mathematics

MDPI

*Article*

# An Algebraic Approach to Clustering and Classification with Support Vector Machines

**Güvenç Arslan [1], Uğur Madran [2,*] and Duygu Soyoğlu [2]**

[1] Department of Statistics, Kırıkkale University, Kırıkkale 71450, Turkey; guvenc.arslan@kku.edu.tr
[2] College of Engineering and Technology, American University of the Middle East, Egaila 54200, Kuwait; duygu.soyoglu@aum.edu.kw
* Correspondence: ugur.madran@aum.edu.kw

**Abstract:** In this note, we propose a novel classification approach by introducing a new clustering method, which is used as an intermediate step to discover the structure of a data set. The proposed clustering algorithm uses similarities and the concept of a clique to obtain clusters, which can be used with different strategies for classification. This approach also reduces the size of the training data set. In this study, we apply support vector machines (SVMs) after obtaining clusters with the proposed clustering algorithm. The proposed clustering algorithm is applied with different strategies for applying SVMs. The results for several real data sets show that the performance is comparable with the standard SVM while reducing the size of the training data set and also the number of support vectors.

**Keywords:** clique; algebraic statistics; machine learning; clustering; classification; support vector machine

**MSC:** 62H30; 68T05



**Citation:** Arslan G.; Madran U.; Soyoğlu D. An Algebraic Approach to Clustering and Classification with Support Vector Machines. *Mathematics* **2022**, *10*, 128. https://doi.org/10.3390/math10010128

Academic Editor: Victor Leiva

Received: 25 November 2021
Accepted: 30 December 2021
Published: 1 January 2022



## 1. Introduction

In order to meet people's demands, such as searching from the internet, taking high-quality pictures and videos, undergoing medical tests, widespread emails, producing multimedia contents, etc., there is a tremendous and continuous advancement in technology regarding data. Such interests require large digital storage with high-dimensional data sets. The data sets include both meaningful and irrelevant data, which is why ''data analysis'' and ''data classification'' topics are gaining more attention in recent years. Data analysis, providing predictions based on the training data, gave birth to machine learning.

Machine learning (ML) is a mega-trend in various fields and can be seen as a collection of computer algorithms that make life easier by predicting outcomes. Fundamentally, ML is based on algorithms (so-called "machines") to automatically learn from observations (so-called "training data") and to use the results for future decisions without explicit programming. ML algorithms have applications in almost all areas.

ML is generally categorized as either *unsupervised learning* (e.g., clustering), *supervised learning* (e.g., classification), or *reinforcement learning* (e.g., dynamic programming). In recent years, a new approach has been introduced: *semi-supervised learning*, which is a mixture of supervised learning and unsupervised learning. For semi-supervised learning, similarity-based approaches have also gained significant popularity in recent years (see, [1–4]).

In this paper, we propose a clustering algorithm that uses similarities between examples by associating weights to each attribute (according to their ability to separate classes) to reduce the size of the data set by combining the data as they form a so-called "clique". The cliques have been heavily used in random matrix theory, and recently, this concept is being used in clustering algorithms (see, for example, [5–7]). More detail is given in Section 2.

Another important topic in ML is classification. Classification is a task for the predictive modeling problem. The aim of classification is to assign a class label for a given input data. There are several types of classification algorithms. Some well-known and widely used classification algorithms are Naive Bayes, Decision Trees, k-Nearest Neighbours, Logistic Regression, and Support Vector Machines.

Support vector machines (SVMs) are state-of-the-art classification methods used in various fields ([8–10]). There are many studies on extensions and improvements for SVMs. In some of these studies, SVMs are used with different clustering algorithms. For example, Wang et al., in [11], proposed KMSVM, which combines k-means clustering with SVMs. Chen et al., in [12], extend a least-squares twin support vector machine to a multiple-birth least-squares support vector machine for multi-class classification. Cheng et al., in [13], study local support vector machines for classification with nonlinear decision surfaces. Arslan et al. proposed a new clustering algorithm to discover the structure of the data set and then used these clusters with SVM in [2]. Another recent study [14] used the CURE clustering algorithm with SVMs. All these studies claim to obtain comparable performance with reduced training data set size and less support vectors. A brief introduction to SVMs is given in Section 3.

Motivated by these recent studies, we propose a new clustering algorithm based on cliques, named *Clique Clustering Algorithm* (CC-Algorithm). In Section 4, the CC-Algorithm is given. The algorithm is also demonstrated by an example in detail.

The main idea of using this clustering algorithm is to reduce the number of training data in the data set that will be used with SVMs (Section 5). There are three different algorithms proposed in this study: *Centers of Cliques-SVM* (CC-SVM), *Homogeneous Cliques Removed-SVM* (HOM-R-SVM), and *Heterogeneous Cliques Removed-SVM* (HET-R-SVM). Each proposed algorithm demonstrates different characteristics in terms of accuracy, performance, and data reduction ability.

## 2. Cliques and Clustering

As data sets become larger in size and complexity, data analysis has become the hit research topic in recent years. One of the main goals of clustering is to understand and analyze the data by splitting it into groups of similar items. Each group is named a cluster, and the method is focused on covering the data with a fewer number of clusters (details can be found from [15]). Although there is no significant definition for clusters since each method has clusters with its own shape, size, and density, it is extensively used in applications, such as web analysis [16], marketing [17], computational biology [18], data mining [19], machine learning [20], and many others.

Generating clusters is generally regarded as unsupervised learning because clustering does not use category labels, whereas classification is regarded as supervised learning. A new approach as a combination of unsupervised and supervised learning showed up in 2006 by Chapelle et al. [21], which is called semi-supervised learning. Most hybrid learning methods use various distance concepts for similarity functions. Similarity functions are generally defined by using Euclidean distance, Chebyshev distance, Jaccard distance, Mahalanobis distance, etc. Different distance functions result in different algorithms. For example, some graph-based approaches can be found in articles [4,22,23]. In [4], the data set represents a weighted graph that each data element is shown as a vertex and linking the vertices by the edges using a similarity value. However, Chen et al., in [22], proposed a partially observed unweighted graph. Ames et al. [6], on the other hand, considered the $k$-disjoint-clique optimization problem as a way to express the clustering problem with graphs and cliques. They claimed to solve this NP-hard optimization problem by using convex optimization.

A **clique** is a complete subgraph of an undirected graph, i.e., there is an edge between any pair of vertices. It was first used by Luce and Perry (1949) [24] in social networks and has many different applications in many areas other than graph theory.

In this study, we consider yet another approach for clustering, which is aimed at providing information from the data set that can be used with various classification strategies. We use cliques to obtain clusters. In our approach, the cliques and, therefore, the clusters are obtained with a heuristic algorithm. Instead of solving a problem like the densest $k$-disjoint-clique optimization problem, the proposed Algorithm 1 obtains dense cliques in a natural way by considering similarities between examples and by using a threshold value.

To achieve this, we subdivide the data into a number of disjoint cliques that maximize the sum of the weights of the edges inside of it. At first sight, the clique idea may look similar to the one in [6] where Euclidean distance is used for the weights of the edges. However, in this current study, the edge weight is calculated by using a similarity function that is totally different from the distance function. To the best of our knowledge, this is the first algorithm using cliques without using any distance function.

**Definition 1.** *The similarity value between two examples $x_i, x_j \in U$ with respect to an attribute $a \in A$, where $A$ is the set of attributes and $U$ is a finite set of objects, is defined as*

$$\mathrm{sim}_a(x_i, x_j) = 1 - \frac{|f(x_i, a) - f(x_j, a)|}{\max(a) - \min(a)}$$

*for a numerical attribute $a$, where $\max(a) = \max_{x_i \in U} f(x_i, a)$, $\min(a) = \min_{x_i \in U} f(x_i, a)$, and $f(x, a)$ is a total function for each $a \in A$ and $x \in U$ and named the information function.*

**Remark 1.** *$f(x_i, a)$ denotes the value of attribute $a$ for the example $x_i$ of the data set $U$.*

The similarity value defined above calculates the similarity between two vertices for a single attribute. In order to consider the total similarity with respect to a set of attributes, we use the similarity definition below, which is obtained by summing individual similarities using special weights.

**Definition 2.** *The similarity value between two examples $x_i, x_j \in U$ with respect to an attribute set $B \subseteq A$ is defined as*

$$sim_B(x_i, x_j) = \sum_{a \in B} w_a sim_a(x_i, x_j) \tag{1}$$

*where $w_a$ corresponds to the weight for an attribute $a \in B$.*

As mentioned in the introduction, weights for attributes are calculated according to their performances for separating the classes. The detailed calculation of the weight for an attribute $a$ is given as:

$$w_a = \frac{\sum_{i=1}^{n} s(B_i(a))}{s(U)} \tag{2}$$

where $s(X)$ denotes the number of elements in the set $X$, and $B_i$'s are defined as below:

$$B_i(a) = A_i(a) \setminus \bigcup_{j=1(i \neq j)}^{n} A_j(a),$$
$$A_i(a) = \{x \in U | \min(C_i(a)) \leq f(x, a) \leq \max(C_i(a))\},$$

where $C_i(a)$ is the set of values for an attribute $a$ belonging to class $i$, $1 \leq i \leq n$, and $n$ is the number of classes in a data set. Equations (1) and (2) were introduced in [25], and we direct the reader to there for more details.

## 3. Support Vector Machines and Classification

Support vector machines (SVMs) were introduced by Vapnik and Chervenenkis in 1974 (see [26]). SVMs are one of the most popular methods used in ML (indeed, for supervised ML) with the aim of classifying given data by using so-called "training data". For an introduction to the general theory of SVMs, we direct the reader to the sources [8,10,27–29] and to the papers [30–33] for recent developments, to name a few.

Basically, SVM is an algorithm for finding a separating hyperplane between two classes so that the margin between two classes is maximized. Suppose that the data set is given as $\{(\boldsymbol{x_i}, y_i) \mid \boldsymbol{x_i} \in \boldsymbol{R}^n, y_i = \pm 1, \text{ for } i = 1, 2, \cdots, \ell\}$, where $y_i$ denotes either of two possible classes $\{-1, 1\}$, for simplicity. In order to find the optimal hyperplane whose normal vector is $\boldsymbol{w}$ and given by Equation $\boldsymbol{x} \cdot \boldsymbol{w} + b = 0$, one should consider the constraints:

$$\begin{aligned} \boldsymbol{x_i} \cdot \boldsymbol{w} + b &\geq 1 \quad \text{if } y_i = 1, \\ \boldsymbol{x_i} \cdot \boldsymbol{w} + b &\leq -1 \quad \text{if } y_i = -1, \end{aligned}$$

which can be combined together as

$$y_i(\boldsymbol{x_i} \cdot \boldsymbol{w} + b) - 1 \geq 0 \quad \text{for } i = 1, 2, \cdots, \ell.$$

By requiring (if necessary, after rescaling $\boldsymbol{w}$ and $b$) that there are some points satisfying the two inequalities as an equality (such points are called "support vectors"), the margin between two classes turns out to be $2/||\boldsymbol{w}||$. Hence, in order to find the hyperplane providing the maximum margin between two classes, it is enough to minimize $||\boldsymbol{w}||$ or equivalently and more simply $||\boldsymbol{w}||^2$. By using Lagrange multipliers, $\lambda_i$ for $i = 1, \cdots, \ell$, one obtains the Lagrangian

$$F(\boldsymbol{w}, b, \lambda_1, \cdots, \lambda_\ell) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{\ell} \lambda_i y_i (\boldsymbol{x_i} \cdot \boldsymbol{w} + b) + \sum_{i=1}^{\ell} \lambda_i.$$

We direct the reader to [8] for a comprehensive and complete account of the remaining convex optimization.

If there is no feasible solution to this optimization problem, i.e., when the data are not separable by a hyperplane, then by introducing extra positive slack variables, $\xi_i$, representing the error (when $\xi_i > 1$), the above constraints can be rewritten as:

$$\begin{aligned} \boldsymbol{x_i} \cdot \boldsymbol{w} + b &\geq 1 - \xi_i \quad \text{if } y_i = 1, \\ \boldsymbol{x_i} \cdot \boldsymbol{w} + b &\leq -1 + \xi_i \quad \text{if } y_i = -1, \\ \xi_i &\geq 0. \end{aligned}$$

In this case, the objective function to be minimized is also changed from

$$\frac{||\boldsymbol{w}||^2}{2}$$

to

$$\frac{||\boldsymbol{w}||^2}{2} + C \sum_{i=1}^{\ell} \xi_i,$$

where $C$ is a cost parameter for the sum of error terms.

For data sets where classes are non-linearly separated (i.e., when they are separated by some hypersurface), one can use the kernel trick and transform the data set into a higher dimensional Euclidean space by using an appropriate kernel function $K(\boldsymbol{x_i}, \boldsymbol{x_j})$. The most common used kernel functions for real-life data in pattern recognition are listed in Table 1 (see [34] for details):

**Table 1.** Common kernel functions.

| Kernel Name | Kernel Function |
|---|---|
| Linear ($k = 1$): | $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \boldsymbol{x_i} \cdot \boldsymbol{x_j}$ |
| Gaussian Radial Basis ($k = 2$): | $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = e^{-\|\|x_i - x_j\|\|^2/2\sigma^2}$ |
| Polynomial ($k = 3$): | $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = (\gamma \boldsymbol{x_i} \cdot \boldsymbol{x_j} + \delta)^p$ |
| Sigmoid ($k = 4$): | $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \tanh(\gamma \boldsymbol{x_i} \cdot \boldsymbol{x_j} + \delta)$ |

In this study, all kernel functions above are tested for each data set, and the best fitted results are chosen with respect to the Standard SVM without reducing any of the data, as will be later explained in Section 5. Among the 10 data sets used in this study, the best performance is obtained with a linear kernel for 3 of them and with a radial basis kernel for 7 of them. The details are summarized in the Table 3.

## 4. A Clique Clustering Algorithm

In this section, the "Clique Clustering Algorithm" (CC-Algorithm) is introduced in Algorithm 1, and a detailed example is given for demonstrating the steps of the algorithm. We note here that the proposed CC-Algorithm starts with the most similar pair of examples to construct the cliques. Thus, the cliques obtained naturally represent highly dense cliques.

**Notation 1.** *The following notations and assumptions are being used in the algorithm below, with some conventions to simplify the algorithm and its implementation. Let C denote the set of all attributes for the given data set.*

*B is a list of index pairs ordered in the decreasing order with respect to the similarity function* $sim_C(x_i, x_j)$*, such that, for* $k < \ell$*,*

$$\begin{aligned} &\text{either, } sim_C(x_{k_1}, x_{k_2}) > sim_C(x_{\ell_1}, x_{\ell_2}), \\ &\quad \text{or, } sim_C(x_{k_1}, x_{k_2}) = sim_C(x_{\ell_1}, x_{\ell_2}), \text{ and } k_1 > l_1 \\ &\quad \text{or, } sim_C(x_{k_1}, x_{k_2}) = sim_C(x_{\ell_1}, x_{\ell_2}), k_1 = l_1, \text{ and } l_2 > k_2 \end{aligned}$$

*where* $B[k] = (k_1, k_2)$ *and* $B[\ell] = (l_1, l_2)$*, and* $B[m]$ *denotes the m-th element of the ordered list, B. Hence, for example,* $B[1]$ *denotes the index* $(i, j)$*, where* $sim_C(x_i, x_j)$ *is the largest. The same ordering is also used for sorting the list A in Step 2.(c) of the algorithm.*

*Moreover, to avoid repetitions, we assume without loss of generality that* $i < j$ *for any* $(i, j) \in B$*. Furthermore, empty list is denoted by* $[\,]$*.*

The clique clustering algorithm is presented below. Algorithm 1.

**Algorithm 1:** Clique Clustering (CC) Algorithm.

**Input:** $B, t, sim_C(x_i, x_j)$.
**Output:** List of Cliques, $L$.

1. Let $B$ denote the ordered list set of all pairs $(i, j)$ with $sim(x_i, x_j) \geq t$ for a given threshold $t$, and this list is ordered by the similarity values from largest to smallest.
2. Set $L := [\,]$ and $B_0 := B$.
   Repeat until $B_0 = [\,]$
   \# The below sub-algorithm will repeat until all pairs in the list $B$ have been considered. #\
   (a) Let $(i, j) = B_0[1]$ be the first element.
   \# I.e., the pair $(i, j)$ such that $sim(x_i, x_j)$ is the largest value. #\
   (b) Consider the index sets $A_i = \{k \mid (i, k) \in B_0 \text{ or } (k, i) \in B_0\}$ and $A_j$ similarly, and then form a list from intersection of $A_i$ and $A_j$; $A = [k : k \in A_i \cap A_j]$.
   \# $A_i$ is the index set of vertices $x_k$ where there is an edge between $x_i$ and $x_k$; similarly, $A_j$ is the index set of vertices $x_k$ where there is an edge between $x_j$ and $x_k$,, and hence, $A$ is the list of indices $k$ such that $\{x_i, x_j, x_k\}$ forms a clique. #\
   (c) Sort $A$ as an ordered list by similarity values from largest to smallest according to $sim(x_i, x_k)$ for $i < j$ and $k \in A$.
   \# In order to obtain the densest cliques, we will check each element of $A$ one by one, after checking whether the remaining elements are included in the clique or not. #\
   (d) Set $A_0 := A$
   Repeat until $A_0 = [\,]$
   \# The below sub-routine will repeat until all indices in the list $A_0$ have been considered. #\
   - Let $k = A_0[1]$ be the first element.
     \# Densest clique construction: As conducted before, $k \in A_0$ with largest similarity value of $sim(x_i, x_k)$. #\
   - $A_0 :=$ Remove$(A_0, [k])$.
   - $A_k := \{\ell \in A \mid (k, \ell) \in B_0 \text{ or } (\ell, k) \in B_0\}$.
     \# Find all vertices $x_\ell, \ell \in A_0$, such that $sim(x_k, x_\ell) \geq t$. Recall that $B_0$ is the set of index pairs $(k, \ell)$ for which $sim(x_k, x_\ell) \geq t$. Hence, either $(k, \ell) \in B_0$ or $(\ell, k) \in B_0$. #\
   - Find, if any, vertices $x_\ell$ such that there is no edge between $x_k$ and $x_\ell$.
     $D := [\ell : \ell \in A \wedge \ell \notin A_k]$
     \# If there exists such $\ell \in D$, any set of vertices containing $\{x_k, x_\ell\}$ cannot form a clique. Thus, we should remove such elements from $A$ for the clique under construction. #\
   - Update $A$ and $A_0$ as $A :=$ Remove$(A, D)$ and $A_0 :=$ Remove$(A_0, D)$.
     \# The set $A$ is reduced by removing some elements. In order to check the remaining elements for forming a clique, we also remove them from the iteration set $A_0$ and repeat with the next element. #\
   - Next \# until $A_0 = [\,]$ #\.

   (e) $L :=$ Append$(L, [A])$ update the list $L$.
   \# A clique is obtained, $A$, and added to the list of cliques. Now, we will remove from $B_0$ any pair associated with clique vertices. #\
   (f) For each $k \in A$, set
   $R_k = \{(k, \ell) \mid (k, \ell) \in B_0\} \cup \{(\ell, k) \mid (\ell, k) \in B_0\}$.
   (g) $R = \cup_{k \in A} R_k$.
   (h) $B_0 :=$ Remove$(B_0, [\ell : \ell \in R])$.
   (i) Next \# until $B_0 = [\,]$ #\.
3. Return $L$.

*An Example*

The following sample data were obtained from the Iris data set, selecting 60 rows, 20 from each class, randomly. Moreover, in order to demonstrate the clique algorithm above, only the first two attributes were used, see Figure 1.

For chosen attributes, $a_1, a_2$, the weights of each attribute are calculated from the sample and represented as a vector $w = (0.7419, 0.2581)$, as described in Section 2. Then, according to this weight vector, the similarities are calculated:

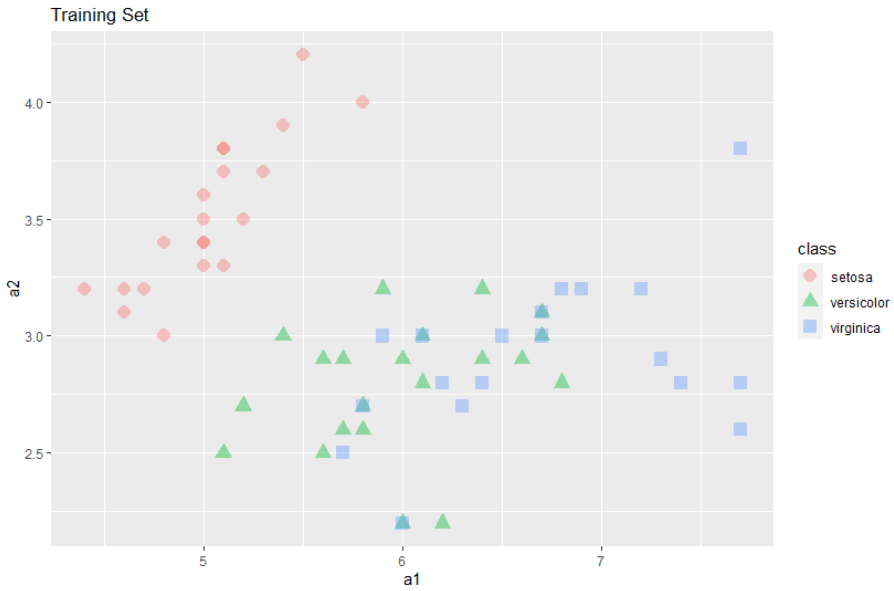



**Figure 1.** Sample data for demonstration.

| Seq. | Sim. | $i$ | $j$ |
|---|---|---|---|
| [1] | 1.0000 | 38 | 52 |
| [2] | 1.0000 | 36 | 56 |
| [3] | 1.0000 | 33 | 58 |
| | ⋮ | | |
| [1768] | 0.2065 | 14 | 48 |
| [1769] | 0.1865 | 14 | 44 |
| [1770] | 0.1865 | 14 | 45 |

After choosing a threshold, in this example, $t = 0.90$, we obtain only the first 255 rows from the above table:

| Seq. | Sim. | $i$ | $j$ |
|---|---|---|---|
| | ⋮ | | |
| [253] | 0.9001 | 29 | 39 |
| [254] | 0.9001 | 27 | 30 |
| [255] | 0.9001 | 8 | 23 |

Performing the CC-Algorithm described above, we obtain the cliques;

| Clq. | $i_1$ | $i_2$ | $i_3$ | $i_4$ | $i_5$ | $i_6$ | $i_7$ | $i_8$ |
|---|---|---|---|---|---|---|---|---|
| [1] | 38 | 52 | 30 | 34 | 51 | 60 | | |
| [2] | 36 | 56 | 25 | 28 | 35 | 39 | 40 | 43 |
| [3] | 33 | 58 | 22 | 26 | 32 | 55 | 57 | 59 |
| ⋮ | ⋮ | | | | | | | |
| [10] | 23 | 41 | | | | | | |
| [11] | 42 | 50 | | | | | | |
| [12] | 3 | 12 | | | | | | |

Among 60 examples, 55 are listed in the first 255 rows above. With the clique algorithm, these 55 example are separated into 12 disjoint subsets. Among 12 cliques, 7 of them are

homogeneous, i.e., all examples belonging to a clique have the same class type. The remaining five cliques are heterogeneous, all having examples exactly from two different classes. In Figure 2, all 12 cliques are shown with different colors, and the remaining 5 data (55 out of 60 are already in cliques) are marked as Clique 0 for completeness and simplicity.

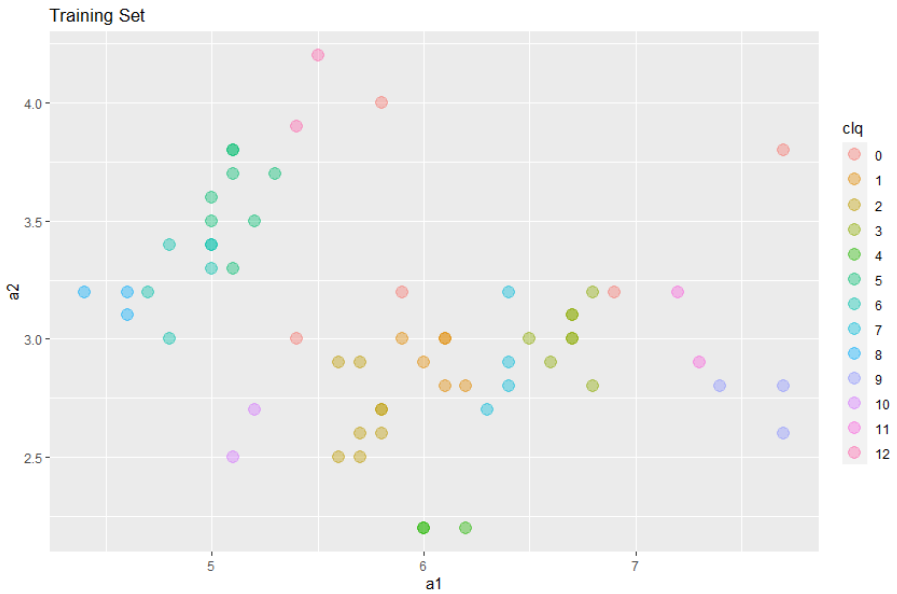



**Figure 2.** Clusters for sample data.

## 5. The Proposed Classification Algorithms

The idea of using clustering as an intermediate step before applying learning algorithms is an active research topic. For example, [11] proposed KMSVM, which combines k-means clustering with SVMs. In [14], a well-known clustering algorithm (CURE) is applied to discover the structure of the training data set before applying SVMs. In this study, we propose yet another approach by using the proposed CC-Algorithm. Different from the previous studies in the literature, the proposed approach uses the concept of complete cliques to obtain clusters of the training data set.

The main idea of the proposed classification algorithms is to discover structural information of the data by using the proposed clique clustering approach. We expect that this will reduce the training data set size while at the same time improving the classification algorithm.

Although one may consider different strategies to achieve this goal, in this study, we considered three basic strategies. In the first strategy, the centers of the cliques are used for training the SVM. This algorithm will be called the *Centers of Cliques-SVM* (CC-SVM). In the second strategy, cliques contain only one class category, i.e., homogeneous cliques are removed and heterogeneous cliques are replaced with their centers before applying SVM. This approach will be called the *Homogeneous Cliques Removed-SVM* (HOM-R-SVM). In the third strategy, cliques contain more than one class category, i.e., heterogeneous cliques are removed and homogeneous cliques are replaced with their centers before applying SVM. This approach will be called the *Heterogeneous Cliques Removed-SVM* (HET-R-SVM).

All three approaches will reduce the original training data set size by some amount. This will also affect the selection of support vectors. It is expected that this will result in avoiding some of the over-fitting problems that may be encountered.

After applying the clique-clustering algorithm, the training set will be partitioned into two parts: the cliques denoted by $\mathcal{C}$ and the remaining elements denoted by $\mathcal{F}$. Assuming that there are $l$ cliques and $r$ remaining elements, we have $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_l\}$ and $\mathcal{F} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_r\}$, where $\boldsymbol{x}_j \in \boldsymbol{R}^d$.

### *5.1. Centers of Cliques (CC-SVM) Algorithm*

The basic idea in this approach is to use the cliques to reduce the data set by replacing each clique with its center. The main steps of this approach are as follows:

1. Apply the Clique Clustering Algorithm (CC-Algorithm in Section 4).
2. Find centers for each class category in each clique.
   (a) For homogeneous cliques, only one center is calculated.
   (b) For heterogeneous cliques, we further identify all classes appearing in the clique and calculate the centers of data having the same class label. Thus, we obtain a minimum of two centers for each clique.
   (c) Elements in $\mathcal{F}$ are also considered centers of cliques of size one.
3. Apply the SVM algorithm.

If we analyze our previous sample example, the number of examples is reduced to $5 + 7 + 2 \times 5 = 22$ data, corresponding to 5 data from $\mathcal{F}$; 7 data from homogeneous cliques, each clique is replaced with its center; and 10 data, 2 from each of the 5 heterogeneous cliques containing examples from two classes and within each such clique, and centers are calculated for each group of data that has the same class labels. All details are illustrated in Figure 3, where all orange arrows show a reduction within homogeneous cliques belonging to "setosa" class, and most green and blue arrows show a reduction within heterogeneous cliques; however, there is one homogeneous green clique and there are two homogeneous blue cliques. With 22 data, we obtain a better accuracy level when compared with the standard SVM using all 60 data.

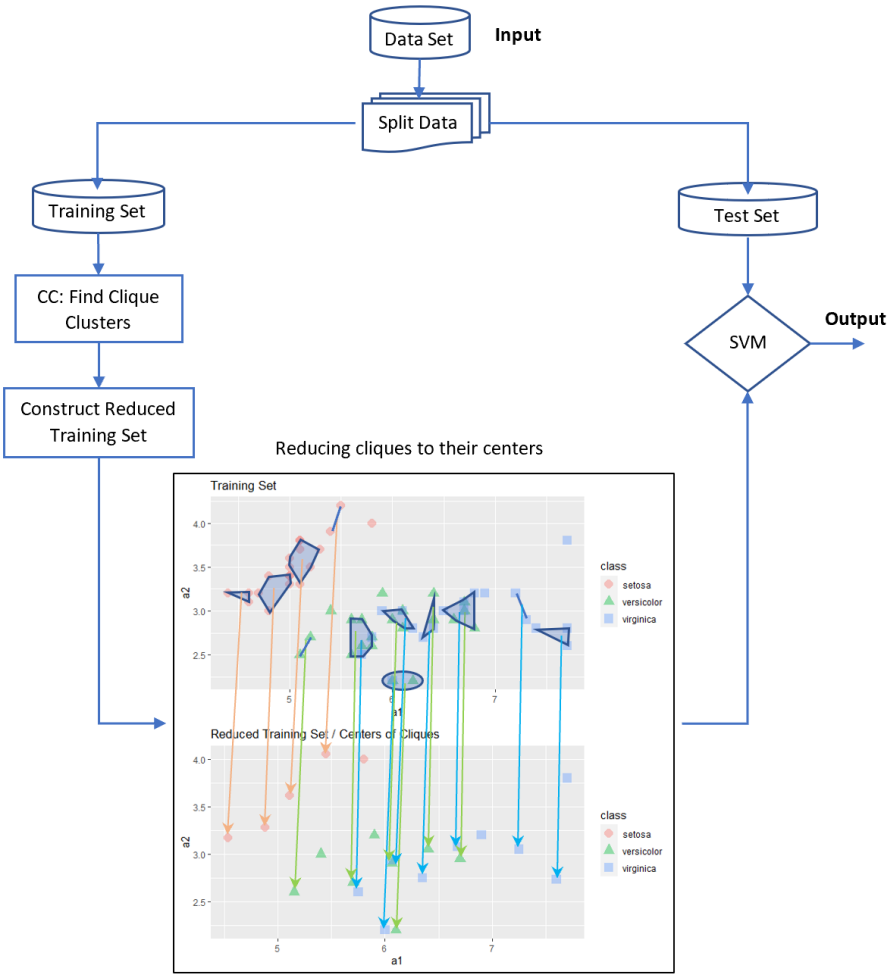



**Figure 3.** Flowchart of CC-SVM.

*5.2. Homogeneous Cliques Removed (HOM-R-SVM) Algorithm*

The basic idea of this approach is to reduce the data set by removing homogeneous cliques and replacing each heterogeneous clique with their centers. The main steps of this approach are as follows:

1. Apply the CC-Algorithm.
2. Remove cliques that contain only one class category.
   (a) Remove all data in homogeneous cliques.
   (b) For heterogeneous cliques, replace heterogeneous cliques with their centers. As in CC-SVM, calculate the centers of the data that have the same class labels.
   (c) The elements in $\mathcal{F}$ are also considered as the training data set.
3. Apply SVM algorithm.

For our sample example, the number of examples is reduced to $5 + 2 \times 5 = 15$ data only. All details are illustrated in Figure 4. Then, the standard SVM is applied only to these 15 data. The resulting accuracy level is comparable to the accuracy of the standard SVM with fewer examples.

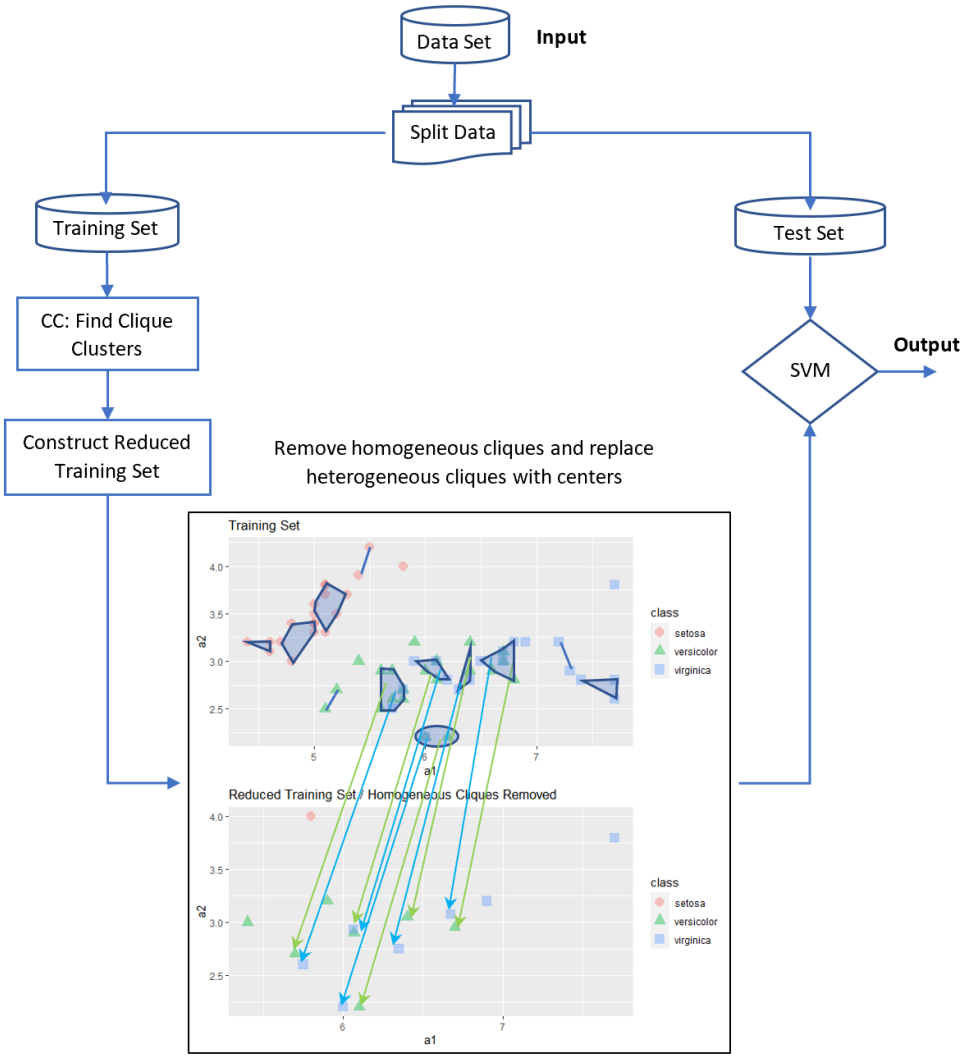



**Figure 4.** Flowchart of HOM-R-SVM.

*5.3. Heterogeneous Cliques Removed (HET-R-SVM) Algorithm*

The basic idea of this approach is to reduce the data set by removing heterogeneous cliques and replacing each homogeneous clique with its center. The main steps of this approach are as follows:

1. Apply the CC-Algorithm.
2. Remove cliques that contain more than one class categories.
   (a) For homogeneous cliques, replace any clique with its center.
   (b) For heterogeneous cliques, remove all data lying inside a clique that has more than one class category.
   (c) Elements in $\mathcal{F}$ are also considered as centers of cliques of size one.
3. Apply SVM algorithm.

Finally, for the sample example, the number of examples is reduced to $5 + 7 = 12$ data. All details are illustrated in Figure 5. The result is the same as the standard SVM with fewer examples.

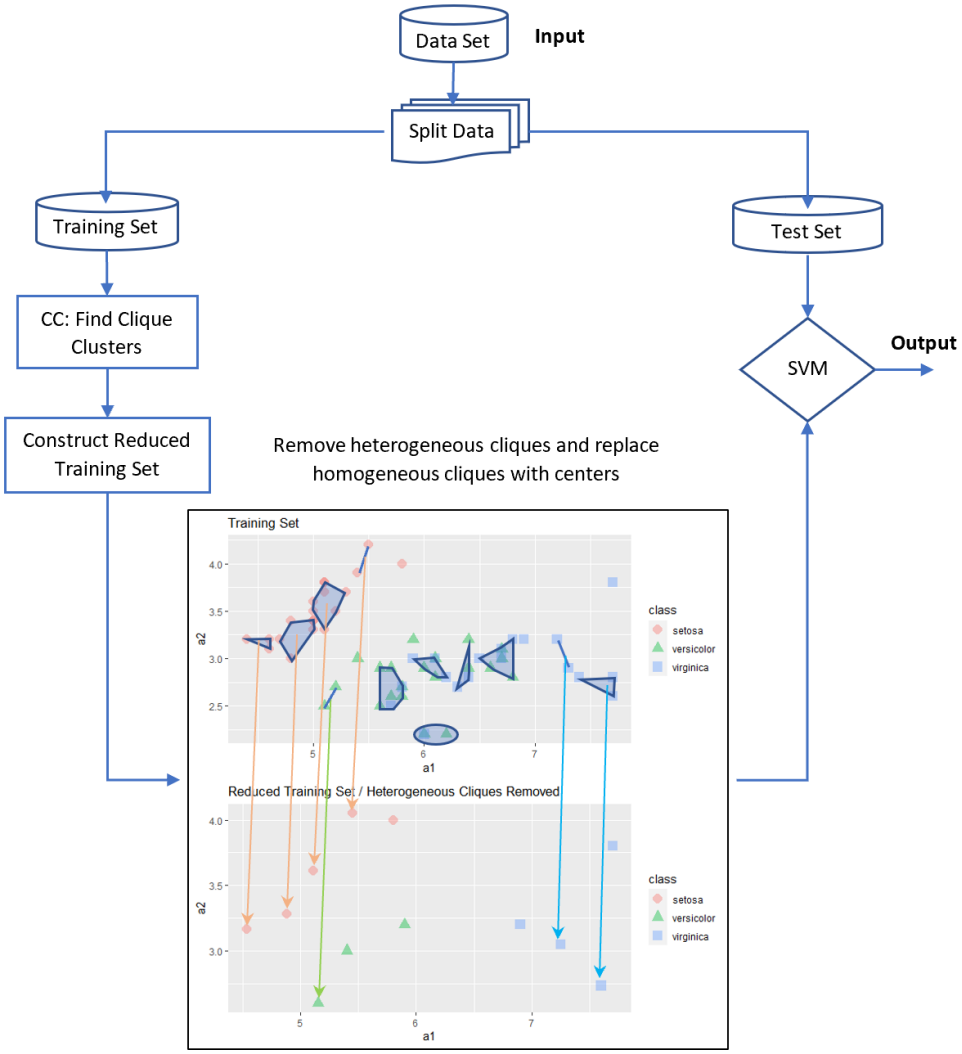



**Figure 5.** Flowchart of HET-R-SVM.

## 6. Application to Real Data Sets and Performance

In order to analyze the performance of the proposed classifiers, we considered 10 data sets that have also been used in some other studies (see, for example, [2,14,25]). We note that all data sets have numerical attributes only. The data sets represent different

application areas and various properties. Some of them have a relatively large number of class categories, whereas some have a relatively large number of examples, see Table 2.

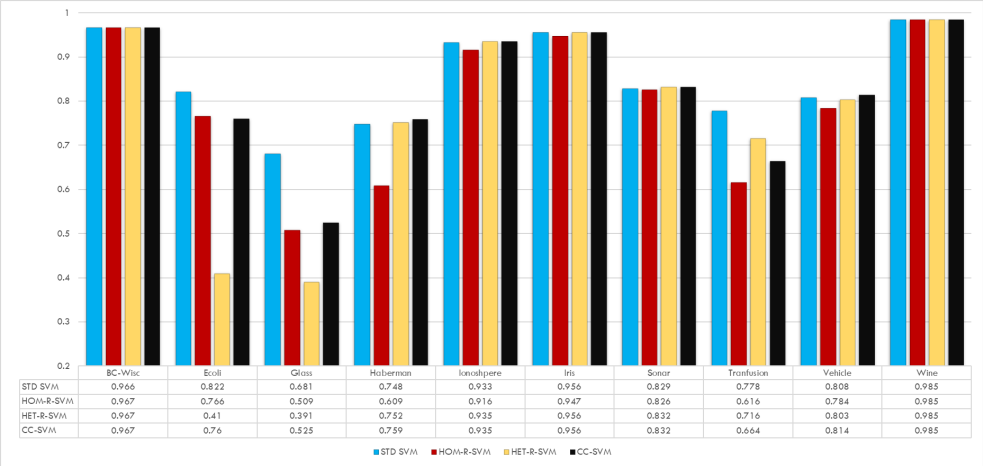
**Table 2.** Summary of Data Sets.

| Data Set | $n$ | $a$ | $c$ | Area |
|---|---|---|---|---|
| Iris | 150 | 4 | 3 | Life |
| Wine | 178 | 13 | 3 | Physical |
| Glass | 214 | 9 | 7 | Physical |
| Connectionist Bench (Sonar_All) | 208 | 60 | 2 | Physical |
| Statlog (Vehicle) | 846 | 18 | 4 | Physical |
| Ionosphere | 351 | 33 | 2 | Physical |
| Blood Transfusion | 748 | 4 | 2 | Business |
| Ecoli | 336 | 6 | 8 | Life |
| Haberman's Survival | 306 | 3 | 2 | Life |
| Breast Cancer Wisconsin | 683 | 9 | 2 | Life |

A few of the data sets needed some slight changes or adjustment before applying the algorithms. For example, 16 rows from "Breast Cancer Wisconsin" have missing values in one attribute, so for consistency, all these 16 rows have been deleted. For the "Ionosphere" data set, the second attribute is constant for all instances, so this attribute is completely removed before using the data. For the "Ecoli" data set, the fourth attribute (presence of charge) is almost constant, and all the columns are equal to 0.5, except for one row with a value of 1. Since train and test data are randomly chosen, for the cases where the data with the exceptional value of 1 for this attribute belong to test data, the SVM will report an error. For this reason, the fourth attribute is completely removed from this data set before running simulations.

Figure 6 shows the performance of all three proposed algorithms in terms of accuracy. In general, it is clear from the figure that the proposed classification algorithms achieve similar accuracy results as for the standard SVM with a reduced training data set size and fewer support vectors. For the Ecoli, Glass, and Blood Transfusion data sets a decrease in accuracy is observed. The degree of decrease depends on the particular algorithm. For example, for the Ecoli and Glass data sets, there is quite a decrease in accuracy for the HET-R-SVM algorithm. On the other hand, for the Blood Transfusion data set, the worst performance is obtained with the HOM-R-SVM algorithm. Finally, one can also see that the CC-SVM algorithm shows a more robust behavior compared to the other proposed SVM algorithms.

Instead of comparing the accuracy values of the test data for each data set, we provide detailed analysis of the results in Table 3. In this table, $k$ denotes the kernel as given in Table 1, $t$ denotes the threshold used in the clique algorithm in Section 4, $n$ denotes the number of the training data set for "Std. SVM" and the average number of reduced data of the corresponding SVM for five runs, whereas $n_s$ similarly denotes the number of support vectors for each SVM, and finally, "Acc." denotes the accuracy of the classification of the test data obtained for each SVM.




| | BC-Wisc | Ecoli | Glass | Haberman | Ionoshpere | Iris | Sonar | Tranfusion | Vehicle | Wine |
|---|---|---|---|---|---|---|---|---|---|---|
| STD SVM | 0.966 | 0.822 | 0.681 | 0.748 | 0.933 | 0.956 | 0.829 | 0.778 | 0.808 | 0.985 |
| HOM-R-SVM | 0.967 | 0.766 | 0.509 | 0.609 | 0.916 | 0.947 | 0.826 | 0.616 | 0.784 | 0.985 |
| HET-R-SVM | 0.967 | 0.41 | 0.391 | 0.752 | 0.935 | 0.956 | 0.832 | 0.716 | 0.803 | 0.985 |
| CC-SVM | 0.967 | 0.76 | 0.525 | 0.759 | 0.935 | 0.956 | 0.832 | 0.664 | 0.814 | 0.985 |

**Figure 6.** Best Accuracy chart for proposed SVMs.

The results in Table 3 show that, overall, the proposed algorithms achieve the same accuracy with a smaller training data set size and fewer support vectors. Only for the Wine and Sonar_All data sets, the reduction may be negligible, but this is indeed to achieve the best performance. For example, for the Wine data set, instead of $t = 0.95$, if $t = 0.90$ is considered, then $n_t = 30.6, n_s = 28$, Acc. $= 0.868$ for HOM-R-SVM; and $n_t = 59.4, n_s = 39.6$, Acc. $= 0.981$ for HET-R-SVM; and $n_t = 61, n_s = 41.2$, Acc. $= 0.981$ for CC-SVM. Similarly, for Sonar_All data set, instead of $t = 0.95$, if $t = 0.85$ is considered, then $n_t = 58.6, n_s = 57.4$, Acc. $= 0.761$ for HOM-R-SVM; and $n_t = 59.2, n_s = 55.2$, Acc. $= 0.797$ for HET-R-SVM; and $n_t = 77.2, n_s = 73$, Acc. $= 0.816$ for CC-SVM.

For the Transfusion data set, we decided to report two different threshold values, $t = 0.85$ and $t = 0.95$, because, for the former one, HOM-R-SVM and CC-SVM have better accuracy values for the test data; however, for the later one, HET-R-SVM performs much better.

In addition to the accuracy rates given in Table 3, Table 4 shows the true positive (TP) rate, false positive (FP) rate, and ROC areas for the data sets considered in this study. These results were calculated using the WEKA software [35] after obtaining the clique clusters with the proposed clustering algorithm. The results confirm the analysis given for Table 3. In general, almost the same performance as the standard SVM is obtained with the proposed algorithms using reduced training data sets in terms of TP and FP rates as well as ROC area.

The reduction in training size and number of support vectors is very high for most of the data sets. In addition, one can observe that the CC-SVM algorithm achieves comparable (sometimes even better) accuracy for all data sets compared to the standard SVM. On the other hand, HOM-R-SVM achieves the best results for the Iris and Breast Cancer Wisconsin data sets, whereas HET-R-SVM achieves the best results for the Vehicle and Haberman's Survival data sets. Details about the amount of reduction is given in Figures 7–9 for each algorithm separately for the data ratio versus accuracy ratio. In these charts, the following ratios are represented:

$$\text{Data ratio} = \frac{n_t \text{ for the proposed SVM}}{n \text{ for the standard SVM}},$$

and

$$\text{Accuracy ratio} = \frac{\text{Accuracy of test data for the proposed SVM}}{\text{Accuracy of test data for the standard SVM}}.$$

**Table 3.** The summary of SVM algorithms' simulations.

| Data Set | $k$ | $t$ | Std. SVM | | | HOM-R-SVM | | | HET-R-SVM | | | CC-SVM | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $n$ | $n_s$ | Acc. | $n_t$ | $n_s$ | Acc. | $n_t$ | $n_s$ | Acc. | $n_t$ | $n_s$ | Acc. |
| Iris | 1 | 0.95 | 105 | 22.8 | 0.956 | 20 | 12.6 | 0.947 | 38 | 14 | 0.956 | 41.2 | 16.6 | 0.956 |
| Wine | 2 | 0.95 | 125 | 61.8 | 0.985 | 112.6 | 59.2 | 0.985 | 118.8 | 60.4 | 0.985 | 118.8 | 60.4 | 0.985 |
| Glass | 2 | 0.95 | 150 | 132 | 0.681 | 39 | 38.8 | 0.509 | 23.4 | 22.2 | 0.391 | 46 | 44.6 | 0.525 |
| Sonar_All | 2 | 0.95 | 146 | 117 | 0.829 | 135.2 | 111.6 | 0.826 | 140.2 | 114.2 | 0.832 | 140.2 | 114.2 | 0.832 |
| Vehicle | 1 | 0.95 | 592 | 322 | 0.808 | 314.8 | 218.8 | 0.749 | 248.6 | 147.2 | 0.785 | 401.6 | 252.8 | 0.787 |
| Ionosphere | 2 | 0.95 | 246 | 100 | 0.933 | 132.2 | 82.8 | 0.916 | 158 | 83.6 | 0.935 | 162 | 88 | 0.935 |
| Transfusion | 2 | 0.85<br>0.95 | 524 | 266.8 | 0.777 | 15.6<br>63.2 | 15.6<br>59.8 | 0.662<br>0.621 | 3.2<br>22.6 | 2.4<br>16.2 | 0.462<br>0.719 | 17.6<br>74.6 | 17.2<br>65.4 | 0.671<br>0.661 |
| Ecoli | 1 | 0.90 | 235 | 94.4 | 0.822 | 15.2 | 14.4 | 0.766 | 4.8 | 4.0 | 0.410 | 15.6 | 14.4 | 0.760 |
| Haberman's Survival | 2 | 0.95 | 214 | 121.8 | 0.748 | 75.8 | 71.2 | 0.609 | 41.4 | 31.8 | 0.752 | 90.6 | 81.2 | 0.759 |
| Breast Cancer Wisconsin | 2 | 0.95 | 478 | 74.2 | 0.966 | 189.8 | 63.2 | 0.967 | 216.4 | 62.4 | 0.967 | 216.8 | 63.2 | 0.967 |

**Table 4.** TP rates, FP rates and AUC analysis.

| Data Set | $k$ | $t$ | Std. SVM | | | HOM-R-SVM | | | HET-R-SVM | | | CC-SVM | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | TP Rate | FP Rate | ROC Area | TP Rate | FP Rate | ROC Area | TP Rate | FP Rate | ROC Area | TP Rate | FP Rate | ROC Area |
| Iris | 1 | 0.95 | 0.978 | 0.009 | 0.984 | 0.956 | 0.018 | 0.969 | 0.956 | 0.018 | 0.969 | 0.978 | 0.009 | 0.984 |
| Wine | 2 | 0.95 | 0.981 | 0.011 | 0.985 | 0.981 | 0.011 | 0.985 | 0.981 | 0.011 | 0.985 | 0.981 | 0.011 | 0.985 |
| Glass | 2 | 0.95 | 0.625 | 0.143 | 0.741 | 0.516 | 0.127 | 0.694 | 0.422 | 0.219 | 0.601 | 0.531 | 0.141 | 0.695 |
| Sonar_All | 2 | 0.95 | 0.629 | 0.450 | 0.589 | 0.548 | 0.548 | 0.500 | 0.597 | 0.490 | 0.554 | 0.597 | 0.49 | 0.554 |
| Vehicle | 1 | 0.95 | 0.780 | 0.074 | 0.853 | 0.720 | 0.094 | 0.813 | 0.728 | 0.091 | 0.819 | 0.756 | 0.082 | 0.837 |
| Ionosphere | 2 | 0.95 | 0.924 | 0.146 | 0.889 | 0.914 | 0.098 | 0.908 | 0.924 | 0.146 | 0.889 | 0.924 | 0.146 | 0.889 |
| Transfusion | 1 | 0.95 | 0.790 | 0.754 | 0.518 | 0.714 | 0.472 | 0.621 | 0.692 | 0.539 | 0.577 | 0.580 | 0.372 | 0.604 |
| Ecoli | 1 | 0.94 | 0.788 | 0.072 | 0.858 | 0.768 | 0.054 | 0.855 | 0.712 | 0.154 | 0.735 | 0.663 | 0.082 | 0.790 |
| Haberman's Survival | 2 | 0.95 | 0.739 | 0.699 | 0.520 | 0.641 | 0.660 | 0.490 | 0.728 | 0.728 | 0.500 | 0.707 | 0.661 | 0.523 |
| Breast Cancer Wisconsin | 2 | 0.95 | 0.946 | 0.026 | 0.960 | 0.937 | 0.031 | 0.953 | 0.937 | 0.031 | 0.953 | 0.937 | 0.031 | 0.953 |

Figure 7 shows that, overall, CC-SVM achieves almost the same performance as the standard SVM with a reduced training data set. Only for the Glass data set a significant decrease in accuracy is observed, which is still the best among the three proposed algorithms. In addition, for the Ecoli, Iris, and Blood Transfusion data sets, the data reduction is very high with ratios of less than 8%.

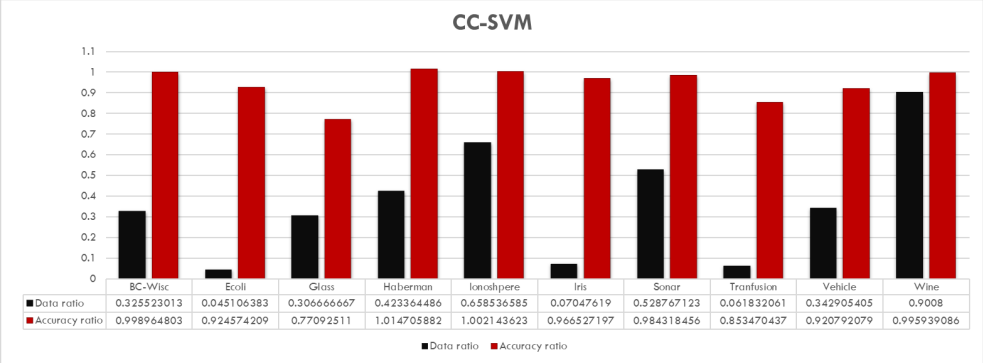



**Figure 7.** Performance of CC-SVM: data ratio vs. accuracy ratio.

Figure 8 shows that overall HOM-R-SVM achieves comparable performance to the standard SVM with a reduced training data set. Only for the Glass data set a significant decrease in accuracy is observed. In addition, for the Ecoli and Transfusion data sets, the data reduction is very high with ratios of less than 7%.

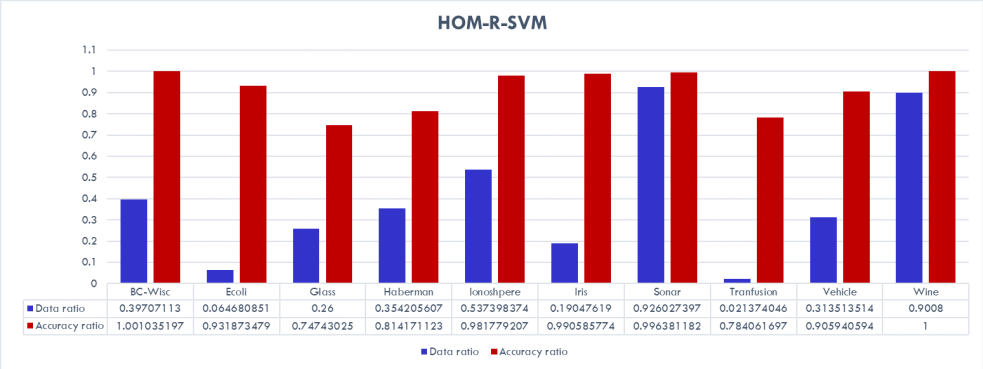



**Figure 8.** Performance of HOM-R-SVM: data ratio vs. accuracy ratio.

Figure 9 shows that, overall, HET-R-SVM achieves almost the same performance as the standard SVM with a reduced training data set, except for the Ecoli and Glass data sets. For these two data sets, a significant decrease in accuracy is observed. In addition, for the Ecoli, Haberman, and Transfusion data sets, the data reduction is very high with ratios of less than 6%.

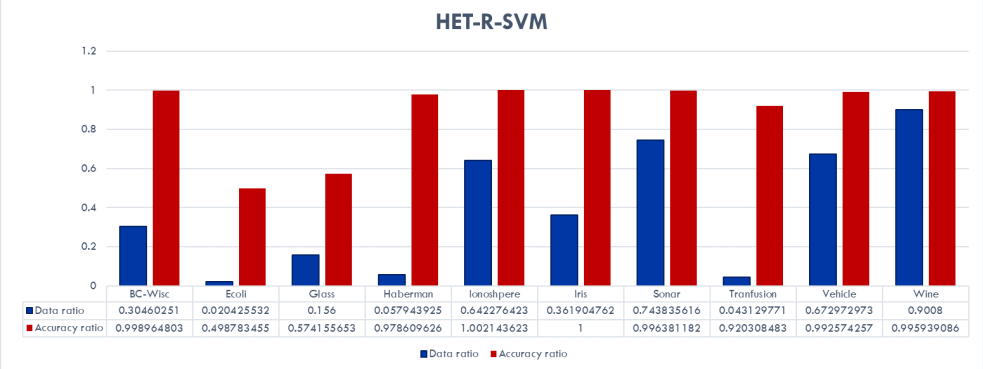



**Figure 9.** Performance of HET-R-SVM: data ratio vs. accuracy ratio.

These results also show that all three algorithms actually capture some different aspects of the data sets. For example, although the best performance (with an accuracy ratio of 1.02 and data reduction ratio of 0.42) for the Haberman data set is obtained with the CC-SVM algorithm, the HET-R-SVM shows a similar performance (with an accuracy ratio of 0.97) with only a fraction of 6% of the original training data set.

## 7. Discussion

The results show that for most of the data sets, the three algorithms show similar performances. For example, for the BC-Wisconsin and Wine data sets, all three algorithm show a similar performance and data reduction behavior. In contrast, for the Ecoli, Glass, and Transfusion data sets, the algorithms show some clear differences. For example, for the Ecoli data set, only the HET-R-SVM shows a significant decrease in accuracy compared to the standard SVM. For the Glass data set, all three algorithms achieve less accuracy compared to the standard SVM. For this data set, the best accuracy is achieved with the CC-SVM algorithm. For the Transfusion data set, the best accuracy is achieved with the HET-R-SVM algorithm.

In summary, one can see that for some data sets, all three algorithms show similar performances. Conversely, for some data sets, one of the algorithms shows the best performance. Thus, for some of the data sets, as expected, either HOM-R-SVM or HET-R-SVM may perform better than the usual scenario where CC-SVM performs better. The choice of the corresponding algorithms depends on the user's needs and the structure of the data set or classes.

For a detailed analysis, we run another set of simulations from $t = 0.80$ to $t = 0.99$ with a step size of 0.01. In all data sets, *CC-SVM* is the most consistent algorithm. Two graphs below show how a SVM algorithm may fluctuate according to the data set.

For the Iris data set, HET-R-SVM and CC-SVM are consistent for $0.80 \leq t \leq 0.99$, but HOM-R-SVM gives reasonable accuracy for larger values of $t$. With larger values of $t$, the ratio $n_t/n$ also increases (see Figure 10).

This behavior, "fluctuating", is not specific to HOM-R-SVM, as the Transfusion data set shows in Figure 10. HET-R-SVM does not give consistent accuracy values for $t < 0.90$, even though HET-R-SVM gives the best accuracy for $t \geq 0.94$. On the other hand, both HOM-R-SVM and CC-SVM are consistent for all values of $0.80 \leq t \leq 0.99$. Even though, the accuracy rates differ, as expected, the ratio $\frac{n_t}{t}$ is almost typical.

Both Figures 10 and 11 show that for smaller values of $t$, the number of examples used in SVM reduces significantly, whereas for larger values of $t$, more data are being used in SVM. Furthermore, depending on the distribution of the data, as seen for Transfusion data set in Figure 11, even for larger values of $t$, there is still a reduction in the size of data before using SVM. This is also similar for the number of support vectors.

In conclusion, we proposed a new clustering algorithm based on cliques that can be used to discover relevant information for classification algorithms. Three different strategies for using the information obtained by the clique clustering algorithm were investigated. The results based on 10 real data sets obtained from the UCI repository [36] show that comparable accuracy performance can be obtained by these proposed algorithms with a smaller training data set size and fewer support vectors. In addition, the results indicate that each proposed classification algorithm can capture different aspects of the training data sets.

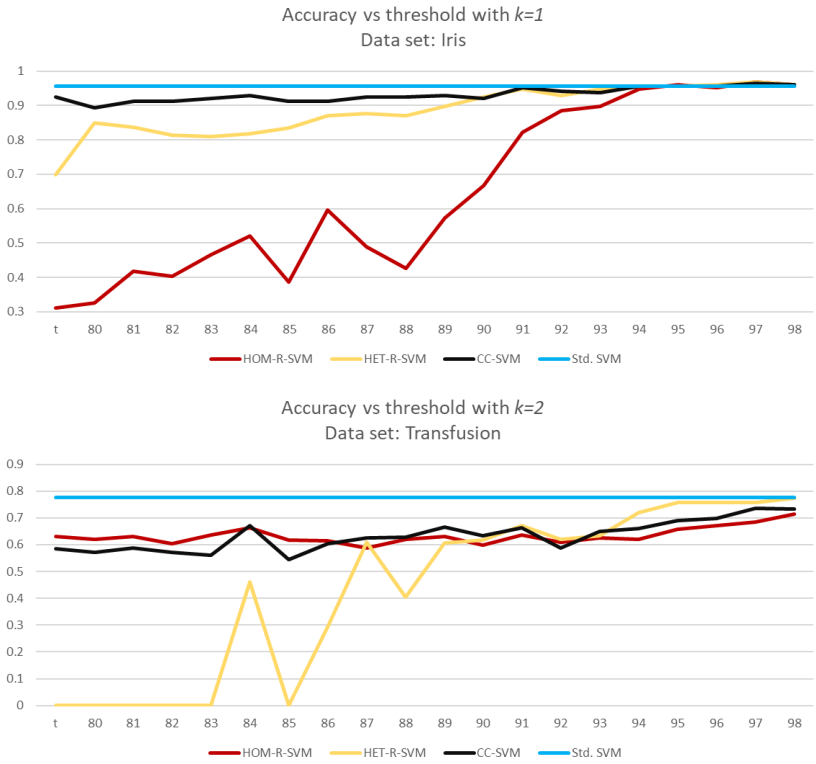



**Figure 10.** Analysis for accuracy values of Iris and Transfusion data sets.

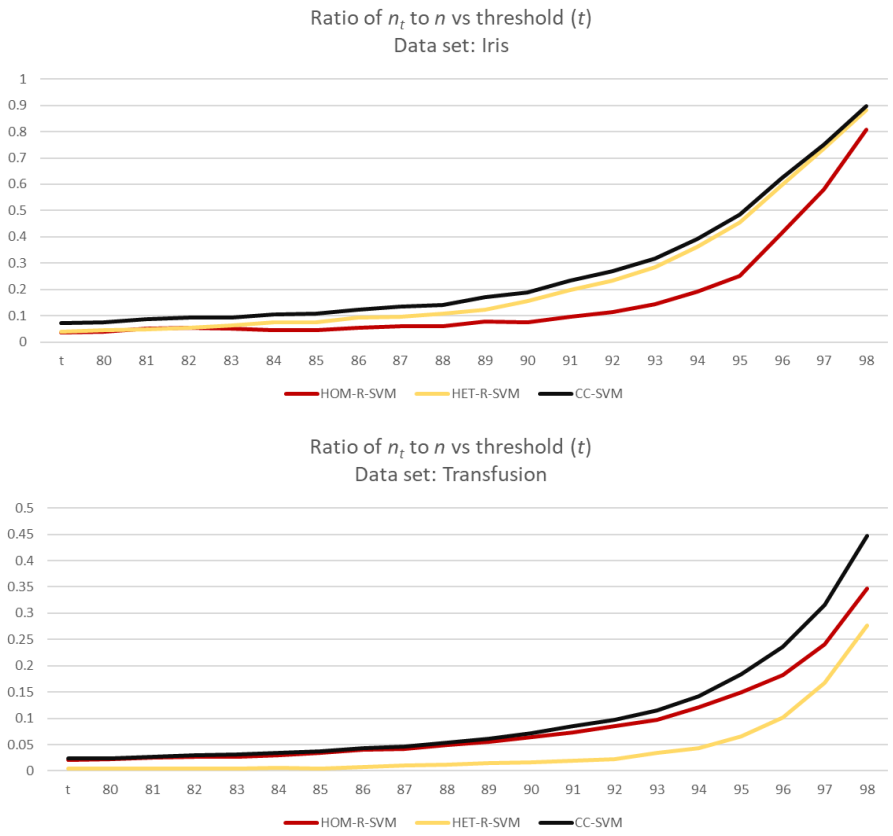



**Figure 11.** Analysis for ratio of reduction for Iris and Transfusion data sets.

It is known that SVMs can have some problems with large data sets. For example, the performance of SVMs decrease significantly when the number of examples in the training data set increases ([37]). The results of this study show that the proposed classification algorithms are of potential use for investigating new classification strategies based on SVMs that can be used with large data sets.

**Author Contributions:** Data curation, G.A., U.M. and D.S.; Formal analysis, G.A., U.M. and D.S.; Funding acquisition, U.M. and D.S.; Investigation, U.M. and D.S.; Methodology, G.A., U.M. and D.S.; Project administration, U.M.; Resources, G.A., U.M. and D.S.; Software, G.A. and U.M.; Validation, U.M. and D.S.; Visualization, G.A., U.M. and D.S.; Writing—original draft, G.A., U.M. and D.S.; Writing—review and editing, G.A., U.M. and D.S. All authors contributed equally to this work. 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:** All the simulation results are available upon request from any of the authors.

**Acknowledgments:** The data sets used in this article have been obtained from UCI [36]. The citations [38,39] have been added upon the data provider's requests. The authors greatly acknowledge ideas provided by anonymous referees that were used to enrich this manuscript and/or to continue these ideas in new research directions. Particularly, the algorithm have been explained in more detail, and the AUC results have been added upon the referees' requests. More improvements with comparisons regarding deep learning will be considered in our next manuscript.

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

## References

1. Alon, N.; Krivelevich, M.; Sudakov, B. Finding a Large Hidden Clique in a Random Graph. In Proceedings of the SODA '98: Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, San Francisco, CA, USA, 25–27 January 1998; pp. 594–598. [CrossRef]
2. Arslan, G.; Karabulut, B.; Ünver, H.M. On Using Structural Patterns in data for classification. *Adv. Appl. Stat.* **2020**, *65*, 33–56. http://dx.doi.org/10.17654/AS065010033. [CrossRef]
3. Yang, W.; Xia, K.; Li, T.; Xie, M.; Song, F. A Multi-Strategy Marine Predator Algorithm and Its Application in Joint Regularization Semi-Supervised ELM. *Mathematics* **2021**, *9*, 291. [CrossRef]
4. Yoshida, T. A graph-based approach for semisupervised clustering. *Comput. Intell.* **2014**, *30*, 263–284. j.1467-8640.2012.00450.x. [CrossRef]
5. Ames, B.P.W. Guaranteed clustering and biclustering via semidefinite programming. *Math. Program.* **2014**, *147*, 429–465. [CrossRef]
6. Ames, B.P.W.; Vavasis, S.A. Convex optimization for the planted k-disjoint-clique problem. *Math. Program.* **2014**, *143*, 299–337. [CrossRef]
7. Ames, B.P.W.; Vavasis, S.A. Nuclear norm minimization for the planted clique and biclique problems. *Math. Program.* **2011**, *129*, 69–89. [CrossRef]
8. Burges, C.J.C. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.* **1998**, *2*, 121–167. [CrossRef]
9. Vapnik, V.N. *Statistical Learning Theory*; John Wiley and Sons: New York, NY, USA, 1998.
10. Vapnik, V.N. *The Nature of Statistical Learning Theory*, 2nd ed.; Springer: Stanford, CA, USA, 2008. [CrossRef]
11. Wang, J.; Wu, X.; Zhang, C. Support vector machines based on K-means clustering for real-time business intelligence systems. *Int. J. Bus. Intell. Data Min.* **2005**, *1*, 54–64. [CrossRef]
12. Chen, S.G.; Wu, X.J. Multiple birth least squares support vector machine for multi-class classification. *Int. J. Mach. Learn. Cyber.* **2017**, *8*, 1731–1742. [CrossRef]
13. Cheng, H.; Tan, P.N.; Jin, R. Efficient algorithm for localized support vector machine. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 537–549. [CrossRef]
14. Karabulut, B.; Arslan, G.; Ünver, H.M. Classification Based on Structural Information in Data. *Arab. J. Sci. Eng.* **2021**, *59*, 1–15. [CrossRef]
15. Jain, A.K. Data clustering: 50 years beyond K-means. *Pattern Recognit. Lett.* **2010**, *31*, 651–666. [CrossRef]
16. Cooley, R.; Mobasher, B.; Srivastava, J. Data Preparation for Mining World Wide Web Browsing Patterns. *Knowl. Inf. Syst.* **1999**, *1*, 5–32. [CrossRef]

17. Punj, G.; Stewart, D.W. Cluster Analysis in Marketing Research: Review and Suggestions for Application. *J. Mark. Res.* **1983**, *20*, 134–148. [CrossRef]
18. Ben-Dor, A.; Shamir, R.; Yakhini, Z. Clustering Gene Expression Patterns. *J. Comput. Biol.* **2004**, *6*, 3–4. [CrossRef]
19. Cutting, D.R.; Karger, D.R.; Pedersen, J.O.; Tukey, J.W. Scatter/Gather: A cluster-based approach to browsing large document collections. In Proceedings of the SIGIR '92: Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Association for Computing Machinery, Copenhagen, Denmark, 21–24 June 1992; pp. 318–329. [CrossRef]
20. Jain, A.K.; Flynn, P.J. Image segmentation using clustering. In *Advances in Image Understanding: A Festschrift for Azriel Rosenfeld*; Ahuja, N., Bowyer, K., Eds.; IEEE Press: Piscataway, NJ, USA, 1996; pp. 65–83.
21. Chapelle, O.; Schölkopf, B.; Zien, A. (Eds.) *Semi-Supervised Learning*; MIT Press: Cambridge, MA, USA, 2006; pp. 1–13.
22. Chen, Y.; Jalali, A.; Sanghavi, S.; Xu, H. Clustering Partially Observed Graphs via Convex Optimization. *J. Mach. Learn. Res.* **2014**, *15*, 2213–2238.
23. Prinen, A.; Ames, B. Exact Clustering of Weighted Graphs via Semidefinite Programming. *J. Mach. Learn. Res.* **2019**, *20*, 1–34.
24. Luce, R.D.; Perry, A.D. A method of matrix analysis of group structure. *Psychometrika* **1949**, *14*, 95–116. [CrossRef]
25. Kayaalp, N.; Arslan, G. A Fuzzy Bayesian Classifier with Learned Mahalanobis Distance. *Int. J. Intell. Syst.* **2014**, *29*, 713–726. [CrossRef]
26. Vapnik, V.; Chervonenkis, A. *Theory of Pattern Recognition [in Russian]*; Nauka: Moscow, Russia, 1974.
27. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning*, 2nd ed.; Springer: Stanford, CA, USA, 2008. [CrossRef]
28. Schölkopf, B.; Smola, A.J. *Learning with Kernels*, 1st ed.; The MIT Press: London, UK, 2001.
29. Smola, A.J.; Schölkopf, B. A tutorial on support vector regression. *Stat. Comput.* **2004**, *14*, 199–222. [CrossRef]
30. Cheng H.; Tan, P.N.; Jin, R. Localized Support Vector Machine and Its Efficient Algorithm. In Proceedings of the 2007 SIAM International Conference on Data Mining (SDM), Radisson University Hotel Minneapolis, Minnesota, MN, USA, 26–28 April 2007. [CrossRef]
31. Jayadeva; Khemchandani, R.; Chandra, S. Twin Support Vector Machines for Pattern Classification. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 905–910. [CrossRef] [PubMed]
32. Meister, M.; Steinwart, I. Optimal learning rates for localized SVMs. *J. Mach. Learn. Res.* **2016**, *17*, 6722–6765.
33. Rastogi, R.; Safdari, H.; Sharma, S. Exploring Data Reduction Techniques for Time Efficient Support Vector Machine Classifiers. In Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI), Bangalore, India, 18–21 November 2018; pp. 2053–2059. [CrossRef]
34. Dimitriadou, E.; Hornik, K.; Leisch, F.; Chang, C.-C.; Lin, C.-C. Package 'e1071'. R Software Package. Available online: https://cran.r-project.org/web/packages/e1071 (accessed on 14 August 2021).
35. Frank, E.; Hall, M.A.; Witten, H. *The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*, 4th ed.; Morgan Kaufmann: Burlington, MA, USA, 2016.
36. Dua, D.; Graff, C. UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences. Available online: http://archive.ics.uci.edu/ml (accessed on 10 December 2021).
37. Almasi, O.N.; Rouhani, M. Fast and de-noise support vector machine training method based on fuzzy clustering method for large real-world datasets. *Turk. J. Elec. Comp.* **2016**, *24*, 219–233. [CrossRef]
38. Mangasarian, O.L.; Street, W.N.; Wolberg, W.H. Breast Cancer Diagnosis and Prognosis via Linear Programming. *Oper. Res.* **1995**, *43*, 548–725. [CrossRef]
39. Yeh, I.; Yang, K.; Ting, T. Knowledge discovery on RFM model using Bernoulli sequence. *Expert Syst. Appl.* **2009**, *36*, 5866–5871. [CrossRef]