Article

A Novel Classification Algorithm Based on Multidimensional F^1 Fuzzy Transform and PCA Feature Extraction

Barbara Cardone [1] and Ferdinando Di Martino [1,2,*]

[1] Dipartimento di Architettura, Università degli Studi di Napoli Federico II, Via Toledo 402, 80134 Napoli, Italy
[2] Centro Interdipartimentale di Ricerca A. Calza Bini, Università degli Studi di Napoli Federico II, Via Toledo 402, 80134 Napoli, Italy
* Correspondence: fdimarti@unina.it; Tel.: +39-081-253-8907; Fax: +39-081-253-8905

Abstract: The bi-dimensional F^1-Transform was applied in image analysis to improve the performances of the F-transform method; however, due to its high computational complexity, the multidimensional F^1-transform cannot be used in data analysis problems, especially in the presence of a large number of features. In this research, we proposed a new classification method based on the multidimensional F^1-Transform in which the Principal Component Analysis technique is applied to reduce the dataset size. We test our method on various well-known classification datasets, showing that it improves the performances of the F-transform classification method and of other well-known classification algorithms; furthermore, the execution times of the F^1-Transform classification method is similar to the ones obtained executing F-transform and other classification algorithms.

Keywords: data classification; fuzzy transform; F^1-transform; PCA

check for updates

Citation: Cardone, B.; Martino, F.D. A Novel Classification Algorithm Based on Multidimensional F^1 Fuzzy Transform and PCA Feature Extraction. *Algorithms* **2023**, *16*, 128. https://doi.org/10.3390/a16030128

Academic Editor: Friedhelm Schwenker

Received: 26 January 2023
Revised: 21 February 2023
Accepted: 22 February 2023
Published: 23 February 2023

1. Introduction

The Fuzzy Transform technique (for short F-transform) [1] is a fuzzy regression method introduced to approximate a continuous function of k variables $f(x_1,x_2, \ldots , x_k)$ defined in the domain $[a_1,b_1] \times [a_2,b_2] \times \ldots \times [a_k,b_k] \subset R^k$. If this function is known in N points $p_j = (p_{j1}, p_{j2}, \ldots ,p_{jk})$ j = 1, … N, it can be approximated by a weighted average whose weights are constants given by the components of the direct F-transform.

F-transform was applied in many image and data analysis problems. An extensive description of the F-transform-based techniques used in image and data analysis is given in [2].

In [3] a generalization of the F-transform, called high degree F-transform and labelled with F^s-transform ($s \geq 0$) was proposed, where the F-transform is given by the zero-degree F^0-transform. In the F^s-transform, with $s > 0$, the constant components of the direct F-transform, were replaced by s-degree polynomial components with the aim to capture more information about the original function. The greater the polynomial degree, the finer the approximation of the original function; however, as the polynomial degree increases, the computational complexity of the algorithm that uses the F^S transform increases considerably. In addition, the meeting of the constraint of sufficient data density with respect to fuzzy partitions further requires additional consumption of memory resources and CPU time [2,4].

Some researchers apply the F^1-transform in image analysis, in which only two input variables are used, and the sufficient data density constraint is always met. In [5,6] an algorithm based on the F^1-transform in two variables was applied in an image processing edge detection problem. In [7] a lossy image compression method based on the F^1-transform in two variables is proposed; the authors show that this method improves the decoded image quality obtained using the F-transform image compression [8]. In [9] a hybrid deep neural network in which the F^s-transform is used in image analysis to construct in a preprocessing phase the convolution kernels in the first two layers of the network.

Generally, when the multidimensional F-transform is applied in classification, regression, and prediction problems, as the dimensionality of the data increases, the computational complexity greatly increases; for this reason, the application of high-order F-transforms in data regression or classification has a high computational complexity, in terms of memory and time consumption, especially in the presence of massive multidimensional data [4]. An application of the first-order unidimensional fuzzy transform was proposed in [10] where the unidimensional F^1-transform is applied to seasonal time series weather datasets in a seasonal forecasting model. Comparison tests show that this model improves the forecasting time series performances obtained by applying the F^0-transform model proposed in [11,12]. In [13] the F^1-transform is applied to remove seasonal components and noise in time series.

Recently, in [14] a new classification algorithm based on the multidimensional F^0-transform for massive datasets called Multi-dimensional F-transform Classification (for short, MFC), was proposed. In MFC the K-fold cross-validation technique is applied to avoid overfitting in the data; the multidimensional direct F^0-transform is applied to each fold and a weighted mean of the multi-dimensional inverse F^0-transform calculated from the direct F^0-transform components obtained in each fold is used to classify data points. Comparisons with well-known classification algorithms showed that MFC has better classification performances than Naive Bayes [15] and Lazy Bk [16] and is comparable with respect to the ones obtained by using Decision tree J48 [17] and Multilayer Perceptron [18] algorithms. On the other hand, the main drawback of this method is the high computational complexity that is reached when the number of features increases.

In [19] a hybrid fast classification method based on F-transform and Principal Component Analysis (for short, PCA) [20–23] is proposed. This method is tested to classify images; the results show that it improves the success rate and computation time obtained by applying the F-transform algorithm.

PCA is a well-known dimensionality reduction multivariate statistics technique whose goal is to reduce the number of features of a dataset by losing the least amount of information possible. PCA is one of the most used feature extraction techniques in data analysis. Its strong point is to be able to reduce the dimensionality of the data, while preserving their information content.

In this research, we propose a hybrid classification method applied to massive data including the PCA and the multidimensional F^1-transform techniques. The main goals of the proposed method are:

- Improve the MFC classification performances; the application of the multidimensional direct and inverse F1-transform allows to increase the accuracy and precision of the classifier with respect to the use of the multidimensional F-transform.
- Significantly reduce the time and memory consumption executing the PCA feature extraction algorithm in the preprocessing phase to reduce the dimensionality of the data.

In Section 2, the concepts of multidimensional direct and invers F^1-transform are introduced and the PCA algorithm is synthetized. Section 3 is focused on the analysis of the architecture and functional characteristics of our classification method based on the multidimensional F1-transform. The results of experimental tests are discussed in Section 4. Finally, in Section 5 the conclusions and future perspectives are included.

2. Preliminaries

2.1. Principal Component Analysis

Data dimensionality reduction techniques are divided into feature selection and feature extraction techniques. Feature selection techniques, such as random forest or grid search algorithms, select a subset of the original features in order to reduce the complexity and computational efficiency of the model. Conversely, feature extraction techniques extract information from the original features set and create a new features subspace. While feature selection techniques aim to select the most significant features, discarding the less significant ones from the set of original features, feature extraction techniques construct a new reduced set of features, starting from the existing ones, able to synthesize most of the information contained in the original set of features.

The use of feature selection techniques is preferable when the explainability of the model and the semantic meaning of the features are required; feature extraction techniques are used to reduce the model complexity, improving its predictive performance.

PCA is one of the most used feature extraction techniques in data analysis. Its strong point is to be able to reduce the dimensionality of the data, while preserving their information content.

Let D be the original dataset with s features $X_1, \ldots, X_s$ and N instances. The i*th* instance is characterized by a vector $(x_{i1}, x_{i2}, \ldots, x_{is})^T$ where x_{ij} is the value of the i*th* instance in correspondence to the j*th* feature.

Let m_j be the mean value of the j*th* feature, given by:

$$m_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij} \qquad j = 1, \ldots, s \tag{1}$$

and st_j the standard deviation is given by:

$$st_j = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(z_{ij} - m_j\right)^2} \qquad j = 1, \ldots, s \tag{2}$$

To remember this definition, we can break it down into eight steps:

1. The aim of this phase is to standardize the range of the initial variables so that each one of them contributes equally to the analysis. For each instance $x_{ij,}$ its normalized value is computed, given by:

$$z_{ij} = \frac{x_{ij} - m_j}{s_j}\ i = 1, \ldots, N \qquad j = 1, \ldots, s \tag{3}$$

2. The relationships among features are analyzed by computing the symmetric covariance matrix C = Z^TZ, where Z^T is the transpose of the normalized matrix Z. The components of C are given by

$$C_{jk} = \frac{1}{N-1}\sum_{i=1}^{N} z_{ji} \cdot z_{ik} \qquad j,\ k = 1, \ldots, s \tag{4}$$

3. In this phase, the s eigenvalues and the s eigenvectors of the covariance matrix are extracted. The eigendecomposition of C is where we decompose C into VDV^{-1}, where V is the matrix of eigenvectors and D is a diagonal matrix in which the diagonal components are the eigenvalues λ_i i =1, ... ,s and the other elements are equal to 0.
4. The eigenvalues on the diagonal of D will be associated with the corresponding column in P—that is, the first element of D is λ_1 and the corresponding eigenvector is the first column of P. This holds for all elements in D and their corresponding eigenvectors in P. We will always be able to calculate PDP^{-1} in this fashion.
5. In this phase the s eigenvalues are sorted in descending order; in the same way the corresponding eigenvectors in the matrix V are ordered, obtaining the matrix V', whose columns correspond to the ordered eigenvectors.
6. The normalized data matrix Z is transformed in the matrix of the principal components Z' multiplying Z by the ordered matrix of the eigenvectors V': $Z' = ZV'$.
7. The significant principal components are selected by analyzing the eigenvalues, sorted in descending order. Three heuristic criteria are generally used for the choice of the number of components:
   - Select only the main components corresponding to the eigenvalues whose sum, compared to the sum of all the eigenvalues, is greater than or equal to a specific threshold, for example, 80% or 90%.

- Adopt the Kaiser criterion, in which only those components are selected which correspond to an eigenvalue greater than or equal to 1, or, equivalently, the components that have variance greater than the average.
- Build the eigenvalue graph, called *Scree Plot*, and select the number of components corresponding to the *elbow point* beyond which the graph the eigenvalues stabilizes.

8. The reduced dataset is constructed considering only the significant principal components.

2.2. Multidimensional F-Transform

Let f: $X \subseteq R^n \rightarrow Y \subseteq R$ be a continuous n-dimensional function defined in a closed interval $X = [a_1,b_1] \times [a_2,b_2] \times \ldots \times [a_n,b_n] \subseteq R^n$ and known in a discrete set of N points $P = \{(p_{11}, p_{12}, \ldots, p_{1n}), (p_{21}, p_{22}, \ldots, p_{2n}), \ldots, (p_{N1}, p_{N2}, \ldots, p_{Nn})\}$.

For each $i = 1, \ldots, n$ let $x_{i1}, x_{i2}, \ldots, x_{imi}$ with $m_i \geq 2$ be a set of m_i points of $[a_i,b_i]$, called nodes, such that $x_{i1} = a_i < x_{i2} < \ldots < x_{imi} = b_i$.

For each $i = 1, \ldots, n$ let $A_{i1}, A_{i2}, \ldots, A_{imi}$: $[a_i, b_i] \rightarrow [0,1]$ be a family of fuzzy sets forming a fuzzy partition of $[a_i,b_i]$, where:

1. $A_{ih}(x_{ih}) = 1$ for every $h = 1,2, \ldots, m_i$;
2. $A_{ih}(x) = 0$ if x is not in (x_{ih-1}, x_{ih+1}), where we assume $x_{i0} = x_{i1} = a_i$ and $x_{in+1} = x_{in} = b_i$ by commodity of presentation;
3. $A_{ih}(x)$ strictly increases on $[x_{ih-1}, x_{ih}]$ for $h = 2, \ldots, m_i$ and strictly decreases on $[x_{ih}, x_{ih+1}]$ for $h = 1, \ldots, m_i - 1$;
4. $\sum_{h=1}^{m_i} A_{ih}(x) = 1$ for every $x \in [a_i, b_i]$.

The fuzzy sets $A_{i1}, A_{i2}, \ldots, A_{imi}$ are called basic functions.

Let $c_i = (b_i - a_i)/(m_i - 1)$. The basic functions $A_{i1}, A_{i2}, \ldots, A_{imi}$ form a uniform fuzzy partition of $[a_i,b_i]$ if:

5. $m_i \geq 3$ and the nodes are equidistant, i.e., $x_{ih} = a_i + d_i \cdot (h - 1)$, where $d_i = (b_i - a_i)/(m_i - 1)$ and $h = 1, 2, \ldots, m_i$.
6. $A_{ih}(x_{ih} - x) = A_{ih}(x_{ih} + x)$ $\forall x \in [0,h]$ and $\forall h = 2, \ldots, m_i - 1$;
7. $A_{ih} + 1(x) = A_{ih}(x - d_i)$ $\forall x \in [x_{ih}, x_{ih} + 1]$ and $\forall h = 1,2, \ldots, m_i - 1$.

We say that the set $P = \{(p_{11}, p_{12}, \ldots, p_{1n}), (p_{21}, p_{22}, \ldots, p_{2n}), \ldots, (p_{N1}, p_{N2}, \ldots, p_{Nn})\}$ is sufficiently dense w.r.t. the set of fuzzy partitions $\{A_{11}A_{12}\ldots A_{1m_1}\}, \ldots, \{A_{i1}A_{i2}\ldots A_{im_i}\}, \ldots, \{A_{n1}A_{n2}\ldots A_{nm_n}\}$ if for each combination $A_{1h_1}A_{2h_2}\ldots A_{nh_n}$ exists at least a point $p_j = (p_{j1}, p_{j2}, \cdots, p_{jn}) \in P$, such that $A_{1h_1}(p_{j1}) \cdot A_{2h_2}(p_{j2}) \cdot \ldots \cdot A_{nh_n}(p_{jn}) > 0$. In this case, we can define the direct multidimensional F-transform of f with the $(h_1,h_2, \ldots, h_s)$*th* component $F_{h_1h_2\ldots h_n}$ given by

$$F_{h_1h_2\ldots h_n} = \frac{\sum_{j=1}^{N} f(p_{j1}, p_{j2}, \ldots p_{jn}) \cdot A_{1h_1}(p_{j1}) \cdot A_{2h_2}(p_{j2}) \cdot \ldots \cdot A_{nh_n}(p_{jn})}{\sum_{j=1}^{N} A_{1h_1}(p_{j1}) \cdot A_{2h_2}(p_{j2}) \cdot \ldots \cdot A_{nh_n}(p_{jn})} \tag{5}$$

The multidimensional inverse F-transform, calculated in the point p_j, is given by:

$$f^F_{n_1n_2\ldots n_s}(p_{j1}, p_{j2}, \ldots, p_{jn}) = \sum_{h_1=1}^{m_1} \sum_{h_2=1}^{m_2} \ldots \sum_{h_n=1}^{m_n} F_{h_1h_2\ldots h_n} \cdot A_{1h_1}(p_{j1}) \cdot \ldots \cdot A_{nh_n}(p_{jn}) \tag{6}$$

It approximates the function f in the point p_j. In [11,12] the multidimensional inverse F-transform (6) is applied in regression analysis to find dependencies between attributes in datasets.

To highlight the use of the multidimensional F-transform, consider, as an example, a dataset, given by two input features defined in the close intervals, respectively, [1.1, 4.9] and [0.1, 1.0].

Suppose we create for each of the two input variables a fuzzy partition consisting of three basic functions, setting $m_1 = 3$ and $m_2 = 3$. We obtain $c_1 = 1.9$ and $c_2 = 0.45$.

Table 1 shows the values of the three nodes for the two input variables.

Table 1. Set of nodes of the two input variables in the example.

Input Variable	Node	Value
First variable	x_1	1.1
	x_2	3.0
	x_3	4.9
Second variable	y_1	0.1
	y_2	0.55
	y_3	1.0

Figure 1 shows the points in the input variable plane. The four rectangles are drawn to show that the dataset is sufficiently dense with respect to the set of the two fuzzy partitions $\{A_{11}, A_{12}, A_{i3}\}$ and partitions $\{A_{21}, A_{22}, A_{23}\}$. In fact, in each rectangle in the figure, there is at least one point; this implies that for every combination of basic functions A_{1i}, A_{2i} $i = 1,2,3$, there exists at least one point $p_j = (p_{j1}, p_{j2})$ such that $A_{1i}\ (p_{j1})\ A_{2i}\ (p_{j1}) \neq 0$.

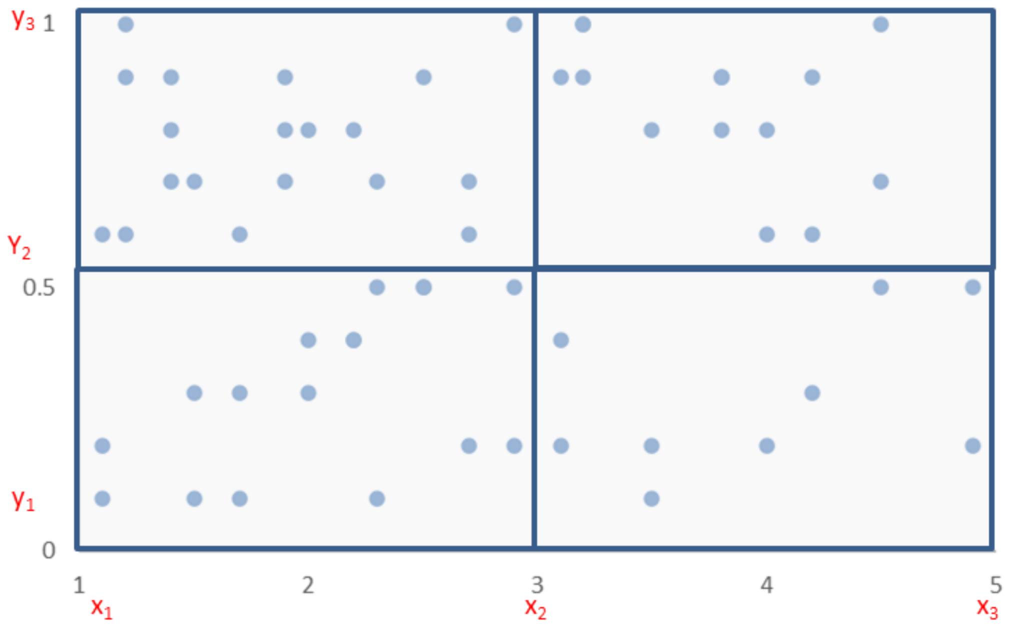


Figure 1. Data points and nodes in the example.

Since the data are sufficiently dense with respect to the sets of fuzzy partitions, it is possible to apply Equation (5) to calculate the components of the multidimensional direct F-transform $F_{h_1h_2}$ h_1, $h_2 = 1,2,3$. Finally, Equation (6) can be applied to calculate the multidimensional inverse F-transform in a point p; it approximates the function *f* in that point.

2.3. High Degree F-Transform

This paragraph introduces the concept of higher degree fuzzy transform or F^r-transform. One-dimensional square-integrable functions will now be considered.

Let A_h, $h = 1, \ldots ,n$, be the h*th* basic function defined on [a,b] and $L_2([x_{h-1},x_{h+1}])$ be the Hilbert space of square-integrable functions *f*,*g*: $[x_{h-1},x_{h+1}] \longrightarrow R$ with the inner product:

$$\langle f, g_h \rangle = \frac{\int_{x_{h-1}}^{x_{h+1}} f(x)g(x)A_k(x)dx}{\int_{x_{h-1}}^{x_{h+1}} A_h(x)dx} \tag{7}$$

Let $L^r{}_2([x_{h-1},x_{h+1}])$, with r positive integer, be a linear subspace of the Hilbert space $L_2([x_{h-1},x_{h+1}])$ with orthogonal basis given by polynomials $\{P^0{}_h, P^1{}_h, \ldots, P^r{}_h\}$ obtained applying the Gram-Schmidt orthonormalization to the linear independent system of polynomials $\{1, x, x^2, \ldots, x^r\}$ defined in the interval $[x_{h-1},x_{h+1}]$. We have:

$$\begin{cases} P_h^0 = 1 \\ P_h^{s+1} = x^{s+1} - \sum\limits_{j=1}^{s} \dfrac{\langle x^{s+1}, P_h^j \rangle}{\langle P_h^j, P_h^j \rangle} \quad s = 1, \ldots, r-1 \end{cases} \tag{8}$$

The following Lemma holds (Cfr. Perfilieva et al., 2011 [3], Lemma 1) :

Lemma 1. *Let F_k^r be the orthogonal projection of the function f on $L^r{}_2([x_{h-1},x_{h+1}])$. Then:*

$$F_h^r(x) = \sum_{s=1}^{r} c_{h,i} P_h^s(x) \tag{9}$$

where

$$c_{h,s} = \frac{\langle f, P_{kh}^s \rangle_k}{\langle P_h^s, P_h^s \rangle_h} = \frac{\int_{x_{k-1}}^{x_{k+1}} f(x) P_h^s(x) A_h(x) dx}{\int_{x_{k-1}}^{x_{k+1}} (P_h^s(x))^2 A_h(x) dx} \tag{10}$$

F_h^r it is the h-th component of the direct F^r-transform of f $F^r[f] = (F^r{}_1, F^r{}_2, \ldots, F^r{}_n)$.
The inverse F^r-transform of f in a point $x \in [a,b]$ is:

$$f_{F,n}^r(x) = \sum_{k=1}^{n} F_h^r A_k(x) \tag{11}$$

For r = 0 we have $P^0{}_h = 1$ and the F^0-transform is given by the F-transform in one variable ($F^0{}_h(x) = c_{h,0}$).

For r = 1 we have $P^1{}_h = (x - x_h)$ and the h-th component of the F^1-transform is given by the formula:

$$F^1{}_h(x) = c_{h,0} + c_{h,1}(x - x_h) = F^0{}_h(x) + c_{h,1}(x - x_h) \tag{12}$$

If the function f is known in a set of N data points $p_1, \ldots p_N$, $c_{h,0}$ and $c_{h,1}$ can be discretized in the form:

$$c_{h,0} = \frac{\sum_{i=1}^{n} f(p_i) A_h(p_i)}{\sum_{i=1}^{n} A_h(p_i)} \tag{13}$$

$$c_{k,1} = \frac{\sum_{i=1}^{n} f(p_i)(p_i - x_h) A_h(p_i)}{\sum_{i=1}^{n} A_h(p_i)(p_i - x_h)^2} \tag{14}$$

Likewise, let $L_2([x_{k1-1}, x_{k1+1}] \times [x_{k2-1},x_{k2+1}] \times \ldots \times [x_{kn-1}, x_{kn+1}])$ be the Hilbert space of square- integrable n-variables functions f: $[x_{k1-1}, x_{k1+1}] \times [x_{k2-1},x_{k2+1}] \times \ldots \times [x_{kn-1}, x_{kn+1}] \rightarrow R$ with the weighted inner product:

$$f, g_{h_1 h_2 \cdots h_n} = \int\limits_{x_{h_1-1}}^{x_{h_1+1}} \int\limits_{x_{h_2-1}}^{x_{h_2+1}} \cdots \int\limits_{x_{hn-1}}^{x_{hn+1}} f(x_1, x_2, \cdots, x_n) g(x_1, x_2, \cdots, x_n) A_{h_1}(x_1) A_{h_2}(x_2) \cdots A_{h_n}(x_n) dx_1 \cdots dx_n \tag{15}$$

Two function $f, g \in L_2([x_{h1-1}, x_{h1+1}] \times [x_{h2-1},x_{h2+1}] \times \ldots \times [x_{hn-1}, x_{hn+1}])$ are orthogonal if $\langle f, g \rangle_{h_1 h_2 \cdots h_n} = 0$.

Let f: $X \subseteq R^n \rightarrow Y \subseteq R$ be a continuous n-dimensional function defined in a closed set $[a_1,b_1] \times [a_2,b_2] \times \ldots \times [a_n,b_n]$. Let A_{hk}, k = 1, … ,n_k, be the k*th* basic function defined on the interval $[a_h,b_h]$ and $L_2([x_{h,k-1},x_{h,k+1}])$ be the Hilbert space of square-integrable functions f,g: $[x_{h,k-1},x_{h,k+1}] \longrightarrow R$.

The inverse F^1-transform of f in a point $x = (x_1, x_2, \ldots, x_n) \in [a_1,b_1] \times [a_2,b_2] \times \ldots \times [a_n,b_n]$ is:

$$f^1_{F,n}(x) = \sum_{h_1=1}^{m_1} \sum_{h_2=1}^{m_2} \ldots \sum_{h_n=1}^{m_n} F^1_{h_1h_2\ldots h_n} \cdot A_{1h_1}(x_1) \cdot \ldots \cdot A_{nh_n}(x_n) \tag{16}$$

where $F^1_{h_1h_2\ldots h_n}(x)$ is the $(h_1, h_2, \ldots, h_n)$*th* component of the direct F^1-transform, given by the formula:

$$F^1_{h_1h_2\ldots h_n}(x) = c^0_{h_1h_2\ldots h_n} + \sum_{s=1}^{n} c^{1s}_{h_1h_2\ldots h_n} \cdot (x_s - x_{h_t}) \tag{17}$$

If f is known in a set of N n-dimensional data points $p_1, \ldots p_N$ where $p_i = (p_{i1}, p_{i2}, \ldots, p_{in})$, we obtain:

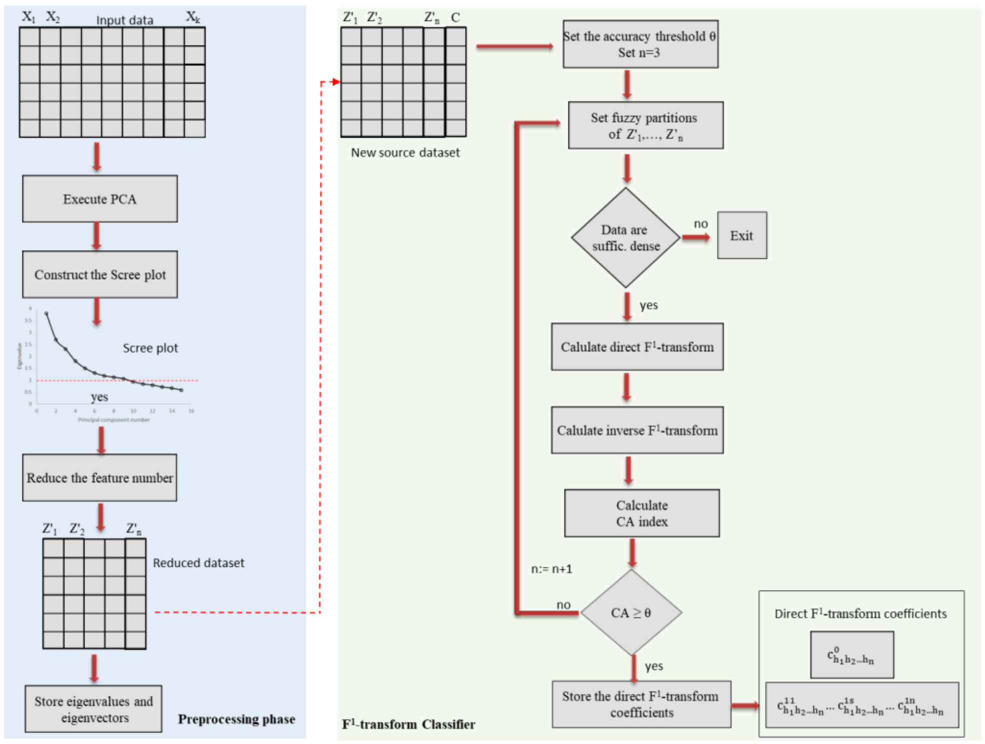
$$c^0_{h_1h_2\ldots h_n} = F_{h_1h_2\ldots h_n} = \frac{\sum_{j=1}^{N} f(p_{j1}, p_{j2}, \ldots p_{jn}) \cdot A_{1h_1}(p_{j1}) \cdot A_{2h_2}(p_{j2}) \cdot \ldots \cdot A_{nh_n}(p_{jn})}{\sum_{j=1}^{N} A_{1h_1}(p_{j1}) \cdot A_{2h_2}(p_{j2}) \cdot \ldots \cdot A_{nh_n}(p_{jn})} \tag{18}$$

$$c^{1s}_{h_1h_2\ldots h_n} = \frac{\sum_{j=1}^{N} f(p_{j1}, p_{j2}, \ldots p_{jn}) \cdot (p_{js} - x_{h_s}) \cdot A_{1h_1}(p_{j1}) \cdot A_{2h_2}(p_{j2}) \cdot \ldots \cdot A_{nh_n}(p_{jn})}{\sum_{j=1}^{N} (p_{js} - x_{h_s})^2 \cdot A_{1h_1}(p_{j1}) \cdot A_{2h_2}(p_{j2}) \cdot \ldots \cdot A_{nh_n}(p_{jn})} \tag{19}$$

where $c^0_{h_1h_2\ldots h_n}$ is the component $F_{h_1h_2\ldots h_n}$ of the multidimensional discrete direct F transform of f, given by (5).

3. The F^1-Transform Classification Method

The proposed classification method executes the multidimensional F^1-transform to classify data points. The method is schematized in Figure 2.



X1 X2 Input data Xk
Execute PCA
Construct the Scree plot
Scree plot
yes
Reduce the feature number
Z'1 Z'2 Z'n
Reduced dataset
Store eigenvalues and eigenvectors
Preprocessing phase
Z'1 Z'2 Z'n C
New source dataset
Set the accuracy threshold θ
Set n=3
Set fuzzy partitions of Z'1,..., Z'n
Data are suffic. dense
no
Exit
yes
Calulate direct F¹-transform
Calulate inverse F¹-transform
Calculate CA index
n:= n+1
no
CA ≥ θ
yes
Store the direct F¹-transform coefficients
Direct F¹-transform coefficients
F¹-transform Classifier

Figure 2. Overview of the F^1-transform classification method.

In a pre-processing phase, PCA is executed in order to reduce the number of features. The scree plot method is applied to detect the principal components. The eigenvalues are sorted in ascending order to create the Scree Plot; then, the elbow point is set; the features corresponding to the eigenvalues under the elbow point on the y-axis will be deleted. Output of the pre-processing phase is the new reduced dataset in the transformed coordinates.

The F^1-transform classifier follows the strategy adopted in the MFC algorithm [14]. Initially the classification accuracy threshold is set to θ and coarse-grained fuzzy partitions are created (n = 3). After creating the fuzzy partitions of the domains of each feature the algorithm checks that the data are sufficiently dense with respect to the fuzzy partitions; if they are, then the direct and inverse F^1-transforms are computed, otherwise, the algorithm terminates because the cardinality of the fuzzy partitions is too fine compared to the density of data points in the feature space and the direct F^1-transform cannot be computed.

The Calculate CA index component measures the accuracy of the classification. To classify a data point we adopt the following method applied in [9].

Let C be the number of classes and let $l_1, l_2, \ldots, l_C$ be the labels of the C classes. Let $p_j = (p_{j1}, p_{j2}, \ldots, p_{jn}, m_j)$ the *jth* data point, where m_j is the position of the corresponding class.

The components of the F1-transform (18) and (19) are given by:

$$c^0_{h_1h_2\ldots h_n} = F_{h_1h_2\ldots h_n} = \frac{\sum_{j=1}^N m_j \cdot A_{1h_1}(p_{j1}) \cdot A_{2h_2}(p_{j2}) \cdot \ldots \cdot A_{nh_n}(p_{jn})}{\sum_{j=1}^N A_{1h_1}(p_{j1}) \cdot A_{2h_2}(p_{j2}) \cdot \ldots \cdot A_{nh_n}(p_{jn})} \quad (20)$$

$$c^{1s}_{h_1h_2\ldots h_n} = \frac{\sum_{j=1}^N m_j \cdot (p_{js} - x_{h_s}) \cdot A_{1h_1}(p_{j1}) \cdot A_{2h_2}(p_{j2}) \cdot \ldots \cdot A_{nh_n}(p_{jn})}{\sum_{j=1}^N (p_{js} - x_{h_s})^2 \cdot A_{1h_1}(p_{j1}) \cdot A_{2h_2}(p_{j2}) \cdot \ldots \cdot A_{nh_n}(p_{jn})} \quad (21)$$

Let $f^1_{F,n}(p_j)$ the inverse F^1-transform computed by (16). The index of the class assigned to the data point p_j is given by an integer $\hat{m}_j$ in {1,2, ... ,C} given by:

$$\hat{m}_j = \left[\min_{m=1,\ldots,C} \left(\left| f^1_{F,n}(p_j) - m \right| \right) \right] \quad (22)$$

where [a] stands for the greatest integer containing the positive real number a.

The classification accuracy CA is given by the ratio between the number of data points correctly classified and the total number of data points.

If CA is less than the threshold θ, then finer fuzzy partitions with n + 1 fuzzy sets are created and the process is iterated. Otherwise, the F^1-transform classifier ends, storing the coefficients of the final direct F^1-transform: $c^0_{h_1h_2\ldots h_n}$ and $c^{11}_{h_1h_2\ldots h_n} \cdots c^{1n}_{h_1h_2\ldots h_n}$ computed, respectively by (18) and (19).

A new input data x = $(x_1, x_2, \ldots, x_n)$ will be classified computing the inverse F^1-transform $f^1_{F,n}(x)$ by (16) and assigning it the w*th* class by (20).

The F^1-transform classification algorithm is schematized below (Algorithm 1):

Algorithm 1. F^1-transform classification

```
1. Execute the PCA feature reduction algorithm
2. Use the Scree plot method to reduce the number of features
3. Create the reduced dataset
4. Set the accuracy threshold θ
5. n:=3
6. CA:=0 // initialize the CA index to 0
7. While CA < θ
8. Create the n-size fuzzy partitions of the feature domains
```

Algorithm 1. *Cont.*

```
9.  If data are sufficiently dense with respect to the n-dimensional fuzzy partitions Then
10. Calculate the direct $F^1$-transform by (17)
11. Calculate the inverse $F^1$-transform by (16)
12. Calculate the CA index
13. If CA $\geq \theta$ Then
14. Store the direct $F^1$-transform coefficients $c^0_{h_1h_2\ldots h_n}$ and $c^{11}_{h_1h_2\ldots h_n} \cdots c^{1n}_{h_1h_2\ldots h_n}$
15. Return "Data classified"
16. End if
17. ld>17. Else
18. Return "Data cannot be classified"
19. ld>19. End if
20. n:= n+1
21. End While
```

4. Experimental Results

We tested the F1-transform classification method on various classification datasets extracted from the UC Irvine Machine Learning Repository. Table 2 shows the size and the number of features of the classification datasets used in these comparison tests.

Table 2. Datasets in the UCI machine learning datasets used in the comparison analysis.

Dataset	Number of Data Points	Number of Features
Adult	48,842	14
Balance Scale	625	4
Bank Marketing	41,188	17
Breast cancer	286	9
Echocardiogram	132	12
Ecoli	336	7
Heart disease	303	14
Hepatitis	155	19
Thyroid disease	7200	21
Wine quality—red wine	1599	12
Wine quality—white wine	4898	12

To measure the classification performances, we compared the F^1-transform method with the Support Vector Machine (SVM) [24], Random Forest (RF) [25], Artificial Neural Network (ANN) [26], and the MFC classification algorithm (MFC) [14].

Our comparison tests were executed using an Intel Core I7 processor having a 5.4 GHz clock frequency.

After removing ambiguous data points, each dataset was randomly segmented into a training and a test set, containing 80% and 20% of the data points, respectively.

While running RF the number of decision trees is set to 100. The Radial Basis kernel function is used to execute SVM, setting the parameters c and gamma, respectively, to 1.0 and 0.1. ANN is executed by constructing a network having two hidden layers, and setting to 100 the maximum number of epochs.

For brevity, we show the complete results obtained for three classification datasets: the two datasets Red and White Wine Quality and the dataset Adult.

4.1. Red and White Wine Quality Classification Dataset

The red and white wine quality datasets are two classification datasets used to classify the quality of red and white variants of the Portuguese vinho verde wine.

The data points are vectors given by 11 physicochemical features. In Table 3 all the features are described.

Table 3. Red and white wine datasets: Description of the features.

Feature Name	Description	Type of Field
fixed acidity	tartaric acid (g/dm^3)	Continuous
volatile acidity	Acetic acid (g/dm^3)	Continuous
citric acid	Citric acid (g/dm^3)	Continuous
residual sugar	Residual sugar (g/dm^3)	Continuous
chlorides	Sodium chloride(g/dm^3)	Continuous
free sulfur dioxide	Free sulfur dioxide (mg/dm^3)	Continuous
total sulfur dioxide	Total sulfur dioxide (mg/dm^3)	Continuous
density	Density (g/cm^3)	Continuous
pH	pH	Continuous
sulphates	Potassium sulphate (g/dm^3)	Continuous
alcohol	Alcohol (vol.%)	Continuous
quality	Wine quality	List (an integer from 0 to 10)

The last feature, called quality, is the output feature containing the class of wine quality in a scale between 0 (very bad) to 10 (excellent).

The white wine quality dataset contains 4898 data points, and the red wine quality dataset contains 1599 data points. The two datasets are unbalanced, with over 70% of the data points classified with quality five and six. Paragraphs 4.1.1 and 4.1.2 show the results obtained, respectively, for the white and for the red wine quality datasets.

4.1.1. White Wine Quality Dataset—Classification Results

Now we show the results obtained on the white wine-quality dataset.

In the preprocessing phase, the PCA algorithm was executed on the training set. In Figure 3 the scree plot is shown. The orange line highlights the elbow point. In the transformed training set, the size of the data points is reduced from eleven to five. Then, the F^1-transform classification algorithm is executed on the transformed training set. Finally, the stored final F^1-transform coefficients are used to classify the data points in the transformed test set.

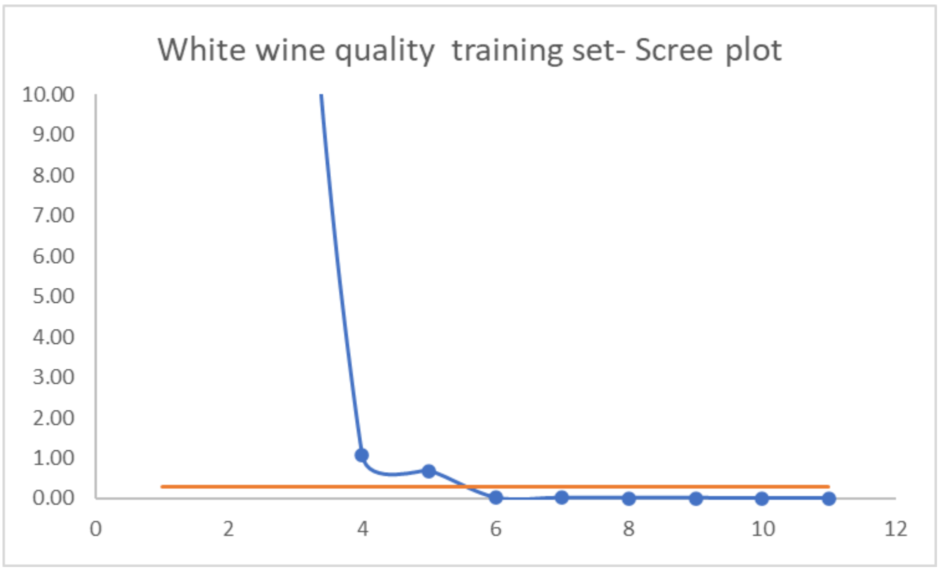


Figure 3. Scree plot obtained for the white wine quality training set; the orange line shows the elbow point.

We compare the classification results with the ones obtained executing SVM, RF, ANN, and MFC on the original training set.

Table 4 shows the accuracy, precision, recall, and F1-score measures computed on the training and test sets executing the five classification methods.

Table 4. White wine dataset: Classification performance comparison.

Phase	Method	Accuracy	Precision	Recall	F1-Score
Training	SVM	0.75	0.77	0.74	0.75
	RF	0.73	0.76	0.71	0.73
	ANN	0.78	0.79	0.77	0.78
	MFC	0.76	0.77	0.75	0.76
	F^1-transform	0.83	0.82	0.83	0.82
Test	SVM	0.74	0.75	0.74	0.74
	RF	0.73	0.74	0.71	0.72
	ANN	0.77	0.80	0.77	0.79
	MFC	0.76	0.76	0.74	0.76
	F^1-transform	0.82	0.81	0.82	0.81

The results in Table 3 show that the best performances are obtained by executing the F^1-transform classification method. The F^1-transform method improves the classification accuracy obtained by executing ANN by about 5% on the training and test sets; it improves the one obtained by executing MFC by about 7% on the training set and 6% on the test set.

4.1.2. Red Wine Quality Dataset—Classification Results

The PCA algorithm was executed on the training set. Figure 4 shows the scree plot. The orange line highlights the elbow point.

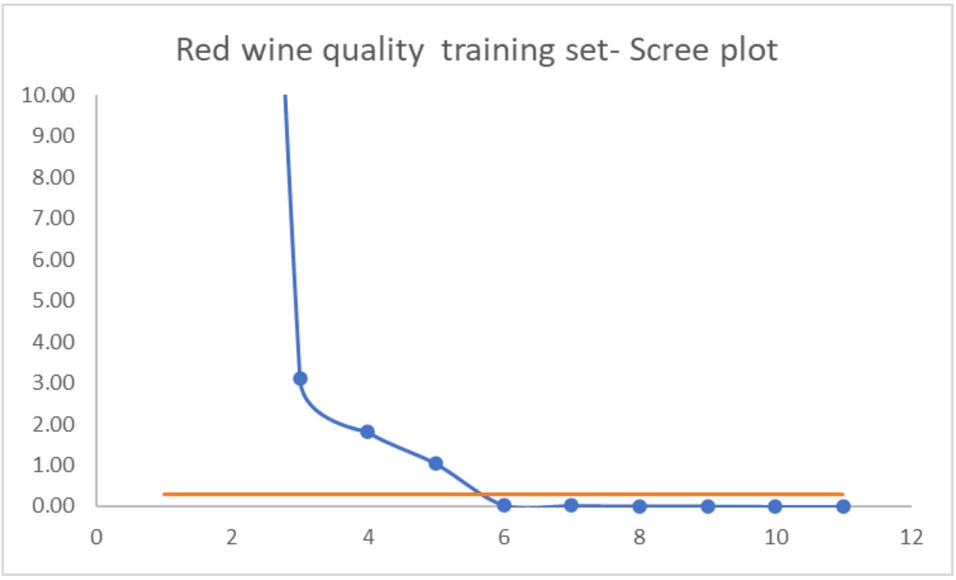


Figure 4. Scree plot obtained for the red wine quality training set; the orange line shows the elbow point.

As well as for the white wine quality training set, the size of the data points in the transformed training set is reduced from eleven to five.

Table 5 shows the Accuracy, Precision, Recall, and F1-score measures computed on the training and test sets executing the five classification methods.

The results of the tests applied on the red wine quality dataset confirm that the best performances are obtained executing the F^1-transform classification method The F^1-transform method improves the classification accuracy of MFC by about 5% on both training and test sets.

Table 5. Red wine dataset: Classification performance comparison.

Phase	Method	Accuracy	Precision	Recall	F1-Score
Training	SVM	0.68	0.68	0.67	0.67
	RF	0.70	0.68	0.66	0.67
	ANN	0.72	0.71	0.69	0.72
	MFC	0.69	0.69	0.68	0.69
	F^1-transform	0.76	0.74	0.75	0.75
Test	SVM	0.69	0.69	0.67	0.68
	RF	0.69	0.68	0.67	0.68
	ANN	0.72	0.73	0.72	0.72
	MFC	0.70	0.70	0.71	0.70
	F^1-transform	0.75	0.76	0.75	0.75

The F^1-transform method improves the classification accuracy obtained by executing ANN by about 4% on the training set and 3% on the test set; it improves the one obtained by executing MFC by about 7% on the training set and 5% on the test set.

4.2. Adult Classification Dataset

The Adult dataset contains information extracted from the United States census bureau database, which includes information on residents of various states to determine whether a citizen earns more or less than 50K USD per year.

The training set contains 32,561 unambiguous data points; the data points are vectors formed by fourteen input features, four continuous and ten categorical. The test set contains 16,282 unambiguous data points. Table 6 shows the description of each feature.

Table 6. Adult dataset: Description of the features.

Feature Name	Description	Type of Field
age	Age	Continuous
workclass	Work class	List
fnlwgt	Survey weight	Continuous
education	Education	List
education-num	Education number	Continuous
marital status	Marital status	List
occupation	Occupation	List
relationship	Type of relationship	List
race	Race	List
sex	Gender	List: (Female, Male)
capital-game	Capital-game	Continuous
capital-loss	Capital-loss	Continuous
hours-per-week	Hours of work per week	Continuous
native-country	Native country	List:
class	Annual income	List: (<=50 K, <50 K)

The output class feature, called class, assumes two values, depending on whether the person makes over 50K or under 50K USD a year. The dataset is unbalanced, where over 75% of the data points are classified with an annual income under 50K USD.

To execute F^1-transform, all the categorical features were transformed into integers assigning to each term in the list of values an integer starting from 1 to the number of unique values.

PCA is executed in order to reduce the number of features. Figure 5 shows the scree plot and the elbow point set to 2.5. The first nine components are selected and the input features are reduced from fourteen to nine.

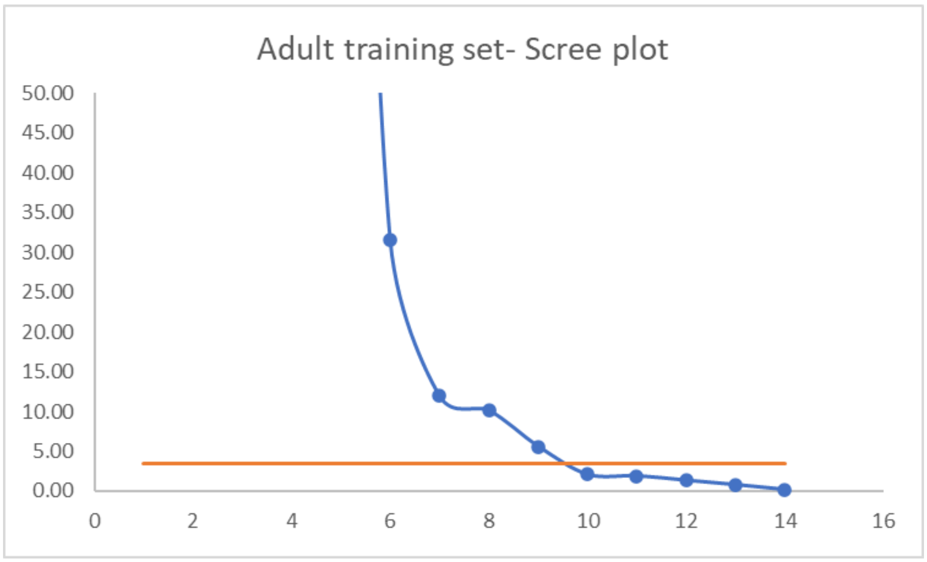


Figure 5. Adult training set—Scree plot; the orange line show the elbow point.

Then, the F^1-transform classification method is applied to the reduced training set. The comparison results with SVM, RF, ANN, and MFC are shown in Table 7.

Table 7. Adult dataset: Classification performance comparison.

Phase	Method	Accuracy	Precision	Recall	F1-Score
Training	SVM	0.87	0.85	0.84	0.84
	RF	0.85	0.84	0.83	0.83
	ANN	0.88	0.86	0.86	0.86
	MFC	0.86	0.85	0.85	0.85
	F^1-transform	0.93	0.89	0.90	0.89
Test	SVM	0.87	0.86	0.85	0.85
	RF	0.85	0.83	0.82	0.67
	ANN	0.88	0.87	0.86	0.86
	MFC	0.87	0.86	0.86	0.86
	F^1-transform	0.92	0.91	0.90	0.90

Even in this case, the best performances are obtained executing F^1-transform.

The F^1-transform method improves the classification accuracy obtained by executing ANN by about 4% on the training set and 3% on the test set; it improves the one obtained by executing MFC by about 5% on the training set and 4% on the test set.

Table 8 shows, for each dataset, the final classification accuracy and the CPU time measured executing the five algorithms.

Table 8. Classification Accuracy and CPU time comparison results.

Dataset	Method	SVM	RF	ANN	MFC	F^1-tr
Adult	Accuracy	0.87	0.85	0.88	0.86	0.93
	CPU time (s)	858.22	849.74	915.19	881.86	877.37
Balance scale	Accuracy	0.95	0.92	0.96	0.95	0.98
	CPU time (s)	26.48	26.19	28.41	27.63	26.92
Bank Marketing	Accuracy	0.77	0.75	0.78	0.77	0.82
	CPU time (s)	832.31	823.09	944.56	867.83	864.95
Breast cancer	Accuracy	0.90	0.88	0.91	0.90	0.94
	CPU time (s)	31.67	31.78	33.59	32.06	31.78

Table 8. *Cont.*

Dataset	Method	SVM	RF	ANN	MFC	F^1-tr
Diabetes	Accuracy CPU time (s)	0.73 44.77	0.73 43.46	0.76 47.09	0.75 45.02	0.77 44.91
Echocardiogram	Accuracy CPU time (s)	0.75 31.67	0.73 30.93	0.75 33.10	0.74 31.28	0.78 31.32
Ecoli	Accuracy CPU time (s)	0.79 32.65	0.78 32.38	0.81 34.32	0.79 32.77	0.83 32.82
Heart disease	Accuracy CPU time (s)	0.83 37.79	0.82 37.56	0.85 40.18	0.85 38.51	0.87 38.47
Hepatitis	Accuracy CPU time (s)	0.88 42.61	0.89 42.49	0.91 44.26	0.89 42.68	0.93 42.73
Thyroid disease	Accuracy CPU time (s)	0.95 103.34	0.94 102.85	0.97 113.93	0.96 102.98	0.99 102.87
Wine quality—red wine	Accuracy CPU time (s)	0.68 67.52	0.70 66.81	0.72 69.54	0.69 68.15	0.76 68.23
Wine quality—white wine	Accuracy CPU time (s)	0.75 58.60	0.73 58.46	0.78 59.87	0.76 58.72	0.83 58.85

In all cases the F^1-transform method improves the final accuracy obtained executing SVM, RS, ANN, and MFC; the final accuracy obtained using F^1-transform improves that obtained using MFC by a value ranging between 2% (dataset Diabetes) and 7% (dataset Adult).

The CPU times employed by executing F^1-transform are compatible with the ones employed by executing the other four classification algorithms. In fact, even if the construction of the multidimensional F^1-transform in the proposed algorithm requires more computational expenditure than that necessary to construct the multidimensional F-transform in the MFC algorithm, the application of the PCA algorithm in the preprocessing phase allows for the reduction of the size of the data points, reducing, thus, the computational complexity. The results show that the F^1-transform classification method improves the classification accuracy of all other classification methods, with CPU times comparable with those of MFC. In fact, the classification accuracy of F^1-transform improves that of MFC by a value between 2% and 7% and the CPU times are similar to those obtained by running MFC.

In a nutshell, in all tests performed F^1-transform produces classification accuracy better than that obtained with other classifiers and compatible execution times. In particular, the results in Table 8 show that:

- The F1-transform classifier improves the accuracy obtained with MFC by a percentage in the range between 2% and 7%;
- Even if the computation times of the multidimensional F^1-transform are higher than those of the multidimensional F-transform, the proposed classification algorithm has execution times similar to those of MFC.

A possible critical point of F^1-transform consists of the choice of the elbow point in the scree plot obtained by executing the PCA method in the preprocessing phase. In all the tests performed in the scree plot obtained, there is a sudden change in slope which allows the elbow point to be easily recognized. It could happen that this trend change in the scree plot is not so evident, to the point of not allowing the precise definition of the elbow point. In these cases, the solution could be to run F^1-transform multiple times, each time choosing a different number of main components and selecting the one that produces the greatest accuracy. However, this solution can significantly increase the CPU times of the algorithm.

5. Conclusions

We propose a classification method based on the multidimensional F^1-transform in which the PCA technique is applied in the preprocessing phase to reduce the size of the features. The main aim of this research is to improve the accuracy of the MFC classification algorithm based on the multidimensional F-transform, without further increasing the processing times.

We compared the performances of the proposed classification method both with those of MFC and with those of the well-known SVM, Random Forest, and ANN classification algorithms; the comparative tests were performed on classification datasets of the UCI machine learning repository.

The results show that the multidimensional F^1-transform classification algorithm increases the classification performances obtained executing SVM, Random Forest, ANN, and MFC; in addition, the execution times are comparable with the ones obtained running MFC.

We intend to carry out future research to test the use of the classification method based on the multidimensional F^1-transform on a more extensive and varied set of datasets, adapting the method to the management of ambiguous data points.

Author Contributions: Conceptualization, B.C. and F.D.M.; methodology, B.C. and F.D.M.; software, B.C. and F.D.M.; validation, B.C. and F.D.M.; formal analysis, B.C. and F.D.M.; investigation, B.C. and F.D.M.; resources, B.C. and F.D.M.; data curation, B.C. and F.D.M.; writing—original draft preparation, B.C. and F.D.M.; writing—review and editing, B.C. and F.D.M.; visualization, B.C. and F.D.M.; supervision, B.C. and F.D.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing is not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Perfilieva, I. Fuzzy transforms: Theory and applications. *Fuzzy Sets Syst.* **2006**, *157*, 993–1023. [CrossRef]
2. Di Martino, F.; Sessa, S. *Fuzzy Transforms for Image Processing and Data Analysis—Core Concepts, Processes and Applications*; Springer Nature: Cham, Switzerland, 2020; p. 217. [CrossRef]
3. Perfilieva, I.; Dankova, M.; Bede, B. Towards a higher degree F-transform. *Fuzzy Sets Syst.* **2011**, *180*, 3–19. [CrossRef]
4. Di Martino, F.; Sessa, S.; Perfilieva, I. A Summary of F-Transform Techniques in Data Analysis. *Electronics* **2021**, *10*, 1771. [CrossRef]
5. Hodáková, P.; Perfilieva, I. F^1-transform of functions of two variables. In Proceedings of the 8th conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2013), Milan, Italy, 11–13 September 2013; Advances in Intelligent Systems Research; Atlantis Press: Milano, Italy, 2013; pp. 547–553. [CrossRef]
6. Perfilieva, I.; Dankova, M.; Hurtik, P. Differentiation by the F-transform and application for edge detection. *Fuzzy Sets Syst.* **2014**, *288*, 96–114. [CrossRef]
7. Di Martino, F.; Sessa, S.; Perfilieva, I. First Order Fuzzy Transform for Images Compression. *J. Signal Inf. Process.* **2017**, *8*, 178–194. [CrossRef]
8. Di Martino, F.; Loia, V.; Perfilieva, I.; Sessa, S. An image coding/decoding method based on direct and inverse fuzzy trans-forms. *Int. J. Approx. Reason.* **2008**, *48*, 110–131. [CrossRef]
9. Molek, V.; Perfilieva, I. Deep Learning and Higher Degree F-Transforms: Interpretable Kernels Before and After Learning. *Int. J. Comput. Intell. Syst.* **2020**, *13*, 1404–1414. [CrossRef]
10. Di Martino, F.; Sessa, S. Seasonal Time Series Forecasting by F^1-Fuzzy Transform. *Sensors* **2019**, *19*, 3611. [CrossRef] [PubMed]
11. Perfilieva, I.; Novák, V.; Dvořák, A. Fuzzy transform in the analysis of data. *Int. J. Approx. Reason.* **2007**, *48*, 36–46. [CrossRef]
12. Di Martino, F.; Loia, V.; Sessa, S. Fuzzy transforms method in prediction data analysis. *Fuzzy Sets Syst.* **2011**, *180*, 146–163. [CrossRef]
13. Novák, V.; Perfilieva, I.; Dvořák, A.; Holčapek, M.; Kreinovich, V. Filtering out high frequencies in time series using F-transform. *Inf. Sci.* **2014**, *274*, 192–209. [CrossRef]
14. Di Martino, F.; Sessa, S. A classification algorithm based on multi-dimensional fuzzy transforms. *J. Ambient Intell. Humaniz. Comput.* **2022**, *13*, 2873–2885. [CrossRef]

15. Dimitoglou, G.; Adams, J.A.; Jim, C.M. Comparison of the C4.5 and a naive Bayes classifier for the prediction of lung cancer survivability. *Comput. Inf. Sci.* **2012**, *4*, 1–9. [CrossRef]
16. Maron, O.; Moore, A.W. The Racing Algorithm: Model Selection for Lazy Learners. In *Lazy Learning*; Aha, D.W., Ed.; Springer: Dordrecht, The Netherlands, 1997; Volume 11, pp. 193–225. [CrossRef]
17. Bhargawa, N.; Sharma, G.; Bhargava, R.; Mathuria, M. Decision tree analysis on J48 algorithm for data mining. *Int. J. Adv. Res. Comput. Sci. Softw. Eng. (IJARCSSE)* **2013**, *3*, 1114–1119.
18. Chaudhuri, B.B.; Bhattacharya, U. Efficient training and improved performance of multilayer perceptron in pattern classification. *Neurocomputing* **2007**, *34*, 11–27. [CrossRef]
19. Hurtik, P.; Perfilieva, I. Fast Training and Real-Time Classification Algorithm Based on Principal Component Analysis and F-Transform. In Proceedings of the 2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS), Toyama, Japan, 5–8 December 2018; pp. 275–280. [CrossRef]
20. Sehgal, S.; Singh, H.; Agarwal, M.; Bhasker, V.; Shantanu. Data analysis using principal component analysis. In Proceedings of the 2014 International Conference on Medical Imaging, m-Health and Emerging Communication Systems (MedCom), Greater Noida, India, 7–8 November 2014; pp. 45–48. [CrossRef]
21. Joliffe, I.T.; Morgan, B. Principal component analysis and exploratory factor analysis. *Stat. Methods Med. Res.* **1992**, *1*, 69–95. [CrossRef] [PubMed]
22. Joliffe, I.T. *Principal Component Analysis. Springer Series in Statistics*, 2nd ed.; Springer: New York, NY, USA, 2010; p. 516, ISBN 978-1441929990.
23. Joliffe, I.T.; Cadima, J. Principal component analysis: A review and recent developments. *Philos. Trans. A* **2016**, *374*, 20150202. [CrossRef] [PubMed]
24. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
25. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
26. Zhang, G.P. Neural Networks for Classification: A Survey. *IEEE Trans. Syst. Man Cybern.—Part C Appl. Rev.* **2000**, *30*, 451–462. [CrossRef]