Soft Computing

Attribute Dependency Data Analysis For Massive Datasets By Fuzzy Transforms

--Manuscript Draft--

Manuscript Number:	SOCO-D-19-01350R1
Full Title:	Attribute Dependency Data Analysis For Massive Datasets By Fuzzy Transforms
Article Type:	Original Research
Keywords:	attribute dependency; data mining; fuzzy transform
Corresponding Author:	Ferdinando Di Martino, Ph.D. Universita degli Studi di Napoli Federico II Dipartimento di Architecttura Napoli, ITALY
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Universita degli Studi di Napoli Federico II Dipartimento di Architecttura
First Author:	Ferdinando Di Martino, Ph.D.
First Author Secondary Information:	
Order of Authors:	Ferdinando Di Martino, Ph.D.
	Salvatore Sessa
Funding Information:	
Abstract:	We present a numerical attribute dependency method for massive datasets based on the concepts of direct and inverse fuzzy transform. In a previous work we used these concepts for numerical attribute dependency in data analysis: therein the multi-dimensional inverse fuzzy transform was useful for approximating a regression function. Here we give an extension of this method in massive datasets because the previous method could not be applied due to the high memory size. Our method is proved on a large dataset formed from 402678 census sections of the Italian regions provided by the Italian National Statistical Institute (ISTAT) in 2011. The results of comparative tests with the well-known methods of regression, called Support Vector Regression and Multilayer Perceptron, show that the proposed algorithm has comparable performance with those obtained using these two methods. Moreover the number of parameters requested in our method is minor with respect to those of the cited in the above two algorithms.
Section/Category:	Methodologies & Application

Dear editors,
We thank very much the reviewers for detailed and valuable comments which improved greatly the quality of the paper. According to the comments, we have made the due amendments to our paper. In red we add our comments and our changes in the paper. We hope that the new version can be suitable for publication.

Reviewer 1

In this paper, the authors provided an attribute dependency data analysis for massive datasets. Here are my comments:

1. I don't think the proposed dataset which contains 402678 data point can be called massive dataset or even big data, since as my experience, it just a small-medium datasets. If the author want to consider massive dataset problems, I think TB is the basic unit and parallel processes, such as SPARK or HADOOP framework should be considered. Clearly, it just a normal data mining problem.
Thanks for these suggestions. The dataset considered in our test consists of a massive dataset composed of all socio-economic census data relating to the resident population, foreigners, families, buildings and properties for all the census areas of all the municipalities of Italy. Each entity is made up of 140 numeric attributes. While not a very large dataset, it can represent a massive dataset; we preferred to use this dataset in our tests to be able to make a complete comparison of the algorithm with the FAD algorithm, which cannot be used in presence of a VL dataset. we intend in the future to experiment with MFADs on many massive datasets of different sizes in order to in the future we intend to experiment with MFADs on many massive datasets of different sizes in order to analyze its performances in detail and verify if the choice of the optimal values of the number of subsets and the of threshold value of the index of determinacy depend on the type of dataset.

2. I don't think the authors are the experts of soft computing. For example, multilayer perceptron should be abbreviated as MLP, rather than MP, and the references are not appropriate, since these they are not important papers to the field.

We apologize to the reviewer. The abbreviation in MP instead of MLP dii Multilayer Perceptron is just a typo that we have corrected in the text. Furthermore, we have also completely updated the references to it in the bibliography. These references are highlighted in red in the references section.

3. The authors stated that SVR and MLP methods can't handle a high number of parameters. Actually, MLP can easily handle hundreds of features today. Besides, we have lots of skills, such as various autoencoders, to handle hundreds and hundreds of parameters in neural networks.

Thanks for these suggestions, which allowed us to understand that we had misrepresented the problem we intended to highlight in the use of MLPs. We refer to the parameters to be set for the execution of an MLP. MLPs require the user to set various parameters need for training, as, for example, the number of hidden layers, the number of neurons per layer, the type of nonlinear activation function, etc. MFAD require to set only the number of subsets and the threshold value of the index of determinacy. This makes MFAD more user-friendly than MLP and SVR. MFAD could represent a trade-off between usability and high performance in the use of massive datasets.

4. The experiments only reflect how their methods process. There is no comparison with other methods and no justification to show their method is better. In addition, I still don't their method has a significant contribution or need in the soft computing field.

In our tests ve've compared MFAD also with MLP and SVR, showing that MFAD algorithm has performances comparable with SVR and MLP. MFAD. To compare the three algorithms, we've measured the index of determinacy given in eq. (14) in all test executed. The results shown that the difference between the index of determinacy obtained using SVR and the one obtained using MFAD is always under 0.02 and the difference between the index of determinacy obtained using SVR and the one obtained using MFAD is always under 0.016. These results allow us to conclude that MFAD provides acceptable performance in the detection of attribute dependencies in the presence of massive datasets. Therefore, unlike FAD, it can be applied to massive data and can represent a trade-off between usability and high performance in detecting attribute dependencies in massive datasets.

5. Typos and grammatical error are easily found in this paper. In my opinion, this paper is still far away to be considered in a journal.

Thanks for these suggestions. All typos and grammatical errors have been corrected.

Reviewer 3

Brief Summary

The paper named "Attribute Dependency Data Analysis For Massive Datasets By Fuzzy Transforms" proposed by

Ferdinando Di Martino and Salvatore Sessa deals with identifying dependencies between attributes of a

large dataset using direct and inverse fuzzy transform. They proposed an efficient algorithm (MFAD)

designed to work on massive data, capable of identifying them. The algorithm was tested on a benchmark dataset,

made available by ISTAT (Italian National Statistical Institute). The approach used by the authors allows the use of a smaller number of features, enhancing the operation of identifying the relationships between the features and therefore improving the Dimensionality Reduction operation. I found this contribution particularly interesting as it allows you to speed up a normal learning process that would normally be performed on multiple features.

In my opinion the paper is well written. I think the contribution is interesting and it deserves to be published after a minor revision.

We truly appreciate your encouragement, careful review, and valuable suggestions.

General Comments

----

1) I recommend to number the references and insert the number next to each one, it makes easier to finding the cited article.

We've numbered the references and insert the corresponding number to cite them in the manuscript. All the number of the references are highlighted in red in the text.

2) Number the references also in the references section

All references are now numbered in the references section.

Attribute Dependency Data Analysis For Massive Datasets By Fuzzy Transforms

Ferdinando Di Martino, Salvatore Sessa

Università degli Studi di Napoli Federico II, Dipartimento di Architettura, Via Toledo 402, 80134 Napoli, Italy
Università degli Studi di Napoli Federico II, Centro Interdipartimentale di Ricerca "A. Calza Bini", via Toledo 402, Napoli, Italy
email: fdimarti@unina.it, sessa@unina.it

Abstract. We present a numerical attribute dependency method for massive datasets based on the concepts of direct and inverse fuzzy transform. In a previous work we used these concepts for numerical attribute dependency in data analysis: therein the multi-dimensional inverse fuzzy transform was useful for approximating a regression function. Here we give an extension of this method in massive datasets because the previous method could not be applied due to the high memory size. Our method is proved on a large dataset formed from 402678 census sections of the Italian regions provided by the Italian National Statistical Institute (ISTAT) in 2011. The results of comparative tests with the well-known methods of regression, called Support Vector Regression and Multilayer Perceptron, show that the proposed algorithm has comparable performance with those obtained using these two methods. Moreover the number of parameters requested in our method is minor with respect to those of the cited in the above two algorithms.

Keywords: attribute dependency, data mining, fuzzy transform

1. Introduction

Data analysis and data mining knowledge discovery processes represent powerful functionalities that can be combined in knowledge based expert and intelligent systems in order to extract and build knowledge starting by data. In particular, attribute dependency data analysis is an activity necessary to reduce the dimensionality of the data and to detect hidden relations between features. Nowadays in many application fields data sources are massive (for example, web social data, sensor data, etc.) and it is necessary to implement knowledge extraction methods that can operate on massive data. Massive (Very Large (VL) and Large (L)) datasets [3] are produced and updated and they cannot be managed by traditional databases. Today the access via web to these datasets has led to develop technologies for managing them (cfr., e.g., [7], [25], [35]).

We recall the regression analysis (cfr., e.g., [15], [18],[22], [23], [31]) for estimating relationships among variables in the datasets (cfr., e.g., [24], [26], [36], [39]) and fuzzy tools for attribute dependency ([38], [42]).

Machine learning soft computing models were proposed in literature to perform nonlinear regressions on high dimensional data; two well known machine learning nonlinear regression algorithms are Support Vector Regression (SVR) [16] and multilayer perceptron (MLP) (cfr., e.g., [5], [6], [19], [20], [21], [27], [33]) algorithms. The main problems of these algorithms are the complexity of the model due to the presence of many parameters to be set by the user, and the presence of overfitting, phenomenon in which the regression function fits optimally the training set data, but fails in predictions on new data. K-fold cross validation techniques are proposed in literature to avoid overfitting [1]. In [37] a pruning method based on variance sensitivity analysis is proposed to find the optimal structure of a multilayer perceptron in order to mitigate overfitting problems. In [17] a novel sparse-coding kernel algorithm is proposed to overcome overfitting in disease diagnosis.

Some authors proposed variations of nonlinear machine learning regression models to manage massive data. In ([34], [4]) a fast-local support vector machine (SVM) method to manage large datasets are presented in which a set of multiple local SVMs for low dimensional data are constructed. In [43] the authors proposed an incremental version of the vector machine regression model to manage large-scale data. In [28] the authors proposed a parallel architecture of a logistic regression model for massive data management. Recently variations of the Extreme Learning Machine (ELM) regression methods for massive data based on the MapReduce model are presented ([2], [41]).

The presence of a high number of parameters makes SVR and MLP methods too complex to be integrated as components into an intelligent or expert system. In this research we propose a model of attribute dependency in massive datasets based on the use of the multi-dimensional fuzzy transform. We extend the attribute dependency method presented in [9] to massive datasets in which the inverse multi-dimensional fuzzy transform is used as a regression function. Our goal is to guarantee a high performance of the proposed method in the analysis of massive data, maintaining, at the same time, the usability of the previous multi-dimensional fuzzy transform attribute dependency. As in [23], we use a random sampling algorithm for subdividing the dataset in subsets of equal cardinality.

The fuzzy transform (F-transform) method [29] is a technique which approximates a given function by means of another function unless an arbitrary constant. This approach is particularly flexible in the applications such as image processing (cfr., e.g., [8], [10], [12], [13], [14]), data analysis (cfr., e.g., [9], [11], [30]). In this last work an algorithm, called FAD (F-transform Attribute Dependence), evaluates an attribute X_z depending from k attributes $X_1, \ldots, X_k$ (predictors) with $z \notin \{1,2,\ldots k\}$, i.e. $X_z = H(X_1, \ldots, X_k)$, and the (unknown) function H is approximated with the inverse multi-dimensional F-transform via a procedure presented in [30]. The error of this approximation in [9] is measured from a statistical index of determinacy ([15], [22]). If it overcomes a prefixed threshold, then the functional dependency is found. Each attribute has an interval $X_i = [a_i,b_i]$, i = 1,…,k, as domain of knowledge. Then an uniform fuzzy partition (whose definition is given in Section 2) of fuzzy sets $\left\{A_{i1}, A_{i2}, \ldots, A_{in_i}\right\}$ defined on $[a_i,b_i]$ is created assuming $n_i \geq 3$.

The main problem in the use of the inverse F-transform for approximating the function H consists in the fact that the data are not sufficiently dense with respect to the fuzzy partitions. The FAD algorithm solves this problem with an iterative process which shall been clearly in Section 3. If the data are not sufficiently dense with respect to the fuzzy partitions, the process stops otherwise an index of determinacy is calculated. If this index is greater than a threshold α, the functional dependency is found and the inverse F- transform is considered as approximation of the function H, otherwise a finer fuzzy partition is set with n:= n+1. The FAD algorithm is schematized in Fig. 1.

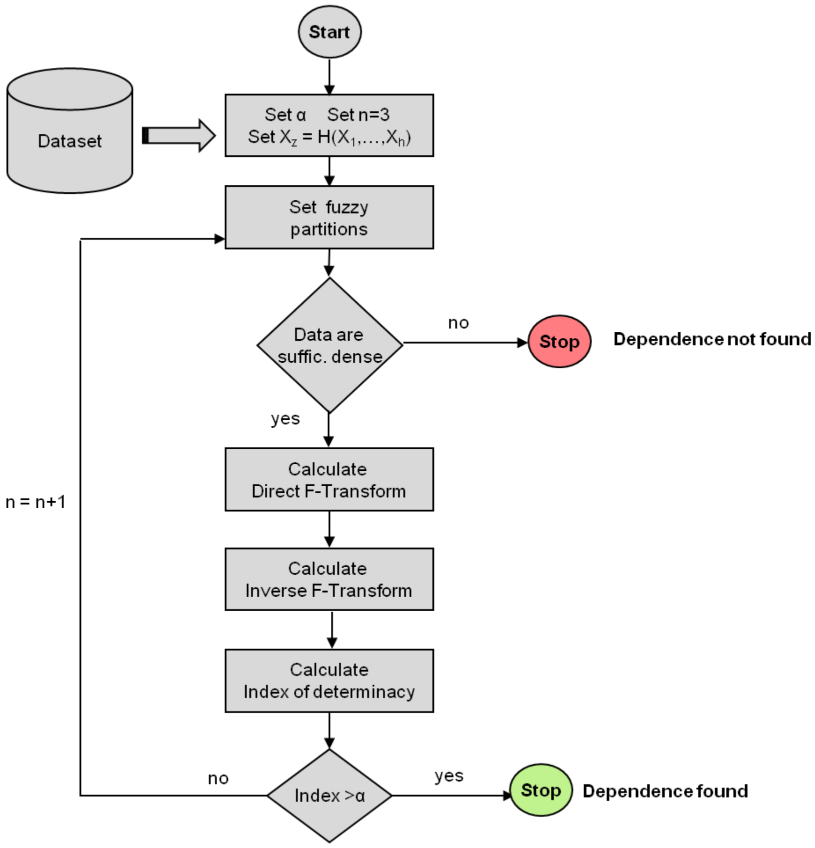

Fig. 1. Flux diagram of the FAD algorithm

In this paper we propose an extension of the FAD algorithm, called MFAD (Massive F-transform Attribute Dependency) for finding dependencies between numerical attributes in massive datasets. In other words, by using a uniform sampling method, we can apply the algorithm of [9] to several sample subsets of the data and hence we extend the results obtained to the overall dataset with suitable mathematical artifices.

Indeed, the dataset is partitioned randomly in s subsets having equal cardinality to which we apply the F-transform method.

Let $D_l = [a_{1l}, b_{1l}] \times \ldots \times [a_{kl}, b_{kl}]$, $l = 1,\ldots,s$, be the Cartesian product of the domains of the attributes X_1, X_2,…, X_k, where a_{il} and b_{il} are the minimum and maximum values of X_i in the l*th* subset. Hence the multi-dimensional inverse F-transform $H^F_{n_{1l} n_{2l} \ldots n_{kl}}$ is calculated for approximating the function H in the domain D_l and an index of determinacy r_{cl}^2 is calculated for evaluating the error in the approximation of H with $H^F_{n_{1l}n_{2l}\ldots n_{kl}}$ in D_l. For simplicity, we put $n_{1l} = n_{2l} = \ldots = n_{kl} = n_l$ and thus $H^F_{n_{1l}n_{2l}\ldots n_{kl}} = H^F_{n_l}$. In order to obtain the final approximation of H, we introduce weights for considering the contribute of the inverse F-transform $H^F_{n_l}$ in the approximation of H. We calculate the weighted mean of $H^F_{n_1}, \ldots, H^F_{n_s}$ replacing the weights with the indices of determinacy $r^2_{c1}, \ldots, r^2_{cs}$.

Calculate the approximated value of H^F in $(x_1, \ldots, x_k) \in \bigcup_{l=1}^{s} D_l$ given by

$$H^F(x_1, x_2, \ldots, x_k) = \frac{\sum_{l=1}^{s} w_l(x_1, x_2, \ldots, x_k) \cdot H^F_{n_l}(x_1, x_2, \ldots, x_k)}{\sum_{l=1}^{s} w_l(x_1, x_2, \ldots, x_k)} \quad (1)$$

where

$$w_l(x_1, x_2, \ldots, x_k) = \begin{cases} r^2_{cl} & \text{if } (x_1, x_2, \ldots, x_k) \in D_l \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

For example, we consider two attributes, X_1 and X_2, as inputs and suppose, for simplicity, that the dataset is partitioned in two subsets. Fig. 2 shows two rectangles D_1 (red) and D_2 (green). The zone labeled as A of the input space is covered by the domain D_2: in this zone the weight w_1 is null and $H^F = H^F_2$. Conversely, in the zone C the contribute of H^F_2 is null and $H^F = H^F_1$. In the zone labeled as B, the inverse F-transforms, calculated for both subsets, contribute to the final evaluation of H, with a weight corresponding to the index of determinacy.

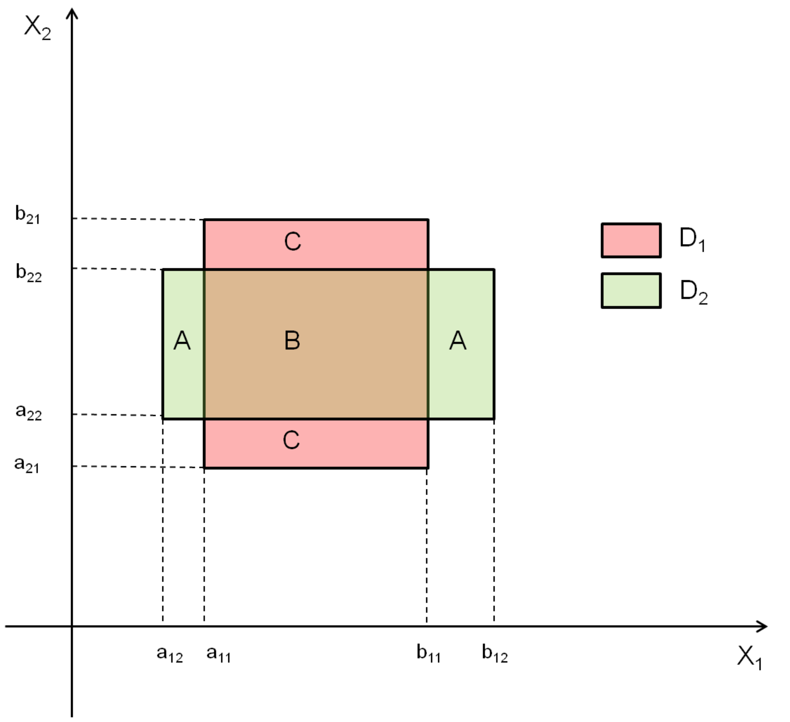

Fig. 2. Example of union of domains of the subsets in which the dataset is partitioned

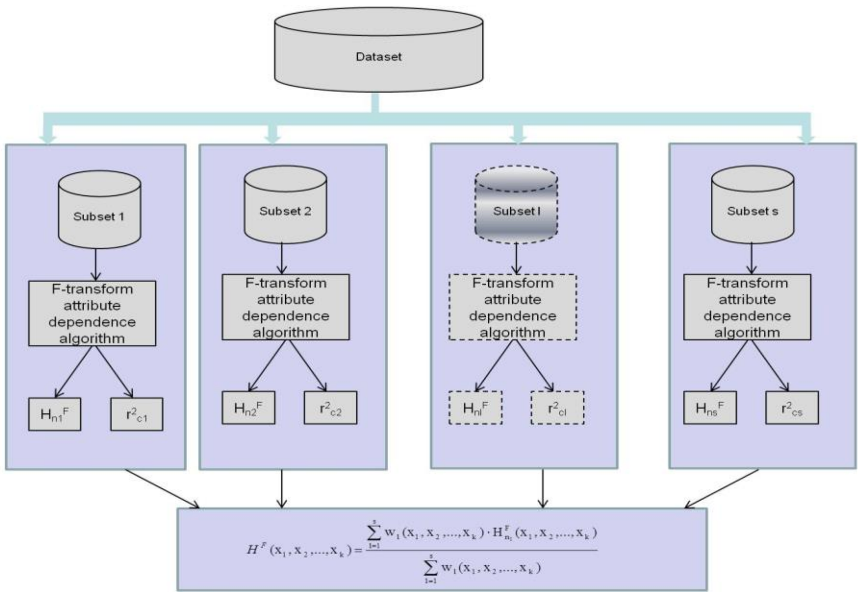

Fig. 3. Schema of the MFAD method

Fig. 3 contains the schema of MFAD. We apply our method on a L dataset loadable in memory, so we can apply also the method of [9] and hence we compare the results obtained by using both methods. As test dataset we consider the last Italian census data acquired during 2011 by ISTAT (Italian National Statistical Institute). Section 2 contains the F-transform in one and more variables [30]. In Section 3 the F-transform attribute dependency method is presented, Section 4 contains the results of our tests. Conclusions are described in Section 5.

2. F-transforms in one and k variables

Following the definitions of [29]. We recall the main notations for making this paper self-contained. Let $n \geq 2$, $x_1, x_2, \ldots, x_n$ be points (nodes) of [a,b], $x_1 = a < x_2 < \ldots < x_n = b$. The fuzzy sets $A_1,\ldots,A_n : [a,b] \to [0,1]$ (basic functions) constitute a fuzzy partition of [a,b] if $A_i(x_i) = 1$ for $i = 1,2,\ldots,n$; $A_i(x) = 0$ if $x \notin (x_{i-1}, x_{i+1})$ for $i=2,\ldots,n$; $A_i(x)$ is a continuous on [a,b]; $A_i(x)$ strictly increases on $[x_{i-1}, x_i]$ for $i = 2, \ldots, n$ and strictly decreases on $[x_i, x_{i+1}]$ for $i = 1,\ldots, n-1$; $\sum_{i=1}^{n} A_i(x) = 1$ for every $x \in [a,b]$. The partition $\{A_1(x),\ldots,A_n(x)\}$ is said uniform if $n \geq 3$, $x_i = a + h\cdot(i-1)$, where $h = (b-a)/(n-1)$ and $i = 1, 2, \ldots, n$ (equidistance); $A_i(x_i - x) = A_i(x_i + x)$ for $x \in [0,h]$ and $i = 2,\ldots, n-1$; $A_{i+1}(x) = A_i(x - h)$ for $x \in [x_i, x_{i+1}]$ and $i = 1,2,\ldots, n-1$.

We know that the function f assumes given values in the points $p_1,\ldots,p_m$ of [a,b],. If the set $P = \{p_1,\ldots,p_m\}$ is sufficiently dense with respect to $\{A_1, A_2, \ldots, A_n\}$, that is for every $i \in \{1,\ldots,n\}$ there exists an index $j \in \{1,\ldots,m\}$ such that $A_i(p_j) > 0$, then the n-tuple $[F_1, F_2, \ldots, F_n]$ is the discrete direct F-transform of f with respect to $\{A_1, A_2, \ldots, A_n\}$, where each F_i is given by

$$F_i = \frac{\sum_{j=1}^{m} f(p_j) A_i(p_j)}{\sum_{j=1}^{m} A_i(p_j)} \tag{3}$$

for i=1,…,n. Then we define the discrete inverse F-transform of f with respect to the basic functions $\{A_1, A_2, \ldots, A_n\}$ by setting

$$f_{F,n}(p_j) = \sum_{i=1}^{n} F_i A_i(p_j) \tag{4}$$

for every $j \in \{1,\ldots,m\}$. Now we recall concepts from (Perfilieva, Novak, & Dvorak, 2008). The F-transforms can be extended to k (≥ 2) variables considering the Cartesian product of intervals $[a_1,b_1]\times[a_2,b_2]\times\ldots\times[a_k,b_k]$. Let $x_{11}, x_{12},\ldots, x_{1n_1} \in [a_1,b_1]$,…, $x_{k1}, x_{k2},\ldots, x_{kn_k} \in [a_k,b_k]$ be $n_1+\ldots+n_k$ assigned points (nodes) such that $x_{i1}=a_i<x_{i2}<\ldots<x_{in_i}=b_i$ and $\{A_{i1}, A_{i2},\ldots, A_{in_i}\}$ be a fuzzy partition of $[a_i,b_i]$ for i = 1,…,k. Let the function $f(x_1,x_2,\ldots,x_k)$ be assuming values in m points $p_j = (p_{j1}, p_{j2},\ldots,p_{jk}) \in [a_1,b_1]\times[a_2,b_2]\times\ldots\times[a_k,b_k]$ for j=1,…,m. The set $P=\{(p_{11}, p_{12}, \ldots, p_{1k}), (p_{21}, p_{22}, \ldots, p_{2k}),\ldots,(p_{m1}, p_{m2},\ldots,p_{mk})\}$ is said sufficiently dense with respect to $\{A_{11}, A_{12},\ldots, A_{1n_1}\},\ldots,\{A_{k1}, A_{k2},\ldots, A_{kn_k}\}$ if for $\{h_1,\ldots,h_k\}\in\{1,\ldots,n_1\}\times\ldots\times\{1,\ldots,n_k\}$ there exists $p_j = (p_{j1},p_{j2},\ldots,p_{jk})\in P$ with $A_{1h_1}(p_{j1})\cdot A_{2h_2}(p_{j2})\cdot\ldots\cdot A_{kh_K}(p_{jk})>0$, $j\in\{1,\ldots,m\}$. Then we define the $(h_1,h_2,\ldots,h_k)$*th* component $F_{h_1h_2\ldots h_K}$ of the discrete direct F-transform of f with respect to $\{A_{11}, A_{12},\ldots, A_{1n_1}\}$, …, $\{A_{k1}, A_{k2},\ldots, A_{kn_k}\}$ as

$$F_{h_1h_2\ldots h_K} = \frac{\sum_{j=1}^{m} f(p_{j1}, p_{j2},\ldots p_{jk})\cdot A_{1h_1}(p_{j1})\cdot A_{2h_2}(p_{j2})\cdot\ldots\cdot A_{kh_K}(p_{jk})}{\sum_{j=1}^{m} A_{1h_1}(p_{j1})\cdot A_{2h_2}(p_{j2})\cdot\ldots\cdot A_{kh_K}(p_{jk})} \tag{5}$$

Thus we define the discrete inverse F-transform of f with respect to $\{A_{11}, A_{12},\ldots, A_{1n_1}\}$, …, $\{A_{k1}, A_{k2},\ldots, A_{kn_K}\}$ by setting for $p_j=(p_{j1}, p_{j2},\ldots,p_{jk})\in[a_1,b_1]\times\ldots\times[a_k,b_k]$:

$$f^F_{n_1n_2\ldots n_K}(p_{j1}, p_{j2},\ldots, p_{jk}) = \sum_{h_1=1}^{n_1}\sum_{h_2=1}^{n_2}\ldots\sum_{h_K=1}^{n_k} F_{h_1h_2\ldots h_K}\cdot A_{1h_1}(p_{j1})\cdot\ldots\cdot A_{kh_K}(p_{jk}) \tag{6}$$

for j=1,…,m. The following Theorem holds [29]:

Theorem 1. Let $f(x_1,x_2,\ldots,x_k)$ be a function assigned on the set of points $P = \{(p_{11},p_{12},\ldots,p_{1k}), (p_{21}, p_{22}, \ldots, p_{2k}),\ldots,(p_{m1}, p_{m2},\ldots,p_{mk})\}\subseteq[a_1,b_1]\times[a_2,b_2]\times\ldots\times[a_k,b_k]$. Then for every ε > 0, there exist k integers $n_1(\varepsilon),\ldots, n_k(\varepsilon)$ and related fuzzy partitions

$$\{A_{11}, A_{12},\ldots, A_{1n_1(\varepsilon)}\},\ldots,\{A_{k1}, A_{k2},\ldots, A_{kn_k(\varepsilon)}\} \tag{7}$$

such that the set P is sufficiently dense with respect to fuzzy partitions (5) and for every $p_j = (p_{j1}, p_{j2},\ldots,p_{jk}) \in P$, j=1,…,m, the following inequality holds:

$$\left| f(p_{j1}, p_{j2},\ldots, p_{jk}) - f^F_{n_1(\varepsilon)n_2(\varepsilon)\cdots n_k(\varepsilon)}(p_{j1}, p_{j2},\ldots, p_{jk})\right| < \varepsilon \tag{8}$$

3. Multi-dimensional algorithm for massive datasets

3.1 FAD Algorithm

We schematize a dataset in tabular form as

	X_1	...		X_i	...	X_r
O_1	p_{11}		.	p_{1i}	.	p_{1r}
.	.	.		.	.	.
.	.	.		.	.	.
.	.	.		.	.	.
O_j	p_{j1}	.		p_{ji}	.	p_{jr}
.	.	.		.	.	.
.	.	.		.	.	.
.	.	.		.	.	.
O_m	p_{m1}	.		p_{mi}	.	p_{mr}

Here $X_1,\ldots,X_i,\ldots,X_r$ are the involved attributes and $O_1,\ldots,O_j,\ldots,O_m$ $(m>r)$ are the instances and p_{ji} is the value of the attribute X_i for the instance O_j. Each attribute X_i can be considered as a numerical variable assuming values in the domain $[a_i,b_i]$, where $a_i = \min\{p_{1i},\ldots,p_{mi}\}$ and $b_i = \max\{p_{1i},\ldots,p_{mi}\}$. We analyse the functional dependency between attributes in the form:

$$X_z = H(X_1,\ldots,X_k) \tag{9}$$

where $z\in\{1,\ldots, r\}$, $k \leq r < m$, $X_z \neq X_1, X_2, \ldots, X_k$, $H: [a_1,b_1]\times[a_2,b_2]\times\ldots\times[a_k,b_k] \rightarrow [a_z,b_z]$ is continuous. In $[a_i,b_i]$, $i = 1,2,\ldots,k$, an uniform partition of $\{A_{i1},\ldots,A_{ij},\ldots,A_{in}\}$ is defined for $i = 1,\ldots, k$ and $j = 2,\ldots, k-1$:

$$A_{i1}(x) = \begin{cases} 0.5\cdot\left(1+\cos\dfrac{\pi}{h_i}(x-x_{i1})\right) & \text{if } x\in[x_{i1}, x_{i2}] \\ 0 & \text{otherwise} \end{cases}$$

$$A_{ij}(x) = \begin{cases} 0.5\cdot\left(1+\cos\dfrac{\pi}{h_i}(x-x_{ij})\right) & \text{if } x\in[x_{i(j-1)}, x_{i(j+1)}] \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

$$A_{in}(x) = \begin{cases} 0.5\cdot\left(1+\cos\dfrac{\pi}{h_i}(x-x_{in})\right) & \text{if } x\in[x_{i(n-1)}, x_{in}] \\ 0 & \text{otherwise} \end{cases}$$

where $h_i = (b_i - a_i)/(n - 1)$, $x_{ij} = a_i + h_i\cdot(j-1)$.

By setting $H(p_{j1},p_{j2},\dots,p_{jk}) = p_{jz}$ for $j=1,2,\dots,m$, the components of H are given by

$$F_{h_1 h_2 \dots h_k} = \frac{\sum_{j=1}^{m} p_{jz} \cdot A_{1h_1}(p_{j1}) \cdot \dots \cdot A_{k h_K}(p_{jk})}{\sum_{j=1}^{m} A_{1h_1}(p_{j1}) \cdot \dots \cdot A_{k h_K}(p_{jk})} \tag{11}$$

The inverse F-transform $H^F_{n_1 n_2 \dots n_k}$ is defined as

$$H^F_n(p_{j1}, p_{j2}, \dots p_{jk}) = \sum_{h_1=1}^{n} \sum_{h_2=1}^{n} \dots \sum_{h_k=1}^{n} F_{h_1 h_2 \dots h_K} \cdot A_{1h_1}(p_{j1}) \cdot \dots \cdot A_{kh_K}(p_{jk}) \tag{12}$$

The error of the approximation is evaluated in $(p_{j1},p_{j2},\dots,p_{jm})$ by using the following statistical index of determinacy (Draper & Smith, 1988; Johnson & Wichern, 1992):

$$r_c^2 = \frac{\sum_{j=1}^{m} \left(H^F_{n_1 n_2 \dots n_k}(p_{j1}, p_{j2}, \dots p_{jk}) - \hat{p}_z \right)^2}{\sum_{j=1}^{m} \left(p_{jz} - \hat{p}_z \right)^2} \tag{13}$$

where $\hat{p}_z$ is the mean of the values of the attribute X_z. If $r_c^2 = 0$ (resp., $r_c^2 = 1$) means that (11) does not fit (resp., fits perfectly) to the data. However we use a variation of (11) for taking into account both the number of independent variables and the scale of the sample used [9] given by

$$r'^2_c = 1 - \left[\left(1 - r_c^2\right) \cdot \frac{m-1}{m-k-1} \right] \tag{14}$$

.

The pseudocode of the algorithm FAD is schematized below.

Function FAD	
Functional dependency	$X_z = H(X_1,\dots,X_k)$
Input:	DT - Dataset composed by m instances with attributes $X_1, X_2,\dots, X_r$ α – threshold value
Output:	Direct F-transform components $F_{h_1 h_2 \dots h_k}$ Index of determinacy r'^2_c

```
n= 3
r'^2_c = 0
```

3 DO WHILE $r'^2_c \le \alpha$
4 Set F as a matrix of dimension n^k
4 F = {0} //initialize to 0 the components of F
5 FOR each combination $\{h_1,..,h_k\}$
6 $F[h_1,...,h_k]$ = DirectFtransformComponent(DT, n, k,z,h[1,...,k] }) // calculate the F-Transform component $F_{h_1 h_2 \dots h_k}$
7 IF $F[h_1,...,h_k]$ = -1 RETURN F, 0 // the dataset is not sufficiently dense
8 NEXT $\{h_1,...,h_k\}$
9 r'^2_c = IndexofDeterminacy(DT, n, k,z, F) // Calculate the index of determinacy r'^2_c
10 n:=n+1
11 END DO
12 RETURN F, r'^2_c
13 END IF
14 END

The function DirectFuzzyTransform() is used to calculate each direct F-transform component. The function BasicFunction() calculates the value $A_{i\,hi}(x)$ for an assigned x of the h_i*th* basic function of the i*th* fuzzy partition. IndexofDeterminacy calculates the index of determinacy.

Function DirectFTransformComponent	
Description	The component of the direct F-transform $F_{h_1 h_2 \dots h_k}$
Input:	DT - dataset analysed n - number of fuzzy sets of the fuzzy partitions k - number of the input attributes z – index of the of the output attribute X_z h[1,....,k] = array of indices of the combination $\{h_1,..., h_k\}$
Output:	Direct F-transform component $F_{h_1 h_2 \dots h_k}$ 0 if the data points are not sufficiently dense with respect to the fuzzy partition

1 Num = 0, Denum = 0, j
2 FOR each p in DT
3 val = 1
4 xz = p.X[z] // value of the attribute X_z
5 FOR i = 1 to k
6 a = min(DT.X[i]) // inf of $[a_i,b_i]$
7 b = max(DT.X[i]) // sup of $[a_i,b_i]$

```
        j = h[i] // index of the hith fuzzy set of the fuzzy partition of [a,b]
        x = p.X[i] // value of the ith attribute Xi
        A = BasicFunction(a,b,n,x,j)
        val = val*A
     NEXT i
     H= val*F
     Denom = Denom + val
   NEXT p
   IF  Denom = 0  RETURN 0
   ELSE RETURN Num/Denum
   END IF
END
```

Function BasicFunction

Description	An uniform fuzzy partition is created for the interval [a,b]
Input:	a - inf value of the interval [a,b]
	b - sup value of the interval [a,b]
	n – number of fuzzy sets in the fuzzy partition
	x – value of x
	k – index of the kth fuzzy set of the fuzzy partition
Output:	Return the basic function value $A_k(x)$

```
Set h = (b - a)/(n – 1)
Set x[1..n]
x[1] = a
FOR i = 2 to n
   x[i] = x[i-1]+h
 NEXT i
 IF  k = 1 THEN x[k-1] = x[1]
    xmin = x[1]
    xmax = x[k+1]
 ELSE IF  k = n THEN x[k-1] = x[1]
               xmin = x[k-1]
               xmax = x[k]
          ELSE
               xmin = x[k-1]
               xmax = x[k+1]
          ENDIF
  ENDIF
  Set A = 0
  IF  xmin ≤ x ≤ xmax THEN
```

$$A = 0.5 \cdot (1 + \cos\frac{\pi}{h}(x - x[i]))$$

```
 END IF
```

```
RETURN A
END
```

Function IndexofDeterminacy

Description	The index of determinacy r'^2_c is calculated
Input:	DT - dataset analyzed n - number of fuzzy sets of the fuzzy partitions k - number of the input attributes z – index of the of the output attribute X_z F - matrix of the direct F-transform components
Output:	Direct F-transform component $F_{h_1 h_2 \dots h_k}$ 0 if the data points are not sufficiently dense with respect to the fuzzy partition

```
Num = 0, Denum = 0, m = 0
mz= mean(X[z])  // mean value of the attribute Xz in DT
FOR each p in DT
  m = m + 1 //in m calculated the number of instances in DT
  val = 1
  xz = p.X[z] // value of the attribute Xz
  FOR each combination {h1..hk}
    val = 1
    FOR i = 1 to k
       b = max(DT.X[i]) // sup of  [ai,bi]
       j = h[i] // index of the hi-th fuzzy set of the fuzzy partition of [a,b]
       x = p.X[i] // value of the i-th attribute Xi
       A = BasicFunction(a,b,n,x,j)
       val = val*A
    NEXT i
    H = H + val* F[h1..hk]
  NEXT {h1..hk}
  num = num +  (H – mz)^2
  denum = denum +  (xz – mz)^2
 NEXT p
 RETURN (Num/Denum)*(m-1)/(m-k-1)
END
```

3.2 MFAD algorithm

We consider a massive dataset DT composed by r attributes where $X_1,\dots,X_i,\dots,X_r$ and m instances $O_1,\dots,O_j,\dots,O_m$ (m>r). We make a partition of DT in s subsets $DT_l,\ldots, DT_s$ with the same cardinality, by using an uniform random sample in such a way each subset is loadable in memory. We apply the FAD algorithm to each subset, calculating the direct F-transform components, the inverse F-transforms $H^F_{n_1} \ldots H^F_{n_s}$, the indices of determinacy $r'^2_{c1},\ldots, r'^2_{cs}$. r'^2_{cs} and the domains $D_l,\ldots, D_s$, where $D_l = [a_{1l}, b_{1l}]\times\ldots\times[a_{kl}, b_{kl}]$, $l = 1,\ldots,s$. All these quantities are saved in memory. If a dependency f is not found for the l*th* subset, the corresponding value of r'^2_{cl} is set to 0. The pseudocode of MFAD is given below.

Algorithm MFAD	
Functional dependency	$X_z = H(X_1,\ldots,X_k)$
Input:	Massive dataset DT composed by m instances with attributes $X_1, X_2,\ldots, X_r$
Output:	Direct F-transform components found for each subset Domain products $D_l,\ldots, D_s$ Indexes of determinacy $r'^2_{c1},\ldots, r'^2_{cs}$

1	The dataset DT is randomly partitioned in s subsets equally sized $DT_l,\ldots, DT_s$
2	FOR l = 1 to s
3	$r'^2_{cl} = 0$
4	F[l], $r'^2_{cl}[l]$ = FAD(DT_l,α) // the FAD algorithm is called to calculate the direct FTransform components and the index of determinacy for the l-th subset,
5	Save in memory the direct F-transform components, and the domain product D_l
6	NEXT l
7	END

Now we consider a point $(x_1,x_2,\ldots,x_k) \in \bigcup_{l=1}^{s} D_l$. In order to approximate the function $H(x_1,x_2,\ldots,x_k)$, we calculate the weights as:

$$w'_l(x_1, x_2,\ldots, x_k) = \begin{cases} r'^2_{cl} & \text{if } (x_1, x_2,\ldots, x_k) \in D_l \\ 0 & \text{otherwise} \end{cases} \qquad l = 1,\ldots, s \tag{15}$$

If for any subset the functional dependency is not found, then w'_l=0 for each l = 1,…,s. Otherwise, the approximated value of $H(x_1,x_2,\ldots,x_k)$ is given by

$$H^F(x_1, x_2, \ldots, x_k) = \frac{\sum_{l=1}^{s} w_i^{'}(x_1, x_2, \ldots, x_k) \cdot H_{n_l}^F(x_1, x_2, \ldots, x_k)}{\sum_{l=1}^{s} w_i^{'}(x_1, x_2, \ldots, x_k)} \tag{16}$$

which is also the value of X_z. To analyse the performance of the MFAD algorithm we execute a set of experiments on a large dataset formed from 402678 census tracts of the Italian regions provided by the Italian National Statistical Institute (ISTAT) in 2011. Therein 140 numerical attributes belong to each of the following categories:

- inhabitants,
- foreigner and stateless inhabitants,
- families,
- buildings,
- dwellings.

The FAD method is applied on the overall dataset, the MFAD method is applied by partitioning the dataset in s subsets and we perform the tests varying the value of the parameter s and by setting the threshold α = 0.7.
In addition, we compare the MFAD algorithm with the Support Vector Regression (SVR) and Multilayer Perceptron (MP) algorithms.

4. Experiments

Table 1 shows the 402678 census tracts of Italy divided for each region.

Table 1. Number of census tracts for each Italian region

ID region	Description	Number of census tracts
001	Piemonte	35672
002	Valle d'Aosta	1902
003	Lombardia	53173
004	Trentino Alto Adige	11712
005	Veneto	33883
006	Friuli Venezia Giulia	8278
007	Liguria	11054
008	Emilia Romagna	38603
009	Toscana	28917
010	Umbria	7480

011	Marche	11862
012	Lazio	32065
013	Abruzzo	9529
014	Molise	2821
015	Campania	24323
016	Puglia	22514
017	Basilicata	5107
018	Calabria	13121
019	Sicilia	36681
020	Sardegna	13981

Table 2 shows the approximate number of census tracts in each subset for each partition of the dataset in s subsets.

Table 2. Number of census tracts for each subset by varying s

s	Number of census tracts
8	$5.0 \cdot 10^4$
9	$4.5 \cdot 10^4$
10	$4.0 \cdot 10^4$
11	$3.7 \cdot 10^4$
13	$3.1 \cdot 10^4$
16	$2.5 \cdot 10^4$
20	$2.0 \cdot 10^4$
26	$1.5 \cdot 10^4$
40	$1.0 \cdot 10^4$

In any experiment we apply the MFAD algorithm to analyze the attribute dependency explored of an output attribute X_z from a set of input attributes $X_1, X_2, \ldots, X_r$. In all the experiments we set $\alpha = 0.7$ and partition randomly the dataset in s subsets. We now show the results obtained in three experiments.

Experiment A

In this experiment we explore the relation between the density of resident population with laurea degree and the density of resident population employed. Generally speaking, a higher density of population with laurea degree should correspond to a greater density of population employed. The attribute dependency explored is $H_z = H(X_1)$, where

Input attribute: X_1 = Resident population with laurea degree

Output attribute: X_z = Resident population over 15 employed

We apply the FAD algorithm on different random subsets of the dataset and then we calculate the index of determinacy (12). In Table 3 we show the value of the index of determinacy r'^2_{cl} obtained for different values of s. For s = 1, we have the overall dataset.

Table 3. Index of determinacy for values of s in experiment A via FAD

s	Index of determinacy
1	0.760
8	0.745
9	0.748
10	0.750
11	0.752
13	0.754
16	0.758
20	0.752
26	0.748
40	0.744

The results in Table 3 show that the dependency has been found. We obtain r'^2_{cl} = 0.760 by using FAD algorithm on the entire dataset, while the best value of r'^2_{cl} (reached by using MFAD) is 0.758 for s = 16. Hence the related smallest difference between the two algorithms is 0.02. Fig. 4 shows in abscissas the input X_1 and in ordinates the output $H^F(x_1)$ for s = 1, 10, 16, 40.

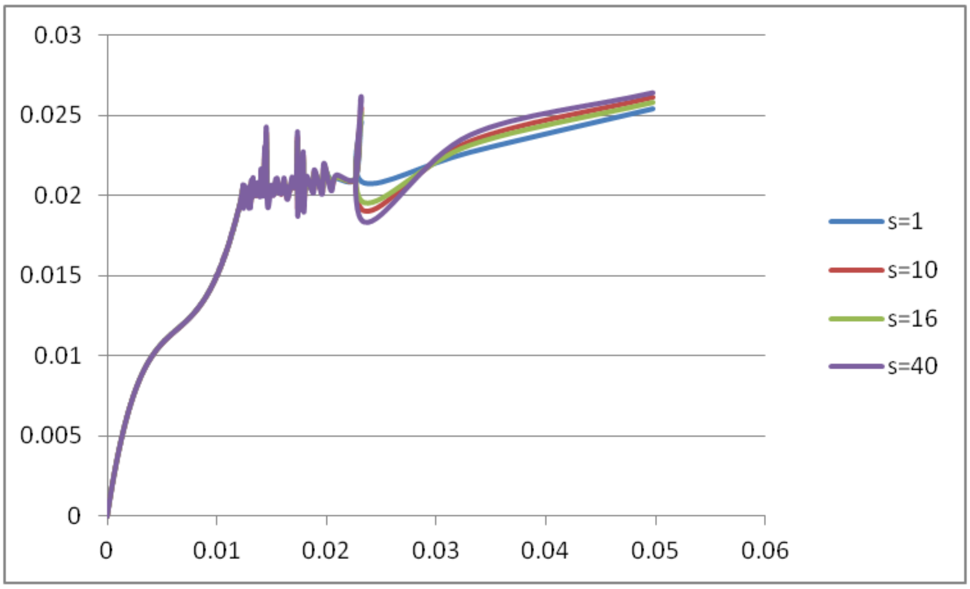

Fig. 4. Tendency of H_z for dataset partitions in the experiment A

Experiment B

In this experiment we explore the relation between the density of residents with job or capital income and the density of families in owned residences. We expect that the greater the density of residents with job or capital income is, the resident families density in owned homes the greater is. The attribute dependency explored is $H_z = H(X_1)$, where:

Input attributes: X_1 = Resident population with job or capital income

Output attribute X_z = Families in owned residences

After some tests, we put $\alpha = 0.8$.

Table 4 shows r'^2_{cl} obtained for different values of s: r'^2_{cl} = 0.881 in FAD algorithm on the entire dataset, r'^2_{cl} = 0.878 in MFAD obtained for s = 13, 16. The smallest index of dependency difference is 0.003.

Table 4. Index of determinacy for values of s in experiment B via FAD

s	Index of determinacy
1	0.881
8	0.872
9	0.872
10	0.874
11	0.875
13	0.877
16	0.878
20	0.878
26	0.875
40	0.872

Fig. 5 shows in abscissas the input X_1 and in ordinates the output $H^F(x_1)$ for s = 1, 10, 16, 40.

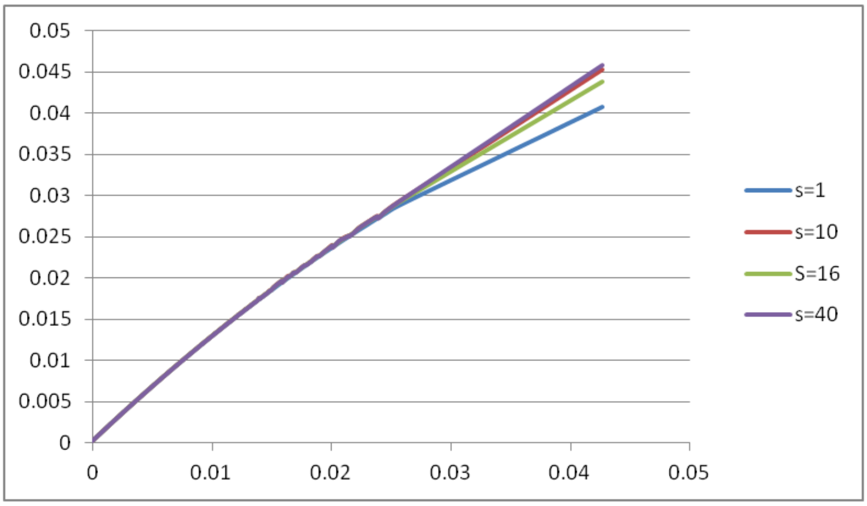

Fig. 5. Trend of H_z for dataset partitions in the experiment B

Experiment C

In this experiment the attribute dependency explored is $H_z = H(X_1,X_2)$, where

Input attributes:

X_1 = Density of residential buildings built with reinforced concrete

X_2 = Density of residential buildings built after 2005

Output attribute:

X_z = Density of residential buildings with state of good conservation

After some tests, we decided α = 0.75 in this experiment. In Table 5 we show r'^2_{cl} obtained for different values of s: r'^2_{cl} = 0.785 in FAD algorithm on the entire dataset. r'^2_{cl} = 0.781 in MFAD algorithm obtained for s = 13, 16. The smallest index of dependency difference is 0.004.

Table 5. Index of determinacy for values of s in the experiment C via FAD

s	Index of determinacy
1	0.785
8	0.776
9	0.776
10	0.778
11	0.780
13	0.781
16	0.781
20	0.780
26	0.779
40	0.777

Now we present the results obtained by considering all the experiments performed on the entire dataset in which the dependency was found (r'^2_{cl} > 0.7). We consider the index of determinacy in the FAD algorithm (s=1) and the minimum and maximum values of the index of determinacy obtained by using the MFAD algorithm for s = 9,10,11,13,16,20,26,40.

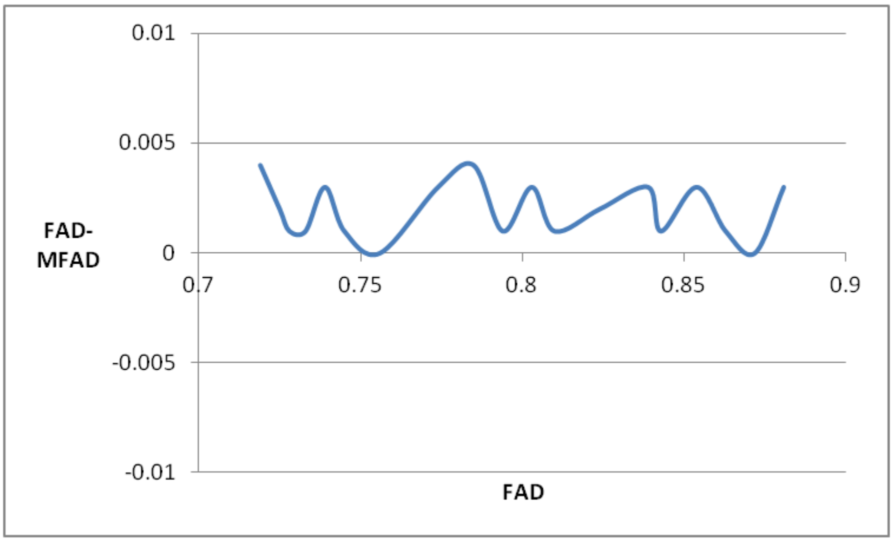

Fig. 6. Trend of the difference between the max value r'^2_{cl} in MFAD and FAD

A functional dependency was found in 43 experiments. Fig. 6 (resp., 7) shows the trend of the difference between the maximum (resp., minimum) value calculated for r'^2_{cl} in MFAD and in FAD for the same experiment. In abscissae we have r'^2_{cl} in the FAD method, in ordinates the difference between the two indices. For all the experiments this difference is always below 0.005 (resp., 0.0015).

These results show that the MFAD algorithm is comparable with the FAD algorithm, independently of the choice of the number of subsets partitioning the entire dataset.

Fig. 8 show the mean CPU time gain obtained by MFAD algorithm with different partitions, with respect to the CPU time obtained by using FAD algorithm (s = 1). The CPU time gain is given by the difference between the CPU time measured by using s = 1, and the CPU time measured by using a partition in s subsets, divided by the CPU time measured for s = 1. The CPU time gain is always positive and the greatest value are obtained for s = 16. These considerations allow to apply the MFAD algorithm to a VL dataset not loadable entirely in memory to which the FAD algorithm is not applicable.

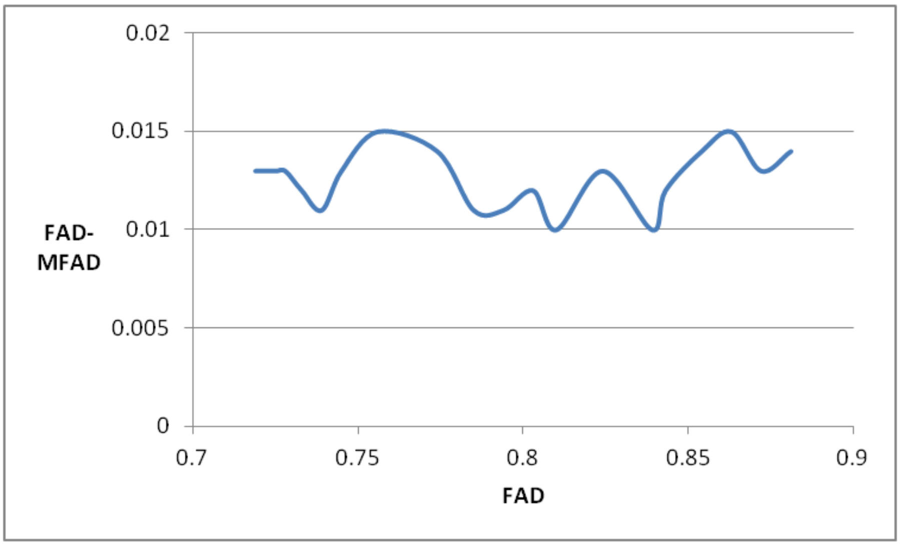

Fig. 7. Trend of the difference between the min value of r'^2_{cl} in MFAD and FAD

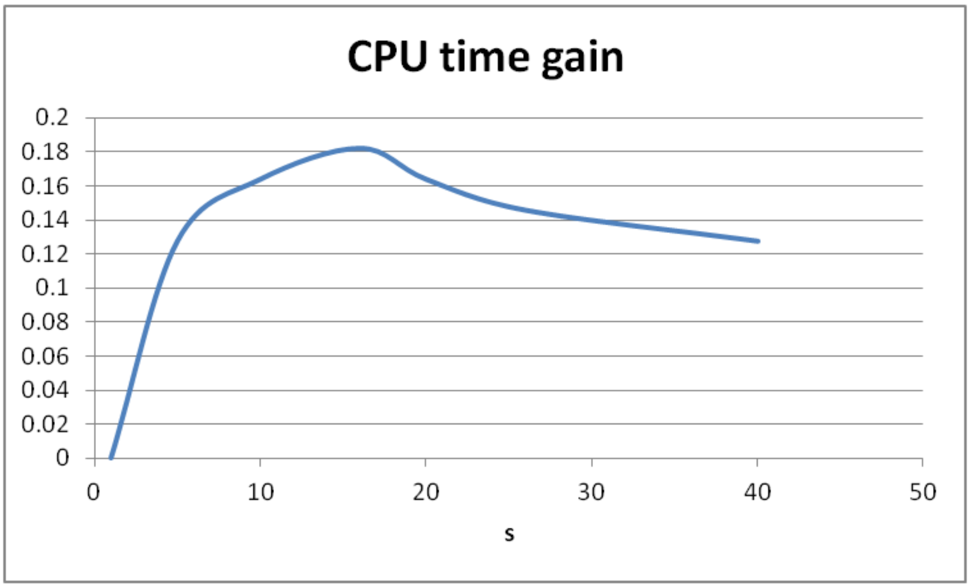

Fig. 8. Trend of CPU time gain with respect to FAD method (s = 1)

Now we compare the results obtained by using the MFAD method with the ones obtained by applying the SVR and MLP algorithms. For the comparison tests we have used the machine learning tool Weka 3.8.

In order to perform the tests by using the SVR algorithm we repeat each experiment using the following different kernel functions: linear, polynomial, Pearson VII universal kernel, and Radial Basis Function kernel, and varying the complexity C parameter in a range between 0 and 10. To compare the performances of the SVR and MFAD algorithms we measure the index of determinacy and store it in every experiment.

In Fig. 9 we show the trend of the difference between the max values of r'^2_{cl} in SVR and MFAD.

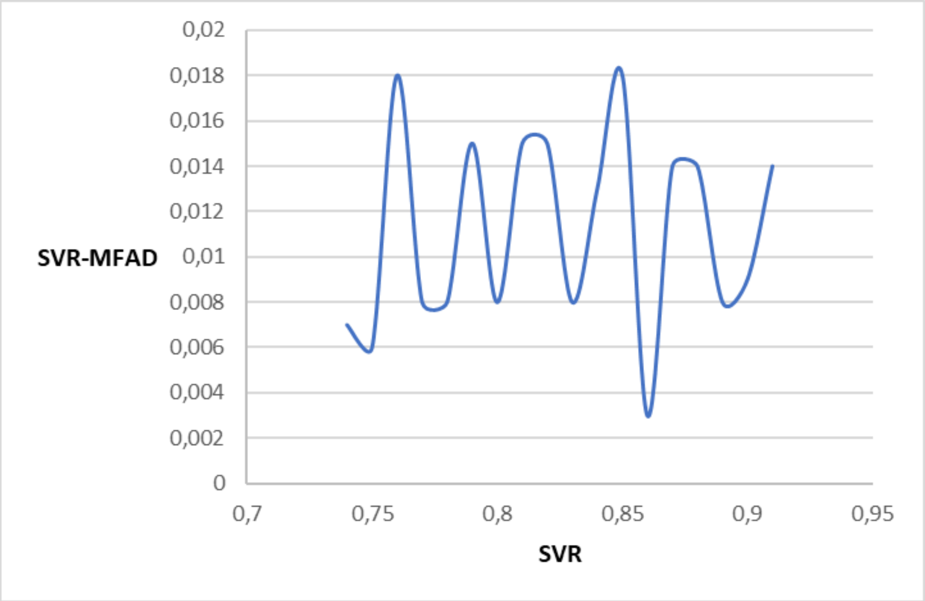

Fig. 9. Trend of the difference between the max value of r'^2_{cl} obtained in SVR and MFAD

Fig. 9 shows that the difference between the optimal value r'^2_{cl} in SVR and MFAD is always under 0.02. In the comparison tests performed by using the MP algorithm, we vary the learning rate and the momentum parameter in [0.1,1]. We use a single hidden layer varying the number of nodes between 2 and 8. Furthermore, we set the number of epochs to 500 and the percentage size of validation set to 0.

In Fig. 10 we show the trend of the difference between the max value of r'^2_{cl} in MP and MFAD.

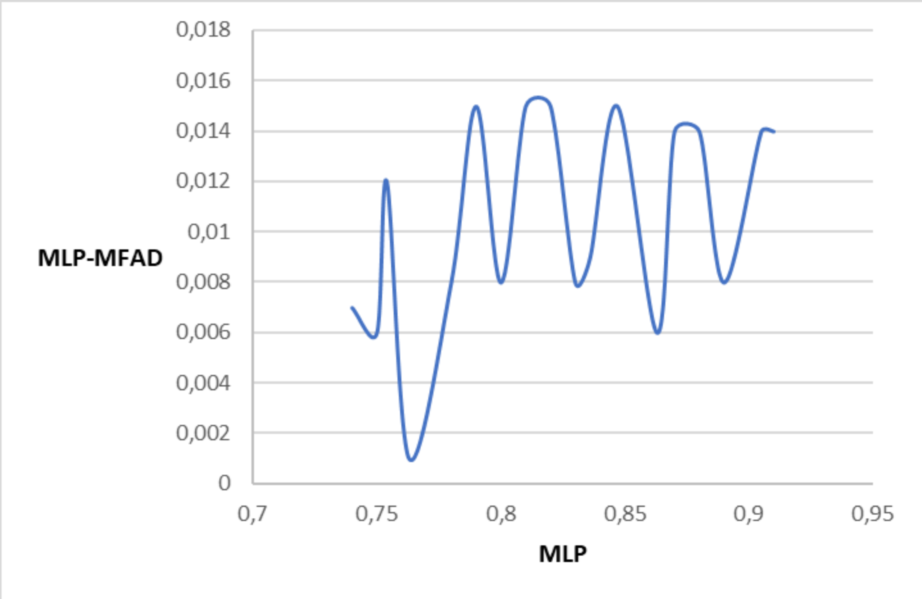

Fig. 10. Trend of the difference between the max value of r'^2_{cl} in MLP and MFAD

Fig. 10 shows that the difference between the max value of the index of determinacy in MLP and MFAD is under the value 0.016.

These results show that the MFAD algorithm of attribute dependency in massive datasets has comparable performances with the SVR and MLP nonlinear regression algorithms. Moreover, it has the advantage of having a smaller number of parameters compared to the other two algorithms, therefore it has greater usability and can be easily integrated into expert systems and intelligent systems for the analysis of dependencies between attributes in massive datasets. Indeed, the only two parameters for the execution of the MFAD algorithm are the number of subsets and the threshold value of the index of determinacy.

6. Conclusions

The FAD method presented in [9] can be used as a regression model for finding attribute dependencies in datasets: the inverse multiple F-transform can approximate the regression function. But this method can be expensive for massive datasets and for VL datasets not loaded in memory. Then we propose a variation of the FAD method for massive datasets called MFAD: the dataset is partitioned in s subsets equally sized, to each subset the FAD method is applied by calculating the inverse F-transform. approximated by a weighted mean where the weights are given from the index of determinacy assigned to each subset. For testing the performance of the MFAD method, we compare tests with respect to the FAD method on an L dataset of the ISTAT 2011 census data. The results show that the performances obtained in MFAD are well comparable in FAD. The comparison tests show that the MFAD algorithm has performances comparable with SVR and MLP algorithms, moreover it has greater usability due to the lower number of parameters to be selected.

These results allow us to conclude that MFAD provides acceptable performance in the detection of attribute dependencies in the presence of massive datasets. Therefore, unlike FAD, MFAD can be applied to massive data and can represent a trade-off between usability and high performance in detecting attribute dependencies in massive datasets.

The critical point of the algorithm is the choice of the number of subsets and the threshold value of the index of determinacy. Further studies on massive datasets are necessary to analyze if the choice of the optimal values of these two parameters depend on the type of dataset analyzed. Furthermore, we intend to experiment the MFAD algorithm in future robust frameworks such as expert systems and decision support systems.

Acknowledgements. This paper was not supported from research funds.

References

[1] Anguita A., Ridella S., Rivieccio F. (2005). K–Fold Generalization Capability Assessment for Support Vector Classifiers, *Proceedings of the IEEE Int. Joint Conf. on Neural Networks, IJCNN 2005*, pp. 855–858. DOI: 10.1109/IJCNN.2005.1555964.

[2] Chen C., Li K., Duan M., Li K. (2017). Chapter 6 - Extreme Learning Machine and Its Applications in Big Data Processing, in *Big Data Analytics for Sensor-Network Collected Intelligence*, Intelligent Data-Centric Systems, pp. 117-150. https://doi.org/10.1016/B9780128093931.000064.

[3] Chen, C.L.P., Zhang, C.Y. (2014). Data-intensive applications, challenges, techniques and technologies: a survey on Big Data. *Information Sciences*, *275*, 314–347. https://doi.org/10.1016/j.ins.2014.01.015.

[4] Cheng C. H., Tan P., Jin R. (2010). Efficient Algorithm for Localized Support Vector Machine, *IEEE Transactions on Knowledge and Data Engineering*, 22 (4), 537-549. https://doi.org/10.1109/TKDE.2009.116.

[5] R. Collobert and S. Bengio (2004). Links between perceptrons, MLPs and SVMs. *ICML '04: Proceedings of the Twenty-First International Conference on Machine Learning*. https://doi.org/10.1145/1015330.1015415

[6] Cybenko G. (1989) Approximation by superpositions of a sigmoidal function. *Math. Control Signal Systems* 2**,** 303–314. https://doi.org/10.1007/BF02551274

[7] Dean J. (2014). *Big Data, data mining, and machine learning: value creation for business leaders and practitioners*. New York: Wiley & Sons Inc. ISBN:15024629159781502462916.

[8] Di Martino F., Loia V., Perfilieva I., Sessa S. (2008). An image coding/decoding method based on direct and inverse fuzzy transforms. *International Journal of Approximate Reasoning*, *48*, 110–131. https://doi.org/10.1016/j.ijar.2007.06.008.

[9] Di Martino F., Loia V., Sessa S. (2010a). Fuzzy transforms method and attribute dependency in data analysis. *Information Sciences*, *180*, 493–505. https://doi.org/10.1016/j.ins.2009.10.012.

[10] Di Martino F., Loia V., Sessa S. (2010b). Fuzzy transforms for compression and decompression of color videos. *Information Sciences*, *180*, 3914–3931. https://doi.org/10.1016/j.ins.2010.06.030.

[11] Di Martino F., Loia V., Sessa S. (2011a). Fuzzy transforms method in prediction data analysis. *Fuzzy Sets and Systems*, *180*, 146–163. https://doi.org/10.1016/j.fss.2010.11.009.

[12] Di Martino F., Loia V., Sessa S. (2011b). A segmentation method for images compressed by fuzzy transforms. *Fuzzy Sets and Systems*, *161*, 56–74. https://doi.org/10.1016/j.fss.2009.08.002.

[13] Di Martino F., Sessa S. (2007). Compression and decompression of image with discrete fuzzy transforms. *Information Sciences*, *177*, 2349–2362. https://doi.org/10.1016/j.ins.2006.12.027.

[14] Di Martino F., Sessa S. (2012). Fragile watermarking tamper detection with images compressed by fuzzy transform. *Information Sciences*, *195*, 62–90. https://doi.org/10.1016/j.ins.2012.01.014.

[15] Draper N.R., Smith H. (1988). *Applied regression analysis*. New York: Wiley & Sons Inc. ISBN: 9780471170822.

[16] Drucker H., Burges C. J. C., Kaufman L., Smola A. J, Vapnik V. (1996). Support vector regression machines, *NIPS'96 Proceedings of the 9th International Conference on Neural Information Processing Systems* 1996, pp. 155–161. MIT Press.

[17] Han H., Jian X. (2019). Overcome Support Vector Machine Diagnosis Overfitting, *Cancer Informatics*, 13(1), 145–158. https://doi.org/10.4137/CIN.S13875

[18] Han, M., Kamber, M., Pei, J. (2012). *Data mining: concepts and techniques*. (3rd ed.). Morgan Kaufmann (Elsevier). ISBN: 9780123814791.

[19] Hastie T., Tibshirani R., Friedman J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York. https://doi.org/10.1007/9780387848587

[20] Haykin S. (1999). *Neural Networks: A Comprehensive Foundation* (2nd ed.). Prentice Hall. ISBN: 0132733501.

[21] Haykin S. (2009). *Neural Networks and Learning Machines* (3rd ed.) Prentice Hall. ISBN:100131471392.

[22] Johnson, R. A., Wichern, D. W. (1992). *Applied Multivariate Statistical Analysis*. London: Prentice-Hall International. ISBN: 9780131877153.

[23] Jun, S.,. Lee, S. J., & Ryu, Y. B. (2015). A divided regression analysis for big data. *International Journal of Software Engineering and Its Applications*, 9, (5), 21–32. https://doi.org/10.14257/ijseia.2015.9.5.03.

[24] Lee, Y. S., Yen, S. J. (2004). Classification based on attribute dependency. Proceedings of 6th International Conference DaWaK’ 04. *Lecture Notes in Computer Sciences*, 5192, 259–268. ISBN: 9783540876045.

[25] Leskovec, J., Rajaraman, A., Ullmann, J. D. (2014). *Mining of Massive Datasets*. Cambridge University Press. (2nd ed.). ISBN: 9781107077232.

[26] Mitra S., Pal S. K., Mitra P. (2002). Data mining in soft computing framework: a survey. *IEEE Transactions on Neural Networks*, 13 (1), 3–14. https://doi.org/10.1109/72.977258.

[27] Murtagh F. (1991). Multilayer perceptrons for classification and regression, *Neurocomputing*, 2 (5-6), 183-197. https://doi.org/10.1016/0925-2312(91)900235.

[28] Peng H., Choi D., Liang C. (2013). Evaluating parallel logistic regression models, *2013 IEEE International Conference on Big Data*, Silicon Valley, CA, USA, 6-9/10/2013. https://doi /10.1109/BigData.2013.6691743.

[29] Perfilieva, I. (2006). Fuzzy transforms: theory and applications. *Fuzzy Sets and Systems*, *157*, 993–1023. https://doi.org/10.1016/j.fss.2005.11.012.

[30] Perfilieva, I., Novàk, V., Dvoràk, A. (2008). Fuzzy transforms in the analysis of data. *International Journal of Approximate Reasoning* , 48, 36–46. https://doi.org/10.1016/j.ijar.2007.06.003.

[31] Piatecky-Shapiro, G., Frawley, W. J. (1991). *Knowledge discovery in databases*. Cambridge (MA), MIT Press. ISBN: 9780262660709.

[32] Raju K. S., Murti M. R., Rao M. V., Satapathy S. C. (2018). Support Vector Machine with K-fold Cross Validation Model for Software Fault Prediction, *International Journal of Pure and Applied Mathematics*, 118 (20), 331-334. ISSN: 1314-3395.

[33] Schmidhube J. (2014). Deep learning in neural networks: an overview. *Neural Networks*. 61: 85–117. https://doi.org/10.1016/j.neunet.2014.09.003.

[34] Segata N., Blanzieri E. (2009.) Fast Local Support Vector Machines for Large Datasets, in: Perner P. (ed.) Machine Learning and Data Mining in Pattern Recognition. *Lecture Notes in Computer Science*, vol. 5632. Springer, Berlin, Heidelberg, pp. 295-310. https://doi.org/10.1007/9783642030703_22.

[35] Singh, S., Firdaus, T., Sharma, A. K. (2015). Survey on Big Data using data mining. *International Journal of Engineering Development and Research*, 3 (4), 135–143. ISSN: 23219939.

[36] Tanaka, H. (1987). Fuzzy data analysis by possibilistic linear models. *Fuzzy Sets and Systems*, 24, 363–375. https://doi.org/10.1016/01650114(87)900339.

[37] Thomas P., Suhner M. C. (2015). A New Multilayer Perceptron Pruning Algorithm for Classification and Regression Application, *Neural Processing Letters*, 42 (2), 437-458. https://doi.org/10.1007/s1106301493665.

[38] Vucetic M., Hudec M., Vujošević M. (2013). A new method for computing fuzzy functional dependencies in relational database systems. *Expert Systems with Applications,* 40 (7), 2738–2745. https://doi.org/10.1016/j.eswa.2012.11.019.

[39] Wood S. N., Goude Y., Shaw S. (2015). Generalized additive models for large data sets. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 64 (1), 139–155. https://doi.org/10.1111/rssc.12068.

[40] Wu X., Zhu X., Wu G. Q., Ding W. (2014). Data mining with Big Data. *IEEE Transactions on Knowledge and Data Engineering,* 26 (1), 97–107. https://doi.org/10.1109/TKDE.2013.109.

[41] Yao L., Ge Z. (2019). Distributed parallel deep learning of Hierarchical Extreme Learning Machine for multimode quality prediction with big process data, *Engineering Applications of Artificial Intelligence*, 81, 450-465. https://doi.org/10.1016/j.engappai.2019.03.011.

[42] Yen S. J., Lee Y. S. (2011). A neural network approach to discover attribute dependency for improving the performance of classification. *Expert Systems with Applications*, 38 (10), 12328–12338. https://doi.org/10.1016/j.eswa.2011.04.011.

[43] Zheng J., Shen F., Fan H., Zhao J. (2013). An online incremental learning support vector machine for large-scale data. *Neural Computing Applications*, 22 (5), 1023–1035. https://doi.org/10.1007/s0052101107931.