symmetry

MDPI

Article

Reduced-Kernel Weighted Extreme Learning Machine Using Universum Data in Feature Space (RKWELM-UFS) to Handle Binary Class Imbalanced Dataset Classification

Roshani Choudhary * and Sanyam Shukla

Department of Computer Science and Engineering, Maulana Azad National Institute of Technology, Bhopal 462003, India; sanyamshukla@gmail.com or sanyamshukla@manit.ac.in

* Correspondence: choudhary.roshani99@gmail.com or 173112005@stu.manit.ac.in

Abstract: Class imbalance is a phenomenon of asymmetry that degrades the performance of traditional classification algorithms such as the Support Vector Machine (SVM) and Extreme Learning Machine (ELM). Various modifications of SVM and ELM have been proposed to handle the class imbalance problem, which focus on different aspects to resolve the class imbalance. The Universum Support Vector Machine (USVM) incorporates the prior information in the classification model by adding Universum data to the training data to handle the class imbalance problem. Various other modifications of SVM have been proposed which use Universum data in the classification model generation. Moreover, the existing ELM-based classification models intended to handle class imbalance do not consider the prior information about the data distribution for training. An ELM-based classification model creates two symmetry planes, one for each class. The Universum-based ELM classification model tries to create a third plane between the two symmetric planes using Universum data. This paper proposes a novel hybrid framework called Reduced-Kernel Weighted Extreme Learning Machine Using Universum Data in Feature Space (RKWELM-UFS) to handle the classification of binary class-imbalanced problems. The proposed RKWELM-UFS combines the Universum learning method with a Reduced-Kernelized Weighted Extreme Learning Machine (RKWELM) for the first time to inherit the advantages of both techniques. To generate efficient Universum samples in the feature space, this work uses the kernel trick. The performance of the proposed method is evaluated using 44 benchmark binary class-imbalanced datasets. The proposed method is compared with 10 state-of-the-art classifiers using AUC and G-mean. The statistical *t*-test and Wilcoxon signed-rank test are used to quantify the performance enhancement of the proposed RKWELM-UFS compared to other evaluated classifiers.

Keywords: class imbalance; classification; Reduced Kernel Extreme Learning Machine; Universum samples

Citation: Choudhary, R.; Shukla, S. Reduced-Kernel Weighted Extreme Learning Machine Using Universum Data in Feature Space (RKWELM-UFS) to Handle Binary Class Imbalanced Dataset Classification. *Symmetry* **2022**, *14*, 379. https://doi.org/10.3390/sym14020379

Academic Editor: Alexander Shelupanov

Received: 20 January 2022
Accepted: 10 February 2022
Published: 14 February 2022

1. Introduction

The performance of a classification problem is affected by various data complexity measures such as class imbalance, class overlapping, length of the decision boundary, small disjuncts of classes, etc. In the classification domain, most of the real-world problems are class imbalanced. Examples of such problems are cancer detection [1,2], fault detection [3], intrusion detection system [4], software test optimization [5], speech quality assessment [6], pressure prediction [7], etc. In a problem when the number of samples in one class outnumbers the numbers of samples in some other class, it is considered as a class imbalanced/asymmetric problem. The class with a greater number of instances is the majority class and the class with fewer instances is the minority class. In real-world problems, usually, the minority class instances have more importance than the majority class.

Traditional classifiers such as the support vector machine (SVM), Naive Bayes, decision tree, and extreme learning machine (ELM) are biased towards the correct classification

of majority class data. Various approaches have been proposed to handle such class-imbalanced classification problems, which can be classified as data sampling, algorithmic and hybrid methods [8].

In classification, the idea of using additional data along with the original training data has been used widely for better training of the model. The virtual example method, oversampling method, noise injection method, and Universum data creation method are some examples that use additional data. The oversampling method generates additional data in the majority class to balance the data distribution in the classes. In the virtual example and noise injection methods, labeled synthetic data are created that may not come from the same distribution as the original data. Universum data creation methods allow the classifier to encode prior knowledge by representing meaningful concepts in the same domain as the problem at hand as stated in [9]. In Universum learning-based classification models, the Universum data are added to the training data to enhance performance. Universum data are data that do not belong to any of the target classes. The two main factors which affect the performance of Universum data are the number of Universum data created and the method used for the creation of Universum data. Different methods have been used for the creation of Universum; among those, the two most common methods widely used are the use of examples from other classes and random averaging [9].

Several methods have been proposed that use Universum data in the training of SVM based classifiers to handle the class imbalance problem, such as the Universum Support Vector Machine (USVM) [9], Twin support vector machine with Universum data (TUSVM) [10], and Cost-Sensitive Universum-SVM (CS-USVM) [11]. A Universum support vector machine-based model for EEG signal classification has been proposed in [12]. A nonparallel support vector machine for a classification problem with Universum learning has been proposed in [13]. An improved non-parallel Universum support vector machine and its safe sample screening rules are proposed in [14]. Tencer et al. [15] used Universum data with other classifiers such as fuzzy models to demonstrate its usefulness in combination with fuzzy models. Recently, a Multiple Universum Empirical Kernel Learning (MUEKL) [16] classifier has been proposed to handle class imbalance by combining the Universum learning with Multiple Empirical Kernel Learning (MEKL).

Extreme Learning Machine (ELM) [17] is a single hidden-layer feed-forward neural network designed for regression and classification with fast speed and better generalization performance, but it cannot handle the classification of class-imbalanced problems effectively. Various ELM based models have been proposed to handle the classification of class imbalance problems, such as Weighted Extreme Learning Machine (WELM) [18], Class-Specific Cost Regulation Extreme Learning Machine (CCR-ELM) [19], Class-Specific Kernelized Extreme Learning Machine (CSKELM) [20], Reduced-Kernelized Weighted Extreme Learning Machine (RKWELM) [21], UnderBagging-based Kernelized Weighted Extreme Learning Machine (UBKWELM) [22], and UnderBagging-based Reduced-Kernelized Weighted Extreme Learning Machine (UBRKWELM) [21]. The proposed work is motivated by the idea that none of the existing ELM-based models for classification encode prior knowledge in the training model using Universum data.

This work proposes a novel hybrid classification model called Reduced-Kernel Weighted Extreme Learning Machine using Universum data in Feature Space (RKWELM-UFS) which incorporates the Universum data in the RKWELM model. The contributions of the proposed approach are listed below.

1. This work is the first attempt that utilized the Universum data in a Reduced-Kernelized Weighted Extreme Learning Machine (RKWELM)-based classification model to handle the class imbalance problem.
2. The Weighted Kernelized Synthetic Minority Oversampling Technique (WKSMOTE) [23] is an oversampling-based classification method in which the synthetic samples are created in the feature space of the Support Vector Machine (SVM). Inspired by WKSMOTE, the proposed work creates the Universum samples in the feature space.

3. The proposed method uses the kernel trick to create the Universum samples in the feature space between randomly selected instances of the majority and minority classes.
4. In a classification problem, the samples located near the decision boundary contribute more to better training. The creation of Universum samples in feature space ensures that the Universum samples lie near the decision boundary.

The rest of the paper is structured as follows. In the related work section, Universum learning, class imbalance learning, ELM classifier, and its variants are discussed in detail. The proposed work section provides a detailed explanation of the proposed RKWELM-UFS classifier. The experimental setup and result analysis section provide the specification of the dataset used in the experiments, parameter settings of the proposed algorithm, the evaluation metrics used for performance evaluation, and the experimental results obtained in form of various tables and figures. The last section provides the concluding remarks and future research directions.

2. Related Work

The following section provides the literature related to Universum learning, class imbalance learning, and some of the existing ELM-based models to handle class imbalance learning.

2.1. Universum Learning

The idea of using Universum data is close to the idea of using the prior knowledge in Bayesian classifiers [9]. However, there is a conceptual difference between the two approaches, i.e., the prior knowledge is knowledge about decision rules used in Bayesian inference, while the Universum is knowledge about the admissible collection of examples. Similarly to the Bayesian prior probability, the Universum data encode prior information.

It has been observed by various researchers [9,15,24] that the effect of Universum is dependent on the quality of Universum samples created. A safe sample screening rule for Universum support vector machines, in which the non-contributed data can be identified and safely eliminated before the training process, can obtain the same solution as solving the original problem is proposed in [25]. An improved version of the non-parallel Universum support vector machine and its safe sample screening rule is proposed in [14]. It is suggested in [24] that not all the Universum samples are helpful for effective classification, so they proposed selecting the informative Universum samples for semi-supervised learning, which is a method used to identify informative samples among the Universum samples. An empirical study on the Universum support vector machine (USVM), which describes some practical conditions for evaluating the effectiveness of random averaging for the creation of Universum data, is performed in [26].

2.2. Class Imbalance Learning

The classification performance of traditional classifiers degrades when there is an imbalance in the ratio of the majority and minority class data. Different approaches have been used in classification to deal with the problem of class imbalance. Table 1 provides the categorization of the proposed methods and other methods used in this work for comparison. Table 1 also provides the strategy and basic ideas used in the respective methods. The broad categories of these approaches are discussed in the following subsections.

2.2.1. Data Level Approach

The data-level methods are based on balancing the ratio of data to convert an imbalanced classification problem into a balanced classification problem. These methods can be seen as data pre-processing methods because they try to handle the class imbalance present in the data before the classification model generation. The data-level approaches can be broadly categorized as under-sampling, oversampling and, hybrid sampling methods.

Table 1. Categorization and comparison of the proposed method and other methods in comparison used to handle classification of class imbalance problems.

Category	Strategy	Method	Basic Idea of the Method
Algorithmic	Cost Sensitive	WELM	This method minimizes the weighted least-square error to handle the class imbalance.
		CCR-KELM	This method assigns a class-specific regularization parameter to handle class imbalance.
		RKELM	This method uses a reduced number of centroids in kernels function to handle class imbalance
Data-level	Under-sampling	Random-Under-sampling	This method uses random under-sampling to balance the imbalanced training data.
	Oversampling	SMOTE	This method creates artificial minority class samples to balance the imbalanced training data.
		CSMOTE	This method generates some artificial samples whose dimension is equal to 5 times the number of minority samples.
	Universum	USVM	This method creates Universum data to sift the separating hyper plane of the SVM classifier
		MUEKL	This method combines Multiple Empirical Kernel Learning with the Universum learning.
Hybrid	Data-level combined with Ensemble	RUS-Boost	This method combines RUS with boosting.
		UBKELM	This method uses random under-sampling with KELM-based ensemble.
		UBRKELM	This method uses random under-sampling with RKELM-based ensemble.
	Cost-sensitive combined with Ensemble	BWELM	This method combines boosting with WELM.
	Data-level combined with cost sensitive	RKWELM-UFS (the proposed method)	The proposed method creates a Universum sample in the feature space and uses RKWELM as the classification algorithm.

The under-sampling methods remove some of the data (i.e., the majority samples) to decrease the imbalance ratio of a training dataset. These methods may suffer from data loss, as some of the important samples may be removed. The efficiency of an under-sampling method lies in its ability to select the right samples which can be removed from the dataset. The under-sampling methods reduce the time complexity of a given class-imbalanced classification problem. A combined weighted multi-objective optimizer for instance reduction in a two-class imbalanced data problem is proposed in [27]. Clustering-Based Under-Sampling (CBUS) [28] uses clustering of majority class data for the under-sampling. Fast Clustering-Based Under-Sampling (FCBUS) [29] is a modified version of CBUS which clusters the minority class data for under-sampling to reduce the time complexity of CBUS.

The oversampling method adds some additional data (in the minority class) to decrease the imbalance ratio of the training dataset. The additional samples are obtained by creating synthetic minority class samples or replicating the existing minority class samples. These methods can lead to over-fitting problems in model generation. The oversampling methods increase the time complexity of a given class imbalance classification problem. The synthetic minority oversampling technique (SMOTE) [30] is a popular oversampling method, widely used to handle class imbalance, in which synthetic minority samples are created. Several variants of SMOTE have been proposed to further enhance the performance of class imbalance dataset classification, such as Borderline SMOTE, Borderline SMOTE1, Borderline SMOTE2, Safe- Level-SMOTE, MSMOTE [31], and CSMOTE [32]. The hybrid sampling methods such as SCUT [16] try to reduce the class imbalance by using both oversampling and under-sampling.

2.2.2. Algorithmic Approach

There are some approaches in which the classification algorithm is able to handle class imbalance problems, such as cost-sensitive and one-class learning approaches. The cost-sensitive methods assign a different cost to the misclassification of different classes. In an imbalance problem, generally, the misclassification cost of minority class samples is higher than the misclassification cost of majority class samples. The efficiency of any cost-sensitive method lies in the selection of misclassification costs for different classes. Multiple Random Empirical Kernel Learning (MREKL) [33] is a cost-sensitive classification model which emphasizes the importance of samples located in overlapping regions of positive and negative classes and ignores the effects of noisy samples to achieve better performance in class imbalance problems. Weighted Extreme Learning Machine (WELM) [18] is a weighted version of Extreme Learning Machine (ELM) [17] that minimizes the weighted error by incorporating a weight matrix in the optimization problem of ELM. Class-Specific Extreme Learning Machine (CSELM) [34] is a variant of WELM which replaces the weight matrix with two constant weight values for each class. Class-Specific Kernel Extreme Learning Machine (CSKELM) [20] is the modification of CSELM which uses the Gaussian kernel function to map the input data to feature space. Class-Specific Cost-Regulation Extreme Learning Machine (CCR-KELM) [19] is the variant of KELM which uses different regularization perimeters for the classes.

The one-class learning approach is also called single-class learning. In these methods, the classifier learns only one class as the target class. In this approach generally, the minority class is considered as the target class. Multi-Kernel Support Vector Data Description with boundary information proposes a novel method called MKL-SVDD [35] by introducing Multi- Kernel Learning (MKL) into the traditional Support Vector Data Description (SVDD) based on the boundary information to form one-class learning.

2.2.3. Hybrid Approach

In a hybrid approach, multiple classification approaches are combined to handle a class imbalance problem. Some hybrid techniques combine ensemble techniques with data sampling methods such as over-sampling or under-sampling to handle class imbalance problems. RUSBoost [36] is a hybrid technique that combines random under-sampling with boosting to create an ensemble of classifiers. UBKELM [22] and UBRKELM [21] are two hybrid classification models that combine underbagging with KELM and RKELM respectively. BPSO-AdaBoost-KNN [37] is a method that implements BPSO as the feature selection algorithm and then designs an AdaBoost-KNN classifier to convert the traditional weak classifier into a strong classifier. UBoost: Boosting with the Universum [38] is a technique that combined the Universum sample creation with a boosting framework. An Adaptive-Boosting (AdaBoost) algorithm [39] uses multiple iterations to learn multiple classifiers in a serial manner to generate a single strong learner.

Some hybrid techniques combine cost-sensitive approaches with ensemble techniques such as Ensemble of Weighted Extreme Learning Machine (EWELM) [40] and Boosting Weighted Extreme Learning machine (BWELM) [41]. In EWELM, the weight of each component classifier in the ensemble is optimized by using a differential evolution algorithm. BWELM is a modified AdaBoost framework that combines multiple Weighted ELM-based classifiers in a boosting manner. The main idea of BWELM is to find the better weights in each base classifier.

2.3. Extreme Learning Machine (ELM) and Its Variants to Handle Class Imbalance Learning

ELM [17,42] is a generalized single hidden-layer feed-forward neural network, which provides good generalization performance and disposes of the iterative time-consuming training process. It uses the Moore–Penrose pseudoinverse for computing the weights between the hidden and the output layer which make it fast. For a given classification dataset with N training samples $\{(x_i, t_i)\}_{i=1}^{N}$, where $x_i = [x_{i1}, x_{i2,\ldots\ldots,} x_{in}]^T \epsilon R^n$ is the input feature vector and $t_i = [t_{i1}, t_{i2,\ldots\ldots,} t_{im}]^T \epsilon R^m$ is the output label vector. Here, the

vector/matrix transpose is denoted by superscript T. During the training time, these weights are randomly generated and are not changed further. The hidden neurons bias matrix is denoted by $b = b_1\left[, b_2, ..b_j, .. b_L\right]^T \epsilon\ R^L$, where b_j is the bias of the jth hidden neuron. In ELM, for a given training/testing sample, i.e., x_i, the hidden layer output $h(x_i)$ is calculated as follows:

$$h(x_i) = G(ax_i + b) \tag{1}$$

Here, $G(.)$ is the activation function of the hidden neurons. In ELM, for a binary classification problem, the decision function, i.e., $f(x_i)$ for a sample x_i is given as:

$$f(x_i) = sign(h(x_i)\beta) \tag{2}$$

where β is the output weight matrix. The hidden layer output matrix H can be written as follows:

$$H = \begin{bmatrix} h_1(x_1) & h_2(x_1) & . & h_L(x_1) \\ h_1(x_2) & h_2(x_2) & . & h_L(x_2) \\ . & . & . & . \\ h_1(x_N) & h_2(x_N) & . & h_L(x_N) \end{bmatrix}_{N \times L} \tag{3}$$

ELM minimizes the training error and the norm of the output weights as:

$$Minimize : \|H\beta - T\|^2\ and\ \|\beta\| \tag{4}$$

In the original implementation of ELM [17], the minimal norm least-square method instead of the standard optimization method was used to find β.

$$\beta = H^+T \tag{5}$$

where H^+ is the Moore–Penrose generalized inverse of matrix H. In [17,42] the orthogonal projection method is used to calculate H^+, which can be used in two cases.

When H^TH is nonsingular then,

$$H^+ = \left(H^TH\right)^{-1}H^T$$

When H^TH is singular then

$$H^+ = H^T\left(HH^T\right)^{-1}$$

In ELM the constrained optimization-based problem for classification with multiple output nodes was formulated as follows:

$$Minimize = \frac{1}{2}\|\beta\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\xi_i\|^2 \tag{6}$$

Subjected to: $h(x_i)\beta = t_i^T - \xi_i^T,\ i = 1, \ldots,\ N$

The output layer weights β can be obtained using two solutions

Case 1. Where the Number of Training Samples is Not Huge:

$$\beta = H^T\left(\frac{I}{C} + HH^T\right)^{-1}T \tag{7}$$

Case 2. Where the Number of Training Samples is Huge:

$$\beta = \left(\frac{I}{C} + H^TH\right)^{-1}H^TH \tag{8}$$

2.3.1. Weighted Extreme Learning Machine (WELM)

Conventional ELM does not account for good generalization performance while dealing with the class-imbalance learning problems. Weighted Extreme Learning Machine (WELM) [18] is a cost-sensitive version of ELM which was proposed for handling the class-imbalanced learning problem effectively. In cost-sensitive learning methods, the different cost is assigned to the misclassification of different class samples. In WELM, two generalized weighting schemes were proposed. These generalized weighting schemes assign weights to the training samples as per their class distribution. In WELM [18], the following optimization problem is formulated:

$$Minimize = \frac{1}{2}\|\beta\|^2 + \frac{1}{2}CW\sum_{i=1}^{N}\|\xi_i\|^2 \tag{9}$$

Subjected to: $h(x_i)\beta = t_i^T - \xi_i^T,\ i = 1, \ldots,\ N$

Here, C is the regularization parameter and $W = diag(W_{ii})$ is a $N \times N$ diagonal matrix whose diagonal elements are the weights assigned to the training samples. The two weighting schemes proposed by WELM are:

Weighting scheme *W*1:

$$W_{ii} = \frac{1}{q_k} \tag{10}$$

Here, $k = t_i$ and q_k is the total number of samples belonging to kth class.

Weighting scheme *W*2:

$$W_{ii} = \begin{cases} \frac{0.618}{qk}\ if\ (q_k > q_{avg}) \\ \frac{1}{qk}\ if\ (q_k <= q_{avg}) \end{cases} \tag{11}$$

Here, q_{avg} represents the average number of samples for all classes. Weight W_{ii} is assigned to the ith samples. Samples belonging to the minority class will be assigned weights equal to $1/q_i$, in both the weighting schemes. The second weighting scheme assigns a lesser weight to the majority class samples compared to the first weighting scheme. The two variants of WELM are sigmoid node-based WELM and Gaussian kernel-based WELM, which are described as follows.

- **Sigmoid node-based Weighted Extreme Learning Machine**

The Sigmoid node-based WELM uses random input weights and Sigmoid activation function i.e., G(.), to find the hidden layer output matrix H given in Equation (3). The solution of the optimization problem of WELM as given in [18] is reproduced below:

$$\beta = \begin{cases} H^T\left(\frac{I}{C} + WHH^T\right)^{-1}WT\ if\ (N > L) \\ \left(\frac{I}{C} + H^TWH\right)^{-1}H^TWH\ if\ (N > L) \end{cases} \tag{12}$$

The two solutions are given for two cases. The first solution is given for the case when the number of training samples is smaller than the number of selected hidden layer neurons. The second solution is given for the case where the number of selected hidden layer neurons is smaller than the number of training samples.

- **Gaussian kernel-based Weighted Extreme Learning Machine (KWELM)**

In KELM [42], the kernel matrix of the hidden layer is represented as follows:

$$\Omega = \begin{bmatrix} K(x_1,\ x_1) & K(x_1,\ x_2) & . & K(x_1,\ x_N) \\ K(x_2,\ x_1) & K(x_2,\ x_2) & . & K(x_2,\ x_N) \\ . & . & . & . \\ K(x_N,\ x_1) & K(x_N,\ x_2) & . & K(x_N,\ x_N) \end{bmatrix}_{N\times N} \tag{13}$$

The Gaussian kernel-based WELM maps the input data to the feature space as follows:

$$K(x_i,\ x_j) = exp\left(\frac{-\|x_i - x_j\|^2}{\sigma^2}\right) \tag{14}$$

Here, σ represents the kernel width parameter, x_i represents the ith sample and x_j represents the jth centroid; $(i,\ j \in 1,\ 2,\ \ldots N)$. $K(x_i,\ x_j)$ represents the distance of the jth centroid x_j to the ith input sample x_i. The number of Gaussian kernel functions i.e., the centroids used in [32] was equal to the number of training samples. On applying Mercer's condition, the kernel matrix of KELM [42] can be represented as given below:

$$\Omega_{KELM} = HH^T : \Omega_{KELM_{i,j}} = h(x_i).h(x_j) = K(x_i,\ x_j\ ) \tag{15}$$

The output of KWELM is determined in [18] which is represented as follow:

$$f(x) = sign\begin{bmatrix} K(x, x_1) \\ . \\ . \\ K(x, x_N) \end{bmatrix}^T \left(\frac{I}{C} + W\Omega_{KELM}\right)^{-1} WT \tag{16}$$

Compared to the Sigmoid node-based WELM, KWELM has better classification performance, as stated in [18].

2.3.2. Reduced Kernel Weighted Extreme Learning Machine (RKWELM)

Reduced-Kernel Extreme Learning Machine (RKELM) [43] is a fast and accurate kernel-based supervised algorithm for classification. Unlike Support Vector Machine (SVM) or Least-Square SVM (LS-SVM), which identify the support vectors or weight vectors iteratively, the RKELM randomly selects a subset of the available data samples as centroids or mapping samples. The weighted version of RKELM i.e., Reduced-Kernel Weighted Extreme Learning Machine (RKWELM) is proposed in [21] for class imbalance learning. In RKWELM, a reduced number of kernels are selected, which act as the centroids. The number of Gaussian kernel functions used in RKWELM is denoted as $\widetilde{N}$ where $\widetilde{N} \subset N$. The kernel matrix of the hidden layer can be reproduced as given by the following equation.

$$\Omega_{RKELM} = \begin{bmatrix} K(x_1,\ x_1) & K(x_1,\ x_2) & . & K(x_1,\ x_N) \\ K(x_2,\ x_1) & K(x_2,\ x_2) & . & K(x_2,\ x_N) \\ . & . & . & . \\ K(x_N,\ x_1) & K(x_N,\ x_2) & . & K(x_N,\ x_{\widetilde{N}}) \end{bmatrix}_{N \times \widetilde{N}} \tag{17}$$

Here, x_i represents the ith sample and x_j represents the jth centroid $(i \in 1,\ 2,\ \ldots N)$ and $(j \in 1,\ 2,\ \ldots N)$. In the case when $\left(N == \widetilde{N}\right)$, the output of RKWELM can be given by the following equation.

The final output of RKWELM, as given in [43], is computed as:

$$f(x) = sign\begin{bmatrix} K(x, x_1) \\ . \\ K(x, x_{\widetilde{N}}) \end{bmatrix}^T \left(\frac{I}{C} + \Omega_{RKELM}^T W \Omega_{RKELM}\right)^{-1} \Omega_{RKELM}^T WT \tag{18}$$

2.3.3. UnderBagging-Based Kernel Extreme Learning Machine (UBKELM)

UnderBagging-Based Kernel Extreme Learning Machine (UBKELM) [22] is an ensemble of KELM. UBKELM creates several balanced training subsets by random undersampling of the majority class samples. K is the number of balanced subsets that are created by selecting M number of majority samples and all the minority samples in each subset, where M is the number of minority samples in the training dataset and K is the ceiling

value of the imbalance ratio of the training dataset. In the subset creation, the majority samples are selected using the random under-sampling method. There are two variants of UBKELM, i.e., UnderBagging-Based Kernel Extreme Learning Machine-Max Voting (UBKELM-MV) and UnderBagging-Based Kernel Extreme Learning Machine-Soft Voting (UBKELM-SV) in which the ultimate outcome of the ensemble is computed by majority voting and soft voting respectively.

2.3.4. UnderBagging-Based Reduced-Kernelized Weighted Extreme Learning Machine

UnderBagging-based Reduced-Kernelized Weighted Extreme Learning Machine (UBRKELM) [21] is an ensemble of Reduced Kernelized Weighted Extreme Learning Machine (RKWELM). The UBRKELM creates several balanced training subsets and learns multiple classification models with these balanced training subsets using RKWELM as the classification algorithm. K is the number of balanced subsets that are created by selecting M number of majority samples and all the minority samples in each subset, where M is the number of minority samples in the training dataset and K is the ceiling value of the imbalance ratio of the training dataset. In UBRKELM the reduced number of kernel functions is used as centroids to learn an RKELM model. Two variants of UBRKWELM are proposed, UBRKWELM-MV and UBRKWELM-SV, in which the final outcome of the ensemble is computed by majority voting and soft voting respectively.

3. Proposed Method

This work proposes a novel Reduced-Kernel Weighted Extreme Learning Machine using Universum data in Feature Space (RKWELM-UFS) to handle the class imbalance classification problem. In the proposed work, the Universum data along with the original training data is provided to the classifier for training purposes, to improve its learning capability. The proposed method creates Universum samples in the feature space because the mapping of input data from the input space to the feature space is not conformal.

The following subsections describe the process of creation of the Universum samples in the input space, the process of creation of the Universum samples in the feature space, the proposed RKWELM-UFS classifier, and the computational complexity of the proposed RKWELM-UFS classification model. Algorithm 1 provides the pseudo-code of the proposed RKWELM-UFS.

3.1. Generation of Universum Samples in the Input Space

To generate a Universum sample x_u between a majority sample x_m and a minority sample x_n, the following equation can be used:

$$x_u = x_m + \delta(x_n - x_m) \tag{19}$$

where δ represents a random number in the uniform distribution U [0, 1].

3.2. Generation of Universum Samples in the Feature Space

To generate a Universum sample in the feature space between a majority sample x_m and a minority sample x_n the following equation can be utilized:

$$\phi(x^{mn}) = \phi(x_m) + \delta^{mn}(\phi(x_n) - \phi(x_m)) \tag{20}$$

where, $\phi(.)$ is the feature transformation function which is generally unknown and δ^{mn} is a random number between [0, 1]. The proposed work uses δ^{mn} = 0.5. Similarly to SVM, LS-SVM, and PSVM, the transformation function $\phi(.)$ need not be known to users; instead, its kernel function $K(x_m,\ x_n)$ can be deployed. If a feature mapping $\phi(.)$ is unknown to users, one can apply Mercer's conditions on ELM to define a kernel matrix for KELM [17] as follows:

$$\Omega_{KELM} = HH^T : \Omega_{KELM_{m,n}} = K(x_m,\ x_n) = \phi(x_m)^T.\phi(x_n) \tag{21}$$

In the proposed work, we have to calculate the kernel function $K\left(x_i,\ x_j^{mn}\right)$, where x_i represents the original target training sample and x_j^{mn} is the Universum sample. According to [23] without computing $\phi(x_i)$ and (x_j^{mn}), we can obtain the corresponding kernel $K\left(x_i, x_j^{mn}\right)$ using the following equation:

$$\begin{gathered} K\left(x_j^{mn},\ x_i\right) = \phi(x_i)^T\ \phi\left(x_j^{mn}\right) = \phi(x_i)^T(\phi(x_m) + \delta^{mn}(\phi(x_n) - (\phi(x_m))) = \\ \phi(x_i)^T(\phi(x_m) + \delta^{mn}\phi(x_i)^T\phi(x_n) - \delta^{mn}\phi(x_i)^T\phi(x_m) = K(x_i,\ x_m) + \\ \delta^{mn}K(x_i,\ x_n) + \delta^{mn}K(x_i,\ x_m)K\left(x_j^{mn},\ x_i\right) = (1-\delta^{mn})K(x_i,\ x_m) + \delta^{mn}K(x_i,\ x_n) \end{gathered} \tag{22}$$

3.3. Proposed Reduced-Kernel Weighted Extreme Learning Machine Using Universum Samples in Feature Space (RKWELM-UFS)

Training of an ELM [42] based classifier requires the computation of the output layer weight matrix β. The proposed RKWELM-UFS uses the same equation as RKWELM [21] to obtain the output layer weight matrix β which is reproduced below:

$$\beta = \left(\frac{I}{C} + \Omega_{RKELM-UFS}^T\ W\Omega_{RKELM-UFS}\right)^{-1}\Omega_{RKELM-UFS}^T\ WT \tag{23}$$

where, W is the diagonal weight matrix, which gives different weights to the majority class, the minority class, and the Universum instances using Equation (10), T is the target vector in which the class label for Universum samples is set to 0 (given the class label of majority and minority class are +1 and -1 respectively), and $\Omega_{RKELM-UFS}$ is the kernel matrix of the proposed RKWELM-UFS.

In the proposed work, the Universum instances are added to the training process along with the original training instances. The reason behind computing β in the same manner as RKWELM is that the proposed RKWELM-UFS computes the kernel matrix $\Omega_{RKELM-UFS}$ by deploying the original training instances excluding the Universum instances as centroids. The value of $\Omega_{RKELM-UFS}$ is obtained by augmentation of the two matrices Ω_{RKELM} *and* Ω_{UFS} . The following subsections describe the computation of Ω_{RKELM} , Ω_{UFS} and $\Omega_{RKELM-UFS}$.

3.3.1. Computation of Ω_{KELM}

The proposed work computes the kernel matrix for the N number of original training instances termed as Ω_{KELM} in the same manner as it was computed in the KELM [42], which is represented as:

$$\Omega_{KELM} = \begin{bmatrix} K(x_1,\ x_1) & K(x_1,\ x_2) & . & K(x_1,\ x_N) \\ K(x_2,\ x_1) & K(x_2,\ x_2) & . & K(x_2,\ x_N) \\ . & . & . & . \\ K(x_N,\ x_1) & K(x_N,\ x_2) & . & K(x_N,\ x_N) \end{bmatrix}_{N\times N} \tag{24}$$

3.3.2. Computation of Ω_{UFS}

Equation (20) can be used to create a Universum sample $\phi(x_{mn})$ between two original training samples $\phi(x_m)$ and $\phi(x_n)$ in feature space. As we have discussed the transformation function $\phi(.)$ is unknown to the user, so the computation of $\phi(x_{mn})$ is not possible here. For convenience, we will refer to the Universum sample $\phi(x_{mn})$ as $\phi(u_i)$. In the proposed work without computing $\phi(u_i)$, we can directly compute the corresponding kernel $K(u_i, x_j)$. $K(u_i, x_j)$ is calculated using Equation (22). In the proposed algorithm, only the original training samples are used as centroids, so the matrix Ω_{UFS} for p number of Universum samples and N number of original training samples can be represented as:

$$\Omega_{UFS} = \begin{bmatrix} K(u_1,\ x_1) & K(u_1,\ x_2) & . & K(u_1,\ x_N) \\ K(u_2,\ x_1) & K(u_2,\ x_2) & . & K(u_2,\ x_N) \\ . & . & . & . \\ K(u_p,\ x_1) & K(u_p,\ x_2) & . & K(u_p,\ x_N) \end{bmatrix}_{p \times N} \tag{25}$$

3.3.3. Computation of $\Omega_{\text{RKELM}-\text{UFS}}$

The addition of Universum samples in the training process requires that the original kernel matrix i.e., Ω_{RKELM} be augmented to include the matrix Ω_{UFS}. The final hidden layer output kernel matrix of the proposed RKWELM-UFS is obtained by augmentation of the two matrices Ω_{KELM} and Ω_{UFS} which is denoted as $\Omega_{RKELM-UFS}$.

$$\Omega_{RKELM-UFS} = \begin{bmatrix} K(x_1,\ x_1) & K(x_1,\ x_2) & . & K(x_1,\ x_N) \\ K(x_2,\ x_1) & K(x_2,\ x_2) & . & K(x_2,\ x_N) \\ . & . & . & . \\ . & . & . & . \\ K(x_N,\ x_1) & K(x_N,\ x_2) & . & K(x_N,\ x_N) \\ K(u_1,\ x_1) & K(u_1,\ x_2) & . & K(u_1,\ x_N) \\ K(u_2,\ x_1) & K(u_2,\ x_2) & . & K(u_2,\ x_N) \\ . & . & . & . \\ . & . & . & . \\ K(u_p,\ x_1) & K(u_p,\ x_2) & . & K(u_p,\ x_N) \end{bmatrix}_{((N+P)\times N)} \tag{26}$$

The output of RKWELM-UFS can be obtained using Equation (18) used in RKWELM, which is reproduced below:

$$f(x) = sign\begin{bmatrix} K(x_t, x_1) \\ . \\ K(x_t, x_i) \\ . \\ . \\ K(x_t, x_N) \end{bmatrix}^T \left(\frac{I}{C} + \Omega_{RKELM-UFS}^T\, W \Omega_{RKELM-UFS}\right)^{-1} \Omega_{RKELM-UFS}^T\, WT \tag{27}$$

Here x_t represents the test instance and x_i represent the training instance for $i = 1, 2, \ldots, N$.

Algorithm 1 Pseudocode of the proposed RKWELM-UFS

INPUT: Training Dataset $X(x_i, t_i)_{i=1}^N$
Number of Universum samples to be generated: p
OUTPUT:
1: Calculate the kernel matrix $\Omega_{KELM}\ \epsilon\ (N \times N)$ as shown in Equation (24) for the N number of original training instances using Equation (21).
2: Calculate the kernel matrix $\Omega_{UFS}\ \epsilon\ (p \times N)$ as shown in Equation (25) for the N number of training instances and p number of Universum instances as follows.
for j = 1 to p
Randomly select one majority instance x_m
Randomly select one minority instance x_n
for i = 1 to N
calculate $K\left(x_j^{mn},\ x_i\right)$ using Equation (22)
End
End
3: Augment the matrix Ω_{KELM} with the matrix Ω_{UFS} to obtain the reduced kernel matrix using Universum samples $\Omega_{RKELM-UFS}$ shown in Equation (26).
4: To obtain the output weight matrix β use the Equation (23).
5: To determine the class label of an instance x use the Equation (27).

3.4. Computational Complexity

For training of the ELM-based classification algorithm, it is necessary to obtain the output layer weight matrix i.e., β. For the proposed RKWELM-UFS β is obtained using Equation (23) which is reproduced below:

$$\beta = \left(\frac{I}{C} + \Omega_{RKELM-UFS}^{T} W \Omega_{RKELM-UFS}\right)^{-1} * \Omega_{RKELM-UFS}^{T} WT \tag{28}$$

Here, $\Omega_{RKELM-UFS}$ is a matrix of size $(N+p) \times N$, where N is the number of training instances and p is the number of Universum samples. The weight matrix, i.e., W, is of size $(N+p) \times (N+p)$ and the target matrix i.e., T is of size $(N+p) \times (c)$ where c is the number of target class labels; here, the number of target class labels is 2 because we are using the binary classification problems. To compute $\Omega_{RKELM-UFS}$ first we need to compute the Ω_{RKELM} and Ω_{UFS}. In the following steps the computational complexity of computing β is identified step by step:

1. The computational complexity of calculating Ω_{RKELM} i.e., the kernel matrix shown in Equation (24) is $O(nN^2)$, where n is the number of features of training data in input space.
2. The computational complexity of calculating matrix Ω_{UFS} shown in Equation (25) is $O(p)$.
3. The computational complexity of the output weights β can be calculated as
   - 3.1 Matrix multiplications: $(\Omega_{RKELM-UFS}^{T} W \Omega_{RKELM-UFS})$
     Computational complexity: $O\left(2N(N+p)^2\right)$
   - 3.2 Computational complexity of computing the inverse of $N \times N$ matrix computed in Step 3.1 is $O(N^3)$
   - 3.3 Computational complexity of matrix multiplications $\left(\Omega^T WT\right)$ is $O(N^2(N+p) + Nc(N+p))$
   - 3.4 Computational complexity of matrix multiplication of 2 matrices obtained in Step 3.1 and Step 3.3 is $O(N^2 c)$

The final computational complexity of calculating β is $O(2N(N+p)^2 + N^3 + N^2(N+p) + N^2 n + N^2 c + Nc(N+p) + p)$. The computational complexity can be simplified to $O(N^3)$ because the value of c is 2, the value of n is smaller than N, and the maximum value of p can be N.

4. Experimental Setup and Result Analysis

This section provides the experiments performed to evaluate the proposed work, which includes the specification of the datasets used for experimentation, the parameter settings of the proposed algorithm, the evaluation metrics used for performance comparison, and the results obtained through experiments and performance comparison with the state-of-the-art classifiers.

4.1. Dataset Specifications

The proposed work uses 44 binary class-imbalanced datasets for performing the experiments. These datasets are downloaded from the KEEL dataset repository [44,45] in 5-fold cross-validation format. Table 2 provides the specification of these datasets. In Table 2, # Attributes denote the number of features, # Instances denotes the number of instances and, IR denotes the class imbalance ratio in the presented datasets. The class imbalance ratio (IR) for the binary class dataset can be defined as follows:

$$IR = \frac{number\ of\ instances\ in\ majority\ class}{number\ of\ instances\ in\ minority\ class} \tag{29}$$

Table 2. Specification of 44 benchmark datasets from KEEL dataset repository.

Dataset Name	# Attributes	IR (%)	# Instances	Dataset Name	# Attributes	IR (%)	# Instances
abalone9-18	8	16.70	731	glass6	9	6.43	214
ecoli-01	7	0.54	220	haberman	3	2.81	306
ecoli-013726	7	43.80	281	iris0	4	2.00	150
ecoli-01235	7	9.26	244	new-thyroid1	5	5.14	215
ecoli-01465	6	13.00	280	new-thyroid2	5	5.14	215
ecoli-01472356	7	10.65	336	page-blocks134	10	16.14	472
ecoli-014756	6	12.25	332	pima	8	1.87	768
ecoli-015	6	11.00	240	segment0	19	6.02	2308
ecoli-02345	7	9.06	202	shuttle-c0c4	9	13.78	1829
ecoli-026735	7	9.53	224	Shuttle-c2c4	9	24.75	129
ecoli-0345	7	9.00	200	Vehicle0	18	3.25	846
ecoli-03465	7	9.25	205	Vehicle1	18	2.91	846
ecoli-034756	7	9.25	257	Vehicle2	18	2.89	846
ecoli-0465	6	9.13	203	Vehicle3	18	3.00	846
ecoli-06735	7	9.41	222	vowel0	13	9.97	988
ecoli-0675	6	10.00	220	wisconsin	9	1.86	683
ecoli1	7	3.39	336	yeast05679vs4	8	10.00	528
Ecoli2	7	5.54	336	yeast1289vs7	8	31.00	947
glass016vs2	9	10.00	192	yeast1458vs7	8	28.00	693
glass0123vs456	9	4.00	214	yeast1vs7	7	15.00	459
glass1	9	1.85	214	yeast1vs8	8	24.00	482
glass4	9	16.00	214	yeast3	8	8.13	1484

The datasets used for the experiments are normalized using min-max normalization in the range [1, −1] using the following equation:

$$x' = \left(\frac{x - min_n}{max_n - min_n} \right) * 2 - 1 \tag{30}$$

Here, the original feature value of nth feature is denoted by x, minimum value of nth feature is denoted by min_n and the maximum value of nth feature is denoted by max_n.

4.2. Evaluation Matrix

The confusion matrix, also called the error matrix, can be employed to evaluate the performance of a classification model. It allows the visualization of the performance of an algorithm. In a confusion matrix TP denotes True Positive, TN denotes True Negative, FP denotes False Positive, and FN denotes False Negative.

Accuracy is not a suitable measure to evaluate the performance of a classifier when dealing with a class-imbalanced problem. The other performance matrices used for the performance evaluation in such problems are G-mean and AUC (area under the ROC curve). The AUC defines the measure of the entire area under the ROC curve in two dimensions. The ROC known as receiver operating characteristic curve is a graph that shows the performance of the model by plotting TP_{rate} and TN_{rate} on the graph.

$$G_{mean} = \sqrt{TP_{rate} * TN_{rate}}$$

$$AUC = \frac{1 + TP_{rate} - TN_{rate}}{2}$$

Here,

$$TP_{rate} = \frac{TP}{TP + FN} \; and \; TN_{rate} = \frac{TN}{TN + FP}$$

4.3. Parameter Settings

The proposed RKWELM-UFS creates Universum samples between randomly selected pairs of majority and minority samples. Because of the randomness, this work presents the

mean (denoted as tstR or TestResult) and standard deviation (denoted as std) of the test G-mean and test AUC obtained for 10 trials. The proposed RKWELM-UFS has two parameters, namely the regularization parameter C and the Kernel width parameter σ (denoted as KP). The optimal values of these parameters are obtained using grid search, by varying them on the range $\{2^{-18}, 2^{-16}, \ldots, 2^{48}, 2^{50}\}$ *and* $\{2^{-18}, 2^{-16}, \ldots, 2^{18}, 2^{20}\}$ respectively.

4.4. Experimental Results and Performance Comparison

The proposed RKWELM-UFS is compared with three sets of algorithms used to handle class imbalance learning. The first set contains the existing approaches which use Universum samples in the classification model generation to handle class-imbalanced problems such as MUEKL [16] and USVM [9]. The second set of approaches consists of the single classifiers such as KELM [46], WKELM [18], CCR-KELM [19], and WKSMOTE [23] which are used to handle class-imbalanced problems. The third set contains the popular ensemble classifiers such as RUSBoost [36], BWELM [41], UBRKELMMV [21], UBRKELM-SV [21], UBKELM-MV [22], and UBKELM-SV [22].

The statistical *t*-test and Wilcoxon signed-rank test are used to evaluate the performance of the proposed RKWELM-UFS and other methods in consideration. In the *t*-test result, the value of H (null hypothesis) is 1 if the test rejects the null hypothesis at the 5% significance level, and 0 otherwise.

In the Wilcoxon signed-rank test result, the value of H (null hypothesis) is 1 if the test rejects the null hypothesis that there is no difference between the grade medians at the 5% significance level. In the statistical tests, the *p*-value indicates the level of significant difference between the compared algorithms; the lower the *p*-value, the higher the significant difference between the compared algorithms. This work uses AUC and G-mean as the measures of the performance evaluation. The AUC results of classifiers MUEKL and USVM shown in Table 3 are obtained from the work MUEKL [16].

Table 3. Performance comparison of the proposed RKWELM-UFS with other existing Universum-based classifiers in terms of average AUC (std, KP, and C denote the standard deviation, Kernel width parameter, and regularization parameter, respectively.

Dataset	MUEKL Test Result% ± std.	USVM Test Result% ± std.	(KP, C)	RKWELM-UFS Test Result% ± std.
abalone9-18	75.06 ± 12.53	69.92 ± 12.75	$(2^6, 2^{26})$	**94.97 ± 0.52**
ecoli-01	**98.67 ± 1.83**	97.29 ± 2.5	$(2^2, 2^8)$	**98.67 ± 0.00**
ecoli-01235	90.68 ± 17.67	85.27 ± 14.45	$(2^{-2}, 2^{-18})$	**92.68 ± 0.00**
ecoli013726	85.00 ± 22.36	92.88 ± 1.63	$(2^4, 2^{-2})$	**95.99 ± 0.00**
ecoli-01465	90.00 ± 13.69	89.62 ± 11.34	$(2^{10}, 2^{50})$	**94.34 ± 1.56**
ecoli01472356	88.00 ± 7.28	86.73 ± 9.4	$(2^2, 2^6)$	**93.95 ± 0.11**
ecoli-014756	91.51 ± 4.76	88.13 ± 4.05	$(2^{12}, 2^{42})$	**94.93 ± 0.04**
ecoli-015	91.59 ± 10.77	88.41 ± 9.62	$(2^8, 2^{34})$	**95.91 ± 0.04**
ecoli-02345	93.89 ± 7.89	88.93 ± 10.36	$(2^8, 2^{34})$	**94.53 ± 0.08**
ecoli-026735	86.51 ± 11.9	78.82 ± 12.03	$(2^2, 2^4)$	**90.22 ± 0.09**
ecoli-0345	**92.22 ± 11.73**	91.11 ± 11.65	$(2^8, 2^{40})$	91.59 ± 3.29
ecoli-03465	91.96 ± 6.76	88.24 ± 7.94	$(2^{10}, 2^{44})$	**97.15 ± 0.07**
ecoli-034756	94.49 ± 5.20	88.40 ± 11.89	$(2^{10}, 2^{40})$	**95.55 ± 0.00**
ecoli-0465	92.23 ± 10.97	89.19 ± 11.15	$(2^6, 2^{34})$	**94.70 ± 0.06**
ecoli-06735	89.50 ± 16.97	86.00 ± 16.62	$(2^{-2}, 2^0)$	**92.68 ± 0.06**
ecoli-0675	**91.75** ± 7.05	87.50 ± 7.55	$(2^6, 2^{40})$	91.59 ± 0.21
ecoli1	90.48 ± 6.29	87.16 ± 5.03	$(2^2, 2^{11})$	**93.62 ± 0.29**
ecoli2	94.31 ± 4.47	88.78 ± 5.23	$(2^0, 2^8)$	**95.35 ± 0.04**
glass1	79.66 ± 7.41	67.64 ± 4.64	$(2^{-4}, 2^4)$	**81.67 ± 0.25**
glass6	93.06 ± 7.08	90.63 ± 6.33	$(2^6, 2^{10})$	**93.41 ± 0.21**
haberman	64.27 ± 4.35	62.84 ± 4.56	$(2^8, 2^{42})$	**68.17 ± 0.73**
new-thyroid1	**100.00 ± 0.00**	96.03 ± 3.7	$(2^{-4}, 2^{24})$	**100.00 ± 0.00**
new-thyroid2	**100.00 ± 0.00**	94.37 ± 4.49	$(2^8, 2^{42})$	99.98 ± 0.04

Table 3. *Cont.*

Dataset	MUEKL Test Result% ± std.	USVM Test Result% ± std.	(KP, C)	RKWELM-UFS Test Result% ± std.
page-blocks134	84.21 ± 19.45	71.49 ± 16.64	$(2^0, 2^{12})$	**100.00 ± 0.00**
pima	73.03 ± 3.11	70.16 ± 5.63	$(2^0, 2^4)$	**79.62 ± 0.05**
Segment0	99.22 ± 0.90	89.02 ± 3.74	$(2^{-4}, 2^2)$	**99.93 ± 0.00**
shuttle-c0c4	**100.00 ± 0.00**	99.77 ± 0.27	$(2^2, 2^{-8})$	**100.00 ± 0.00**
shuttle-c2c4	**100.00 ± 0.00**	**100.00 ± 0.00**	$(2^4, 2^{18})$	**100.00 ± 0.00**
vehicle0	99.18 ± 0.66	81.28 ± 6.51	$(2^4, 2^{34})$	**99.88 ± 0.13**
vehicle1	77.43 ± 4.15	62.37 ± 5.14	$(2^6, 2^{32})$	**90.26 ± 0.45**
vehicle2	99.15 ± 0.68	83.59 ± 1.52	$(2^2, 2^{28})$	**99.73 ± 0.00**
vehicle3	76.47 ± 4.81	65.08 ± 3.32	$(2^8, 2^{40})$	**89.88 ± 0.25**
vowel0	**100.00 ± 0.00**	93.61 ± 3.63	$(2^{-10}, 2^{-18})$	**100.00 ± 0.00**
wisconsin	97.99 ± 0.61	97.09 ± 1.77	$(2^{12}, 2^{42})$	**99.08 ± 0.09**
yeast3	87.72 ± 2.18	89.60 ± 2.12	$(2^{10}, 2^{38})$	**95.09 ± 0.12**
Average	90.26 ± 6.73	85.34 ± 6.83		**94.15 ± 0.25**

4.4.1. Performance Analysis in Terms of AUC

Tables 3–5 provide the performance of the proposed RKWELM-UFS and other classification models in terms of AUC. The reported test AUC of the proposed RKWELM-UFS given in Tables 3–5 is the averaged test AUC obtained in 10 trials, using 5-fold cross-validation in each trial. Table 3 provides the performance of the proposed RKWELM-UFS and the existing Universum-based classifiers MUEKL and USVM on 35 datasets in terms of average AUC, where the RKWELM outperforms the other classifiers on 32 datasets. Table 4 provides the performance of the proposed RKWELM-UFS and the existing single classifiers like KELM, WKELM, CCR-KELM, and WKSMOTE on 21 datasets in terms of average AUC, where the RKWELM outperforms the other classifiers on 14 datasets. Table 5 provides the performance of the proposed RKWELM-UFS and the existing ensemble of classifiers such as RUSBoost, BWELM, UBRKELM-MV, UBRKELMSV, UBKELM-MV, UBKELM-SV on 21 datasets in terms of average AUC, where the RKWELM outperforms the other classifiers on 10 datasets.

Table 4. Performance comparison of the proposed RKWELM-UFS with existing single classifiers in terms of average AUC (std., KP, and C denotes the standard deviation, Kernel width parameter, and regularization parameter, respectively.

Dataset	KELM Test Result%	WKELM Test Result%	CCR-KELM Test Result%	WKSMOTE Test Result%	(KP, C)	RKWELM-UFS Test Result% ± std.
abalone9vs18	83.81	**95.24**	83.81	90.91	$(2^6, 2^{26})$	94.97 ± 0.52
ecoli01vs5	89.50	92.39	89.50	**96.22**	$(2^8, 2^{34})$	95.91 ± 0.04
glass0123vs456	93.84	97.03	93.84	**98.86**	$(2^{-2}, 2^4)$	97.66 ± 0.54
glass016vs2	81.36	84.11	81.36	83.52	$(2^{16}, 2^{40})$	**88.25 ± 0.16**
glass4	88.08	93.37	88.00	**94.86**	$(2^8, 2^{36})$	93.34 ± 0.07
haberman	63.91	67.81	63.91	67.34	$(2^8, 2^{42})$	**68.17 ± 0.73**
iris0	**100.00**	**100.00**	**100.00**	**100.00**	$(2^{-10}, 2^{-18})$	**100.00 ± 0.00**
newthyroid1	99.60	**100.00**	99.60	99.71	$(2^{-4}, 2^{24})$	**100.00 ± 0.00**
newthyroid2	99.60	**100.00**	99.60	99.92	$(2^8, 2^{42})$	99.98 ± 0.04
pageblock13vs4	98.00	**100.00**	98.00	99.96	$(2^0, 2^{12})$	**100.00 ± 0.00**
pima	74.14	78.30	74.14	79.18	$(2^0, 2^4)$	**79.62 ± 0.05**
segment0	97.89	98.07	99.80	99.91	$(2^{-4}, 2^2)$	**99.93 ± 0.00**
shuttleC0vsC4	**100.00**	**100.00**	**100.00**	**100.00**	$(2^2, 2^{-8})$	**100.00 ± 0.00**
shuttleC2vsC4	**100.00**	**100.00**	**100.00**	**100.00**	$(2^4, 2^{18})$	**100.00 ± 0.00**
vowel0	**100.00**	**100.00**	**100.00**	57.38	$(2^{-10}, 2^{-18})$	**100.00 ± 0.00**

Table 4. *Cont.*

Dataset	KELM Test Result%	WKELM Test Result%	CCR-KELM Test Result%	WKSMOTE Test Result%	(KP, C)	RKWELM-UFS Test Result% ± std.
wisconsin	98.15	98.75	98.15	98.8	$(2^{12}, 2^{42})$	**99.08 ± 0.09**
yeast05679vs4	74.54	**85.72**	74.54	78.8	$(2^{-2}, 2^{6})$	85.63 ± 0.37
yeast1289vs7	65.45	78.73	65.45	77.51	$(2^{6}, 2^{28})$	**79.96 ± 0.66**
yeast1458vs7	61.76	70.63	61.76	**74.91**	$(2^{6}, 2^{18})$	71.49 ± 0.41
yeast1vs7	72.15	81.00	72.15	82.89	$(2^{4}, 2^{16})$	**83.67 ± 0.10**
yeast2vs8	79.39	83.55	79.39	85.55	$(2^{-2}, 2^{48})$	**85.56 ± 0.00**
Average	86.72	90.70	86.81	88.87		**91.58 ± 0.18**

Table 5. Performance Comparison of the proposed RKWELM-UFS with existing ensemble of classifiers in terms of average AUC (std, KP, and C denote the standard deviation, Kernel width parameter, and regularization parameter, respectively).

Dataset	RUSBoost TstR ± std.	BWELM TstR	UBRKELM MV TstR ± std.	UBRKELM SV TstR ± std.	UBKELM MV TstR ± std.	UBKELM SV TstR ± std.	(KP, C)	RKWELM UFS TstR ± std.
abalone9vs18	93.67 ± 0.87	94.13	96.74 ± 1.21	**96.84 ± 0.68**	96.79 ± 0.55	96.55 ± 0.50	$(2^{6}, 2^{26})$	94.97 ± 0.52
ecoli01vs5	93.94 ± 2.37	93.94	**97.53 ± 0.30**	97.00 ± 0.09	94.90 ± 2.34	94.13 ± 2.25	$(2^{8}, 2^{34})$	95.91 ± 0.04
glass0123vs456	97.48 ± 0.72	96.57	97.61 ± 0.97	97.40 ± 0.34	97.36 ± 0.80	97.35 ± 0.00	$(2^{-2}, 2^{4})$	**97.66 ± 0.54**
glass016vs2	59.25 ± 4.34	85.43	87.10 ± 1.39	87.00 ± 1.40	86.07 ± 0.46	86.74 ± 1.58	$(2^{16}, 2^{40})$	**88.25 ± 0.16**
glass4	96.18 ± 2.78	96.37	97.51 ± 2.49	97.51 ± 1.91	97.38 ± 0.19	**97.83 ± 0.00**	$(2^{8}, 2^{36})$	93.34 ± 0.07
haberman	**70.38 ± 4.29**	68.22	68.50 ± 0.15	68.36 ± 0.22	69.25 ± 0.97	69.32 ± 1.46	$(2^{8}, 2^{42})$	68.17 ± 0.73
iris0	54.85 ± 5.12	**100.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.0 ± 0.0**	$(2^{-10}, 2^{-18})$	**100.0 ± 0.0**
newthyroid1	99.70 ± 0.51	**100.00**	**100.00 ± 0.00**	**100.00 ± 1.11**	**100.00 ± 0.00**	**100.00 ± 0.00**	$(2^{-4}, 2^{24})$	**100.00 ± 0.00**
newthyroid2	99.60 ± 0.21	**100.00**	**100.00 ± 0.00**	99.98 ± 0.52	**100.00 ± 0.00**	**100.00 ± 0.00**	$(2^{8}, 2^{42})$	99.98 ± 0.04
pageblock13vs4	99.86 ± 0.2	98.00	99.97 ± 1.90	99.91 ± 0.13	100.00 ± 0.00	99.68 ± 0.28	$(2^{0}, 2^{12})$	**100.00 ± 0.00**
pima	79.91 ± 0.93	79.10	80.03 ± 1.14	**80.78 ± 0.48**	79.87 ± 0.42	80.55 ± 0.48	$(2^{0}, 2^{4})$	79.62 ± 0.05
segment0	**100.0 ± 0.0**	99.89	99.95 ± 0.00	99.91 ± 0.13	99.84 ± 0.00	99.64 ± 5.80	$(2^{-4}, 2^{2})$	99.93 ± 0.00
shuttleC0vsC4	80.00 ± 6.67	99.20	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	$(2^{2}, 2^{-8})$	**100.00 ± 0.00**
shuttleC2vsC4	81.91 ± 7.10	99.20	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	$(2^{4}, 2^{18})$	**100.00 ± 0.00**
vowel0	100.00 ± 0.00	100.00	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.0 ± 0.0**	$(2^{-10}, 2^{-18})$	**100.0 ± 0.00**
wisconsin	98.37 ± 0.31	98.15	99.05 ± 0.00	98.98 ± 0.14	98.72 ± 0.15	**99.08 ± 0.14**	$(2^{12}, 2^{42})$	**99.08 ± 0.09**
yeast05679vs4	87.97 ± 1.02	84.08	86.09 ± 0.30	86.25 ± 1.28	85.86 ± 0.84	**88.75 ± 1.07**	$(2^{-2}, 2^{6})$	85.63 ± 0.37
yeast1289vs7	74.91 ± 1.81	78.44	80.59 ± 0.84	**80.60 ± 0.53**	80.52 ± 0.08	80.59 ± 0.07	$(2^{6}, 2^{28})$	79.96 ± 0.66
yeast1458vs7	65.94 ± 2.14	70.72	72.77 ± 0.00	72.77 ± 1.08	72.98 ± 2.11	**73.08 ± 1.37**	$(2^{6}, 2^{18})$	71.49 ± 0.41
yeast1vs7	**86.53 ± 2.05**	82.89	82.90 ± 1.17	82.41 ± 0.76	84.06 ± 1.29	86.09 ± 0.58	$(2^{4}, 2^{16})$	83.67 ± 0.10
yeast2vs8	79.92 ± 2.45	84.42	84.32 ± 2.14	84.62 ± 1.29	84.08 ± 0.12	84.35 ± 0.00	$(2^{-2}, 2^{48})$	**85.56 ± 0.00**
Average	85.73 ± 2.19	90.89	91.94 ± 0.67	91.92 ± 0.58	91.79 ± 0.49	**92.08 ± 0.74**		91.58 ± 0.18

Figure 1a–c shows the boxplot diagram for the AUC results of the classifiers on various datasets shown in Tables 3–5 respectively. The boxplot creates a visual representation of the data to visualize the performance. It can be seen in Figure 1a,b that the proposed method RKWELM-UFS has the highest median value and smallest inter-quantile range, which shows that the RKWELM-UFS is performing better than MUEKL, USVM, KELM, WKELM, CCR-KELM, and WKSMOTE. It can be seen in Figure 1c that RKWELM-UFS is performing better than RUSBoost. Table 6 provides the *t*-test results and Table 7 provides the Wilcoxon Signed-rank test results on the AUC of various algorithms provided in Tables 3–5 for comparison. The results provided in Tables 6 and 7 suggest that the proposed RKWELM-UFS performs significantly better than MUEKL, USVM, KELM, WKELM, CCR-KELM, RUSBoost, and BWELM, and its performance is approximately similar to that of WKSMOTE, UBRKELM-MV, UBRKELM-SV, UBKELMMV, and UBKELM-SV in terms of AUC.

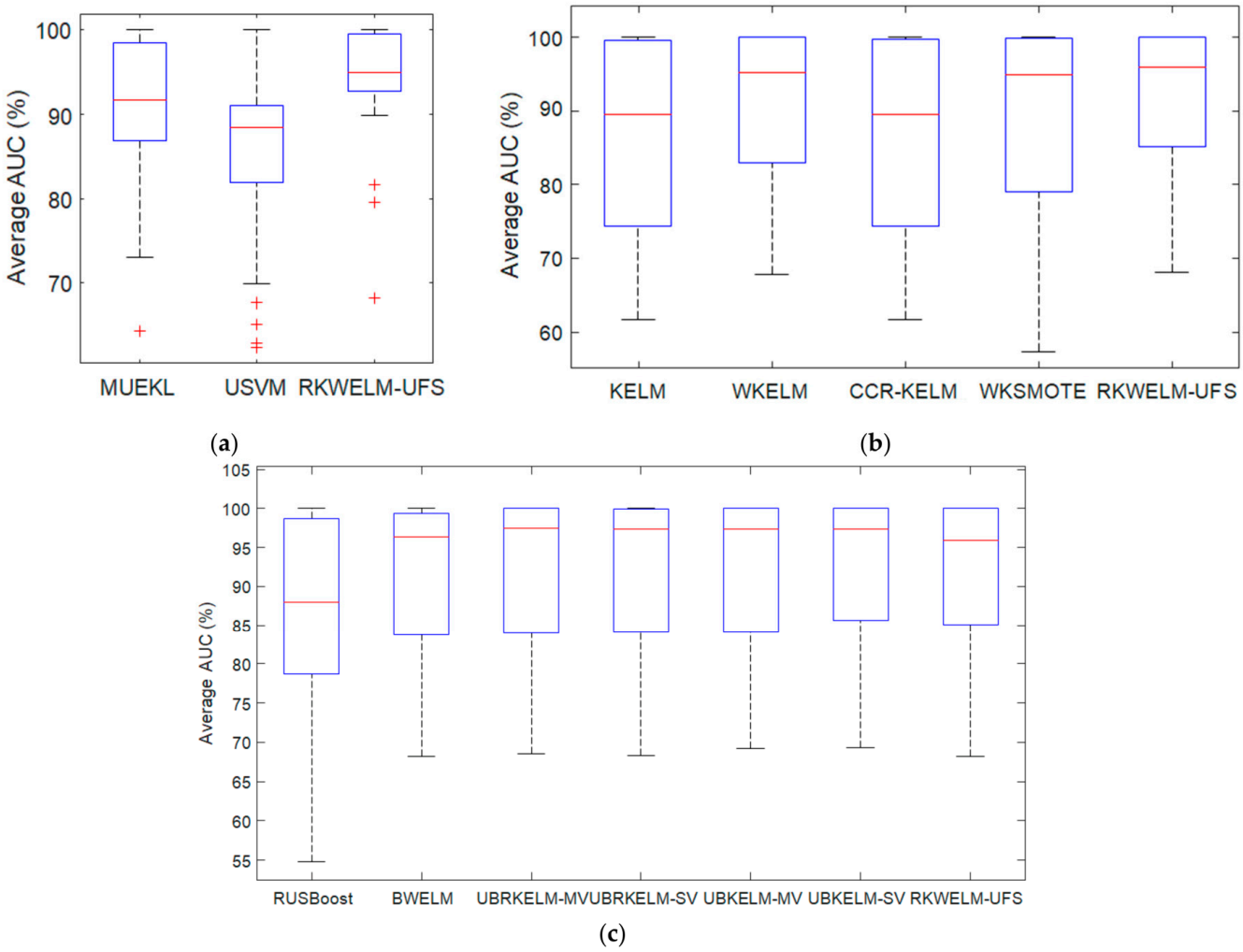

Figure 1. Boxplot diagrams; each box visually represents the performance in terms of average AUC of algorithms labeled on X axis. (**a**) Boxplot for results of Table 3. (**b**) Boxplot for results given in Table 4. (**c**) Boxplot for results given in Table 5.

Table 6. T-test results for performance comparison in terms of AUC between the methods given in Tables 3–5.

Methods Compared	Stats	p	H (0.05)
MUEKL vs. RKWELM-UFS	[−5.610820138186867; −2.152174147527417]	6.32×10^{-5}	1
USVM vs. RKWELM-UFS	[−11.440221923933274; −6.167915218923865]	8.34×10^{-8}	1
KELM vs. RKWELM-UFS	[−6.93534639982032; −2.78405360017968]	8.98×10^{-5}	1
WKELM vs. RKWELM-UFS	[−1.46246304953657; −0.301698855225338]	4.81×10^{-3}	1
CCR-KELM vs. RKWELM-UFS	[−6.88366104582675; −2.66145323988753]	1.33×10^{-4}	1
WKSMOTE vs. RKWELM-UFS	[−6.99623994510679; 1.56922089748774]	2.01×10^{-1}	0
RUSBoost vs. RKWELM-UFS	[−11.4335320049264; −0.266820376026017]	0.040906	1
BWELM vs. RKWELM-UFS	[−1.21328125091469; −0.165166368132924]	0.012527	1
UBRKELM-MV vs. RKWELM-UFS	[−0.169420432040612; 0.87763947965966]	0.17363	0
UBRKELM-SV vs. RKWELM-UFS	[−0.196630485917308; 0.872468581155405]	0.20219	0
UBKELM-MV vs. RKWELM-UFS	[−0.352563193428057; 0.776972717237582]	0.44236	0
UBKELM-SV vs. RKWELM-UFS	[−0.184407389564474; 1.18500738956447]	0.14312	0

Table 7. Wilcoxon Signed-rank test results for performance comparison in terms of AUC between the methods given in Tables 3–5.

Methods Compared	Zval	Signedrank	*p*-Value	H (0.05)
MUEKL vs. RKWELM-UFS	−4.62478463	12.00	3.75×10^{-6}	1
USVM vs. RKWELM-UFS	−5.086213249	0.00	3.65×10^{-7}	1
KELM vs. RKWELM-UFS	−3.621365173	0	2.93×10^{-4}	1
WKELM vs. RKWELM-UFS	0	10	2.62×10^{-3}	1
CCR-KELM vs. RKWELM-UFS	−3.621365173	0	2.93×10^{-4}	1
WKSMOTE vs. RKWELM-UFS	−1.763789403	45	7.78×10^{-2}	0
RUSBoost vs. RKWELM-UFS	−2.015964161	51	0.043803724	1
BWELM vs. RKWELM-UFS	−2.765775456	22	0.005678762	1
UBRKELM-MV vs. RKWELM-UFS	1.189301687	91	0.234320972	0
UBRKELM-SV vs. RKWELM-UFS	0.930757842	86	0.351978842	0
UBKELM-MV vs. RKWELM-UFS	0	73	0.488708496	0
UBKELM-SV vs. RKWELM-UFS	1.241010456	92	0.214601886	0

4.4.2. Performance Analysis in Terms of G-mean

Tables 8 and 9 provide the performance of the proposed RKWELM-UFS and other classification models in terms of the G-mean. The reported test G-mean of the proposed RKWELM-UFS given in Tables 8 and 9 is the averaged test g-mean obtained in 10 trials, using 5-fold cross-validation in each trial. Table 8 provides the performance of the proposed RKWELM-UFS and the existing single classifiers such as KELM, WKELM, CCR-KELM, and WKSMOTE on 21 datasets in terms of average G-mean, where the RKWELM outperforms the other classifiers on 16 datasets. Table 9 provides the performance of the proposed RKWELM-UFS and the existing ensemble of classifiers such as RUSBoost, BWELM, UBRKELM-MV, UBRKELMSV, UBKELM-MV, and UBKELM-SV on 21 datasets in terms of average Gmean, where the RKWELM outperforms the other classifiers on seven datasets.

Table 8. Performance Comparison of the proposed RKWELM-UFS with existing single classifiers in terms of average G-mean (std., KP, and C denote the standard deviation, Kernel width parameter, and regularization parameter, respectively).

Dataset	KELM Test Result%	WKELM Test Result%	CCR-KELM Test Result%	WKSMOTE Test Result%	(KP, C)	RKWELM-UFS TestResult% ± std.
abalone9vs18	72.71	89.76	76.56	91.94	$(2^{6}, 2^{26})$	**92.23 ± 0.57**
ecoli01vs5	88.36	91.34	88.36	88.00	$(2^{10}, 2^{42})$	**93.01 ± 0.11**
glass0123vs456	93.26	95.41	93.26	94.19	$(2^{-2}, 2^{4})$	**96.06 ± 0.55**
glass016vs2	63.20	83.59	81.36	79.00	$(2^{16}, 2^{40})$	**84.46 ± 0.50**
glass4	85.93	91.17	87.22	89.00	$(2^{8}, 2^{36})$	**91.49 ± 0.14**
haberman	57.23	**66.26**	59.71	65.21	$(2^{4}, 2^{12})$	66.02 ± 0.57
iris0	**100.00**	**100.00**	**100.00**	**100.00**	$(2^{-10}, 2^{-18})$	**100.00 ± 0.00**
newthyroid1	99.16	**99.72**	99.16	88.69	$(2^{-2}, 2^{12})$	99.44 ± 0.00
newthyroid2	99.44	**99.72**	99.44	90.72	$(2^{-6}, 2^{-18})$	99.44 ± 0.00
pageblock13vs4	97.89	98.07	97.84	97.38	$(2^{0}, 2^{16})$	**100.00 ± 0.00**
pima	71.16	75.58	73.61	74.00	$(2^{0}, 2^{4})$	**75.60 ± 0.19**
segment0	97.89	98.07	99.57	**100.00**	$(2^{-8}, 2^{-18})$	99.54 ± 0.00
shuttleC0vsC4	**100.00**	**100.00**	**100.00**	**100.00**	$(2^{2}, 2^{-8})$	**100.00 ± 0.00**
shuttleC2vsC4	94.14	**100.00**	**100.00**	**100.00**	$(2^{4}, 2^{18})$	**100.00 ± 0.00**
vowel0	**100.00**	**100.00**	**100.00**	**100.00**	$(2^{-10}, 2^{-18})$	**100.00 ± 0.00**
wisconsin	97.22	97.70	97.18	96.33	$(2^{12}, 2^{42})$	**97.89 ± 0.07**
yeast05679vs4	68.68	82.21	**82.24**	81.00	$(2^{-2}, 2^{6})$	81.03 ± 0.47

Table 8. *Cont.*

Dataset	KELM Test Result%	WKELM Test Result%	CCR-KELM Test Result%	WKSMOTE Test Result%	(KP, C)	RKWELM-UFS TestResult% ± std.
yeast1289vs7	60.97	71.41	59.28	69.83	$(2^{-2}, 2^{0})$	**73.35 ± 0.05**
yeast1458vs7	59.89	**69.32**	66.24	67.00	$(2^{-4}, 2^{6})$	67.54 ± 0.09
yeast1vs7	64.48	77.72	68.32	76.00	$(2^{2}, 2^{2})$	**77.77 ± 0.15**
yeast2vs8	77.24	77.89	78.91	80.00	$(2^{0}, 2^{26})$	**81.36 ± 1.42**
Average	83.28	88.81	86.11	87.06		**89.74 ± 0.17**

Table 9. Performance Comparison of the proposed RKWELM-UFS with existing ensemble of classifiers in terms of average G-mean (std., KP, and C denote the standard deviation, Kernel width parameter, and regularization parameter, respectively).

Dataset	RUSBoost TstR ± std.	BWELM TstR	UBRKELM MV TstR ± std.	UBRKELM SV TstR ± std.	UBKELM MV TstR ± std.	UBKELM SV TstR ± std.	(KP, C)	RKWELM UFS TstR ± std.
abalone9vs18	86.40 ± 1.33	90.12	92.28 ± 0.00	**92.30 ± 0.00**	91.53 ± 0.96	91.07 ± 3.45	$(2^{6}, 2^{26})$	92.23 ± 0.57
ecoli01vs5	88.92 ± 1.55	89.36	93.53 ± 0.00	93.09 ± 0.09	93.63 ± 0.43	**94.02 ± 1.07**	$(2^{10}, 2^{42})$	93.01 ± 0.11
glass0123vs456	93.74 ± 0.84	94.21	95.67 ± 0.68	95.91 ± 0.51	95.24 ± 0.90	95.45 ± 0.25	$(2^{-2}, 2^{4})$	**96.06 ± 0.55**
glass016vs2	52.46 ± 3.04	84.21	84.26 ± 0.50	84.42 ± 0.35	**84.48 ± 0.43**	83.89 ± 1.29	$(2^{16}, 2^{40})$	84.46 ± 0.50
glass4	87.31 ± 2.82	90.30	91.69 ± 2.10	91.57 ± 2.86	**92.91 ± 2.82**	92.86 ± 3.30	$(2^{8}, 2^{36})$	91.49 ± 0.14
haberman	53.36 ± 7.21	65.14	66.34 ± 0.13	**70.20 ± 4.23**	66.70 ± 0.88	66.49 ± 1.50	$(2^{4}, 2^{12})$	66.02 ± 0.57
iris0	19.85 ± 10.38	**100.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	$(2^{-10}, 2^{-18})$	**100.0 ± 0.0**
newthyroid1	98.05 ± 0.95	**100.00**	99.55 ± 0.20	99.61 ± 0.14	99.29 ± 0.49	99.47 ± 0.13	$(2^{-2}, 2^{12})$	99.44 ± 0.00
newthyroid2	96.94 ± 0.91	**99.72**	99.44 ± 0.13	99.44 ± 0.13	99.13 ± 0.00	99.30 ± 0.08	$(2^{-6}, 2^{-18})$	99.44 ± 0.00
pageblock13vs4	97.96 ± 1.21	99.89	99.41 ± 0.12	99.91 ± 0.13	**100.00 ± 0.00**	**100.00 ± 0.00**	$(2^{0}, 2^{16})$	**100.00 ± 0.00**
pima	70.34 ± 1.45	75.48	76.11 ± 0.21	**76.22 ± 0.22**	75.76 ± 0.31	75.84 ± 0.34	$(2^{0}, 2^{4})$	75.60 ± 0.19
segment0	**99.99 ± 0.00**	99.89	99.80 ± 0.13	99.68 ± 0.28	99.63 ± 1.10	99.64 ± 5.80	$(2^{-8}, 2^{-18})$	99.54 ± 0.00
shuttleC0vsC4	60.00 ± 13.33	**100.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	$(2^{2}, 2^{-8})$	**100.00 ± 0.00**
shuttleC2vsC4	68.50 ± 15.89	**100.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	$(2^{4}, 2^{18})$	**100.00 ± 0.00**
vowel0	100.00 ± 0	**100.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	$(2^{-10}, 2^{-18})$	**100.00 ± 0.00**
wisconsin	95.46 ± 0.77	97.18	97.81 ± 0.00	97.81 ± 0.00	97.72 ± 0.20	97.79 ± 0.18	$(2^{12}, 2^{42})$	**97.89 ± 0.07**
yeast05679vs4	77.55 ± 1.63	80.96	81.82 ± 1.98	82.62 ± 0.09	82.24 ± 0.33	**83.45 ± 2.29**	$(2^{-2}, 2^{6})$	81.03 ± 0.47
yeast1289vs7	67.83 ± 2.96	72.67	**75.54 ± 0.84**	75.27 ± 0.12	74.28 ± 1.05	74.73 ± 1.72	$(2^{-2}, 2^{0})$	73.35 ± 0.05
yeast1458vs7	59.59 ± 3.43	69.87	69.54 ± 1.48	69.54 ± 1.74	**71.24 ± 1.69**	70.15 ± 1.31	$(2^{-4}, 2^{6})$	67.54 ± 0.09
yeast1vs7	73.49 ± 1.79	77.72	**78.90 ± 7.44**	78.41 ± 0.76	77.73 ± 0.00	77.90 ± 1.54	$(2^{2}, 2^{2})$	77.77 ± 0.15
yeast2vs8	72.16 ± 1.83	78.35	80.77 ± 2.42	80.10 ± 0.22	80.05 ± 2.44	**81.69 ± 1.51**	$(2^{0}, 2^{26})$	81.36 ± 1.42
Average	77.39 ± 3.59	89.34	90.08 ± 0.80	**90.30 ± 0.58**	90.08 ± 0.58	90.10 ± 1.21		89.74 ± 0.17

Figure 2a,b shows the boxplot diagram for the G-mean results of the classifiers on various datasets shown in Tables 8 and 9, respectively. It can be seen in Figure 2a that the proposed RKWELM-UFS has the highest median value and smallest inter-quantile range, which shows that the RKWELM-UFS is performing better than KELM, WKELM, CCR-KELM, and WKSMOTE in terms of the G-mean. It can be seen in Figure 2b that RKWELM-UFS is performing better than RUSBoost and BWELM in terms of the G-mean. Table 10 provides the *t*-test results and Table 11 provides the Wilcoxon signed-rank test results on the G-mean of various algorithms provided in Tables 8 and 9 for comparison. The results provided in Tables 10 and 11 suggest that the proposed RKWELM-UFS performs significantly better than KELM, CCR-KELM, WKSMOTE, and RUSBoost, and performs approximately similarly to WKELM, BWELM, UBRKELM-MV, UBRKELM-SV, UBKELM-MV, UBKELM-SV in terms of the G-mean.

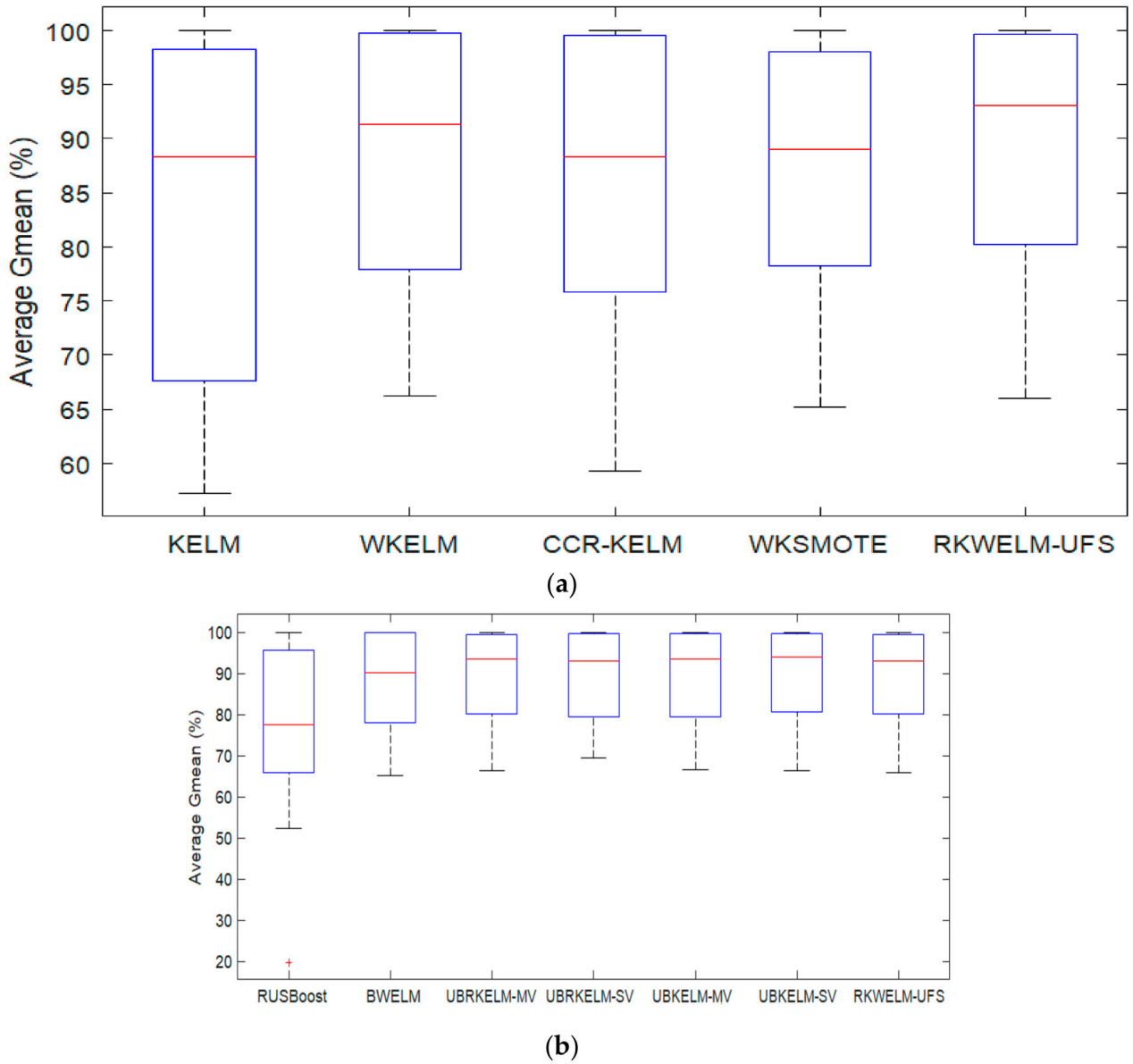

Figure 2. Boxplot diagrams; each box visually represents the performance in terms of average G-mean of algorithms labelled on X axis. (**a**) Boxplot for G-mean results given in Table 8. (**b**) Boxplot for G-mean results given in Table 9.

Table 10. T-test results for performance comparison in terms of G-mean between the methods given in Tables 8 and 9.

Methods Compared	Stats	p	H (0.05)
KELM vs. RKWELM-UFS	[−8.97586793133104; −3.15547492581182]	3.12×10^{-4}	1
WKELM vs. RKWELM-UFS	[−1.10027216529072; 0.0251197843383338]	6.01×10^{-2}	0
CCR-KELM vs. RKWELM-UFS	[−5.34275272694909; −1.13049489209853]	4.44×10^{-3}	1
WKSMOTE vs. RKWELM-UFS	[−3.63784233634963; −0.927786235078946]	2.18×10^{-3}	1
RUSBoost vs. RKWELM-UFS	[−20.961933935114; −3.45036130298126]	0.0086978	1
BWELM vs. RKWELM-UFS	[−1.12033850136527; 0.0575670727938417]	7.45×10^{-2}	0
UBRKELM-MV vs. RKWELM-UFS	[−0.0326448040855254; 0.626063851704572]	0.074868	0
UBRKELM-SV vs. RKWELM-UFS	[−0.0440133837744108; 0.984099098060123]	0.070939	0
UBKELM-MV vs. RKWELM-UFS	[−0.207557995239237; 0.715262757143997]	0.26467	0
UBKELM-SV vs. RKWELM-UFS	[−0.0647504586179029; 0.780074268141712]	0.092621	0

Table 11. Wilcoxon signed-rank test results for performance comparison in terms of G-mean between the methods given in Tables 8 and 9.

Methods Compared	Zval	Signed Rank	*p*-Value	H (0.05)
KELM vs. RKWELM-UFS	−3.723555406	0	1.96×10^{-4}	1
WKELM vs. RKWELM-UFS	−1.822772421	38	6.83×10^{-2}	0
CCR-KELM vs. RKWELM-UFS	−3.289998425	7	1.00×10^{-3}	1
WKSMOTE vs. RKWELM-UFS	−3.479350852	3	5.03×10^{-4}	1
RUSBoost vs. RKWELM-UFS	−3.882597643	1	1.03×10^{-4}	1
BWELM vs. RKWELM-UFS	−1.917193327	36	5.52×10^{-2}	0
UBRKELM-MV vs. RKWELM-UFS	1.585826579	110	0.112778655	0
UBRKELM-SV vs. RKWELM-UFS	1.917193327	117	0.055213376	0
UBKELM-MV vs. RKWELM-UFS	0.723922766	82	0.469113152	0
UBKELM-SV vs. RKWELM-UFS	1.525663664	97.5	0.127093648	0

5. Conclusions and Future Work

The use of additional data for training along with the original training data has been employed in many approaches. The Universum data are used to add prior knowledge about the distribution of data in the classification model. Various ELM-based classification models have been suggested to handle the class imbalance problem, but none of these models use prior knowledge. The proposed RKWELM-UFS is the first attempt that employs Universum data to enhance the performance of the RKWELM classifier. This work generates the Universum samples in the feature space using the kernel trick. The reason behind the creation of the Universum instances in the feature space is that the mapping of input data to the feature space is not conformal. The proposed work is evaluated on 44 benchmark datasets with an imbalance ratio between 0:45 to 43:80 and a number of instances between 129 to 2308. The proposed method is compared with 10 state-of-the-art methods used for class-imbalanced dataset classification. G-mean and AUC are used as metrics to evaluate the performance of the proposed method. The paper also incorporates statistical tests to verify the significant performance difference between the proposed and compared methods.

In Universum data-based learning, it has been observed that the efficiency of such classifiers depends on the quality and volume of Universum data created. The methodology of choosing or creating the appropriate Universum samples should be the subject of further research. In the proposed work, the Universum samples are created between randomly selected pairs of majority and minority class samples. In the future, some strategic concepts can be used to select the majority and minority samples instead of random selection. In the future, Universum data can be incorporated in other ELM-based classification models to enhance their learning capability on class imbalance problems. The future work also includes the development of a multi-class variant of the proposed RKWELM-UFS.

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

Funding: The work received no funding from any organization, institute, or person.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Schaefer, G.; Nakashima, T. Strategies for addressing class imbalance in ensemble classification of thermography breast cancer features. In Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC), Sendai, Japan, 25–28 May 2015; pp. 2362–2367.
2. Sadewo, W.; Rustam, Z.; Hamidah, H.; Chusmarsyah, A.R. Pancreatic Cancer Early Detection Using Twin Support Vector Machine Based on Kernel. *Symmetry* **2020**, *12*, 667. [CrossRef]
3. Hao, W.; Liu, F. Imbalanced Data Fault Diagnosis Based on an Evolutionary Online Sequential Extreme Learning Machine. *Symmetry* **2020**, *12*, 1204. [CrossRef]
4. Mulyanto, M.; Faisal, M.; Prakosa, S.W.; Leu, J.-S. Effectiveness of Focal Loss for Minority Classification in Network Intrusion Detection Systems. *Symmetry* **2020**, *13*, 4. [CrossRef]
5. Tahvili, S.; Hatvani, L.; Ramentol, E.; Pimentel, R.; Afzal, W.; Herrera, F. A novel methodology to classify test cases using natural language processing and imbalanced learning. *Eng. Appl. Artif. Intell.* **2020**, *95*, 103878. [CrossRef]
6. Furundzic, D.; Stankovic, S.; Jovicic, S.; Punisic, S.; Subotic, M. Distance based resampling of imbalanced classes: With an application example of speech quality assessment. *Eng. Appl. Artif. Intell.* **2017**, *64*, 440–461. [CrossRef]
7. Mariani, V.C.; Och, S.H.; dos Santos Coelho, L.; Domingues, E. Pressure prediction of a spark ignition single cylinder engine using optimized extreme learning machine models. *Appl. Energy* **2019**, *249*, 204–221. [CrossRef]
8. He, H.; Garcia, E.A. Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.* **2009**, *21*, 1263–1284.
9. Weston, J.; Collobert, R.; Sinz, F.; Bottou, L.; Vapnik, V. Inference with the universum. In Proceedings of the 23rd International Conference on Machine Learning, New York, NY, USA, 25 June 2006; pp. 1009–1016.
10. Qi, Z.; Tian, Y.; Shi, Y.J. Twin support vector machine with Universum data. *Neural Netw.* **2012**, *36*, 112–119. [CrossRef]
11. Dhar, S.; Cherkassky, V. Cost-sensitive Universum-svm. In Proceedings of the 2012 11th International Conference on Machine Learning and Applications, Boca Raton, FL, USA, 12–15 December 2012; pp. 220–225.
12. Richhariya, B.; Tanveer, M.J. EEG signal classification using Universum support vector machine. *Expert Syst. Appl.* **2018**, *106*, 169–182. [CrossRef]
13. Qi, Z.; Tian, Y.; Shi, Y. A nonparallel support vector machine for a classification problem with Universum learning. *J. Comput. Appl. Math.* **2014**, *263*, 288–298. [CrossRef]
14. Zhao, J.; Xu, Y.; Fujita, H.J. An improved non-parallel Universum support vector machine and its safe sample screening rule. *Knowl. Based Syst.* **2019**, *170*, 79–88. [CrossRef]
15. Tencer, L.; Reznáková, M.; Cheriet, M.J. Ufuzzy: Fuzzy models with Universum. *Appl. Soft Comput.* **2017**, *59*, 1–18. [CrossRef]
16. Wang, Z.; Hong, S.; Yao, L.; Li, D.; Du, W.; Zhang, J. Multiple universum empirical kernel learning. *Eng. Appl. Artif. Intell.* **2020**, *89*, 103461. [CrossRef]
17. Huang, G.-B.; Zhu, Q.-Y.; Siew, C.-K. Extreme learning machine: Theory and applications. *Neurocomputing* **2006**, *70*, 489–501. [CrossRef]
18. Zong, W.; Huang, G.-B.; Chen, Y.J. Weighted extreme learning machine for imbalance learning. *Neurocomputing* **2013**, *101*, 229–242. [CrossRef]
19. Xiao, W.; Zhang, J.; Li, Y.; Zhang, S.; Yang, W. Class-specific cost regulation extreme learning machine for imbalanced classification. *Neurocomputing* **2017**, *261*, 70–82. [CrossRef]
20. Raghuwanshi, B.S.; Shukla, S. Class-specific kernelized extreme learning machine for binary class imbalance learning. *Appl. Soft Comput.* **2018**, *73*, 1026–1038. [CrossRef]
21. Raghuwanshi, B.S.; Shukla, S. Underbagging based reduced kernelized weighted extreme learning machine for class imbalance learning. *Eng. Appl. Artif. Intell.* **2018**, *74*, 252–270. [CrossRef]
22. Raghuwanshi, B.S.; Shukla, S. Class imbalance learning using UnderBagging based kernelized extreme learning machine. *Neurocomputing* **2019**, *329*, 172–187. [CrossRef]
23. Mathew, J.; Pang, C.K.; Luo, M.; Leong, W.H. Classification of imbalanced data by oversampling in kernel space of support vector machines. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *29*, 4065–4076. [CrossRef]
24. Chen, S.; Zhang, C. Selecting informative Universum sample for semi-supervised learning. In Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence, Pasadena, CA, USA, 11–17 July 2009.
25. Zhao, J.; Xu, Y.J. A safe sample screening rule for Universum support vector machines. *Knowl. Based Syst.* **2017**, *138*, 46–57. [CrossRef]
26. Cherkassky, V.; Dai, W. Empirical study of the Universum SVM learning for high-dimensional data. In Proceedings of the International Conference on Artificial Neural Networks, Limassol, Cyprus, 14–17 September 2009; pp. 932–941.
27. Hamidzadeh, J.; Kashefi, N.; Moradi, M. Combined weighted multi-objective optimizer for instance reduction in two-class imbalanced data problem. *Eng. Appl. Artif. Intell.* **2020**, *90*, 103500. [CrossRef]
28. Lin, W.-C.; Tsai, C.-F.; Hu, Y.-H.; Jhang, J.-S. Clustering-based undersampling in class-imbalanced data. *Inf. Sci.* **2017**, *409*, 17–26. [CrossRef]
29. Ofek, N.; Rokach, L.; Stern, R.; Shabtai, A. Fast-CBUS: A fast clustering-based undersampling method for addressing the class imbalance problem. *Neurocomputing* **2017**, *243*, 88–102. [CrossRef]
30. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]

31. Zhu, T.; Lin, Y.; Liu, Y. Synthetic minority oversampling technique for multiclass imbalance problems. *Pattern Recognit.* **2017**, *72*, 327–340. [CrossRef]
32. Agrawal, A.; Viktor, H.L.; Paquet, E. SCUT: Multi-class imbalanced data classification using SMOTE and cluster-based undersampling. In Proceedings of the 2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K), Lisbon, Portugal, 12–14 November 2015; pp. 226–234.
33. Wang, Z.; Chen, L.; Fan, Q.; Li, D.; Gao, D. Multiple Random Empirical Kernel Learning with Margin Reinforcement for imbalance problems. *Eng. Appl. Artif. Intell.* **2020**, *90*, 103535. [CrossRef]
34. Raghuwanshi, B.S.; Shukla, S. Class-specific extreme learning machine for handling binary class imbalance problem. *Neural Netw.* **2018**, *105*, 206–217. [CrossRef]
35. Guo, W.; Wang, Z.; Hong, S.; Li, D.; Yang, H.; Du, W. Multi-kernel Support Vector Data Description with boundary information. *Eng. Appl. Artif. Intell.* **2021**, *102*, 104254. [CrossRef]
36. Seiffert, C.; Khoshgoftaar, T.M.; Van Hulse, J.; Napolitano, A. RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Trans. Syst. Man Cybern.-Part A Syst. Hum.* **2009**, *40*, 185–197. [CrossRef]
37. Haixiang, G.; Yijing, L.; Yanan, L.; Xiao, L.; Jinling, L. BPSO-Adaboost-KNN ensemble learning algorithm for multi-class imbalanced data classification. *Eng. Appl. Artif. Intell.* **2016**, *49*, 176–193. [CrossRef]
38. Shen, C.; Wang, P.; Shen, F.; Wang, H. UBoost: Boosting with theUniversum. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *34*, 825–832. [CrossRef]
39. Freund, Y.; Schapire, R.; Abe, N.J. A short introduction to boosting. *J. Jpn. Soc. Artif. Intell.* **1999**, *14*, 1612.
40. Zhang, Y.; Liu, B.; Cai, J.; Zhang, S. Ensemble weighted extreme learning machine for imbalanced data classification based on differential evolution. *Neural Comput. Appl.* **2017**, *28*, 259–267. [CrossRef]
41. Li, K.; Kong, X.; Lu, Z.; Wenyin, L.; Yin, J. Boosting weighted ELM for imbalanced learning. *Neurocomputing* **2014**, *128*, 15–21. [CrossRef]
42. Huang, G.B.; Zhou, H.M.; Ding, X.J.; Zhang, R. Extreme Learning Machine for Regression and Multiclass Classification. *IEEE Trans. Syst. Man Cybern. Part B* **2012**, *42*, 513–529. [CrossRef] [PubMed]
43. Deng, W.Y.; Ong, Y.S.; Zheng, Q.H. A Fast Reduced Kernel Extreme Learning Machine. *Neural Netw.* **2016**, *76*, 29–38. [CrossRef] [PubMed]
44. Alcala-Fdez, J.; Fernandez, A.; Luengo, J.; Derrac, J.; Garcia, S.; Sanchez, L.; Herrera, F. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. *J. Mult.-Valued Log. Soft Comput.* **2011**, *17*, 255–287.
45. Alcala-Fdez, J.; Sanchez, L.; Garcia, S.; del Jesus, M.J.; Ventura, S.; Garrell, J.M.; Otero, J.; Romero, C.; Bacardit, J.; Rivas, V.M.; et al. KEEL: A software tool to assess evolutionary algorithms for data mining problems. *Soft Comput.* **2009**, *13*, 307–318. [CrossRef]
46. Zeng, Y.J.; Xu, X.; Shen, D.Y.; Fang, Y.Q.; Xiao, Z.P. Traffic Sign Recognition Using Kernel Extreme Learning Machines with Deep Perceptual Features. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 1647–1653. [CrossRef]