Pythong. Available in: http://scikit-learn.org/stable/. Date Acess: 10/01/2015.

[2] Oracle® Database Concepts 11g Release 1 (11.1). Available in: http://docs.oracle.com/cd/B28359_01/server.111/b28318.pdf. Date Acess: 07/11/2014.

[3] Introduction to Linear Regression. Available in: http://onlinestatbook.com/2/regression/intro.html Date Acess: 10/01/2015.

[4] Linear Regression. Available in: http://en.wikipedia.org/wiki/Linear_regression Date Acess: 10/01/2015.

[5] Machine Learning with Python - Linear Regression. Available in: http://aimotion.blogspot.com.br/2011/10/machine-learning-with-python-linear.html Date Acess: 13/01/2015.

[6] Machine Learning with Python – Logistic Regression. Available in: http://aimotion.blogspot.com.br/2011/11/machine-learning-with-python-logistic.html Date Acess: 13/01/2015.

[7] Support Vector Machines for Regression. Available in: http://www.svms.org/regression/ Date Acess: 13/01/2015.

[8] Clustering – K-means. Available in: http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html Date Acess: 14/01/2015.

[9] Clustering – scikit-learn 0.15.2 documentation. Available in: http://scikit-learn.org/stable/modules/clustering.html#mini-batch-kmeans Date Acess: 14/01/2015.

[10] Introduction to Mean Shift algorithm. Available in: https://saravananthirumuruganathan.wordpress.com/2010/04/01/introduction-to-mean-shift-algorithm/ Date Acess: 03/01/2015.

[11] Oracle® Database Concepts 10g Release 1 (10.1). Available in: http://docs.oracle.com/pdf/B10698_01.pdf. Date Acess: 07/11/2014.

[12] Random Forest. Available in: http://www.statsoft.com/Textbook/Random-Forest. Date Acess: 10/01/2015.

[13] Random Forest – classification description. Available in: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm. Date Acess: 10/01/2015.

[14] Decision Tree. Available in: http://www.saedsayad.com/decision_tree.htm. Date Acess: 10/01/2015.

[15] Decision tree. Available in: http://en.wikipedia.org/wiki/Decision_tree. Date Acess: 10/01/2015.

[16] Naive Bayes classifier. Available in: http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Naive_Bayes_classifier.html. Date Acess: 10/01/2015.

[17] UCI Machine Learning Repository. Available in: http://archive.ics.uci.edu/ml/. Date Acess: 03/01/2015.